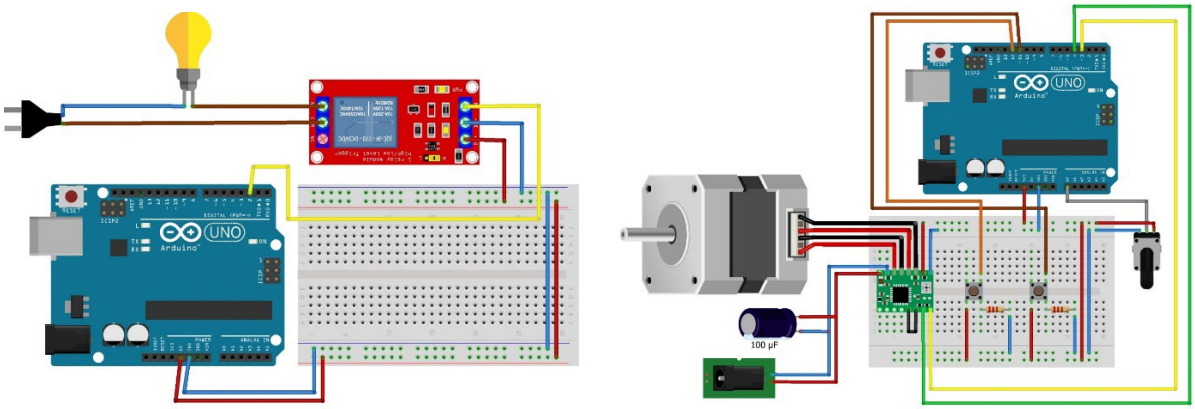
JOŠKO SMOLČIĆ



Arduino

Repetitorij s laboratorijskim vježbama

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Split, veljača 2024.

Sadržaj

1. ARDUINO PLATFORMA

Arduino je tehnologija otvorenog koda čija je platforma zasnovana na fleksibilnim, lakim za korištenje, hardveru i softveru. Namijenjen je za umjetnike, dizajnere, i sve zainteresirane koji žele stvarati interaktivne objekte ili okruženje. Arduino platforma je u stalnom razvoju i do danas postoji više od dvadesetak Arduino mikrokontrolerskih pločica za različitu namjenu. Osnovni model koji je ovdje obrađen je Arduino Uno R3, iako je Arduino danas prisutan već u svojoj četvrtoj izvedbi Arduino Uno R4 koja nudi više brzine, memorije, konektora i mogućnosti povezivanja od bilo koje verzije ploče prije. Arduino proizvodi, odlikuju se jednostavnošću korištenja. Za programiranje Arduina koristi se za tu svrhu prilagođeni C++ programski jezik.

1.1 Arduino Uno R3

Slika 1 Arduino UNO R3

Arduino Uno je mikrokontrolerska ploča zasnovana na Atmega328 mikrokontroleru. Ima 14 digitalnih izlazno /ulaznih pinova od kojih se 6 može koristiti kao PWM izlazi, 6 analognih ulaza, 16 MHz keramički otpornik, USB priključak, priključak za napajanje, ICSP zaglavlje, i tipku za reset. Sadržava sve što je potrebno kao podrška mikrokontroleru; jednostavno spajanje na računalo sa USB kabelom te mogućnost napajanja sa AC/DC adapterom ili baterijama.

Za razliku od prethodnih arduino ploča Arduino Uno ne koristi FTDI USB čip za serijsku komunikaciju. Umjesto toga Atmega 16u2 je programiran kao USB-serijski pretvarač.

Atmega 328 ima kapacitet memorije 32 KB (sa 0,5 KB iskoristivih pri podizanju). Također još ima i 2 KB SRAM memorije i i 1 KB EEPROM memorije (može biti čitana i zapisivana sa EEPROM Library)

Tehničke karakteristike

Parametar	Vrijednost
Mikrokontroler	ATmega328
Radni napon	5V
Ulazni napon (preporučeno)	7-12V
Ulazni napon (ograničenje)	6-20V
Digitalni I/O Pinovi	14 (od čega 6 omogućava PWM izlaz)
Analogni Input Pinovi	6
DC Struja za I/O Pin	40 mA
DC Struja za 3.3V Pin	50 mA
Flash Memorija	32 KB (ATmega328) od čega se 0.5 KB koristi pri učitavaju i podizanju sustava
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Brzina	16 MHz

2. POČETNI KORACI S ARDUINOM

Za uspješno korištenje Arduina potrebno je potrebno je obaviti nekoliko početnih koraka a to su slijedećih devet:

1. **Izbor Arduino pločice te načina povezivanja i napajanja.**

Slika 2 Arduino UNO R3 i kabel za spajanje

Povezivanje Arduino pločice s računalom vrši se USB kabelom. Ovisno o samoj vrsti pločice koju koristimo razlikuje se i vrsta USB kabela. Za osnovnu Arduino Uno R3 pločicu koristi se standardni USB kabel (A konektor na B konektor) što je vidljivo na slici iznad.

2. Arduino softver

Upravljački softver olakšava pisanje Arduino koda i njegovo upisivanje u Arduino pločicu. Postoje posebne verzije softvera za Windows, Mac OS X i Linux. Trenutno posljednja verzija za Windows je Arduino IDE 2.2.1 Upravljački softver možete pronaći na službenoj Arduino stranici http://www.arduino.cc/ na direktnom linku http://arduino.cc/en/Main/Software
Softver se nalazi u ZIP formatu te je potebno napraviti „unzip“ nakon čega dobijete datoteku za instalaciju softvera.

3. Spajanje Arduino pločice

Osnovna Arduino Uno R3 pločica kao i većina drugih arduino pločica ima automatsko napajanje preko USB kabela. Na samoj Arduino pločici postoji i dodatni konektor za eksterno napajanje. Mogući ulazni napon je od 6V do 20V. Proporučeni napon je od 7V do 12V.

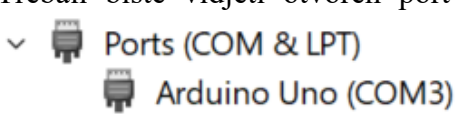
Nakon spajanja pločice postoji zelena (PWR) LED dioda na pločici koja signalizira napajanje te treba konstantno svijetliti.

4. Instalacija drivera

Nakon instalacije Arduino IDE softvera spajanje Arduino pločice bi trebalo biti potpuno automatizirano (plug and play). Ipak postoji način da to napravite korak po korak ako se za to pokaže potreba.

Spojite vašu pločicu USB kabelom s računalom zatim:

- Kliknete desnim klikom na **Start Menu**  i otvorite **Device Manager**.
- Do **Device Managera** možete doći i ako kliknete na **Control Panel.** Tada otvorite **System and Security.** Zatim kliknete na **System** te tada otvorite **Device Manager**.
- Pogledate pod Portove (COM & LPT). Ukoliko nemate COM & LPT, pogledajte pod **"Other Devices"** te **"Unknown Device".**
- Trebali biste vidjeti otvoren port pod nazivom "Arduino UNO (COMxx)".



- Zapamtite koji je COM port pridružen vašoj Arduino pločici, te napravite desni klik na taj port te izaberite opciju **"Update Driver Software"**
- Zatim izaberete opciju **"Browse my computer for Driver software"**
- U konačnici pronađete driver nazvan **"arduino.inf"**, koji je lociran u **"Drivers"** folderu unutar prethodno skinutog Arduino softvera, te kliknite na taj driver te će zatim Windowsi dovršiti instalaciju drivera.

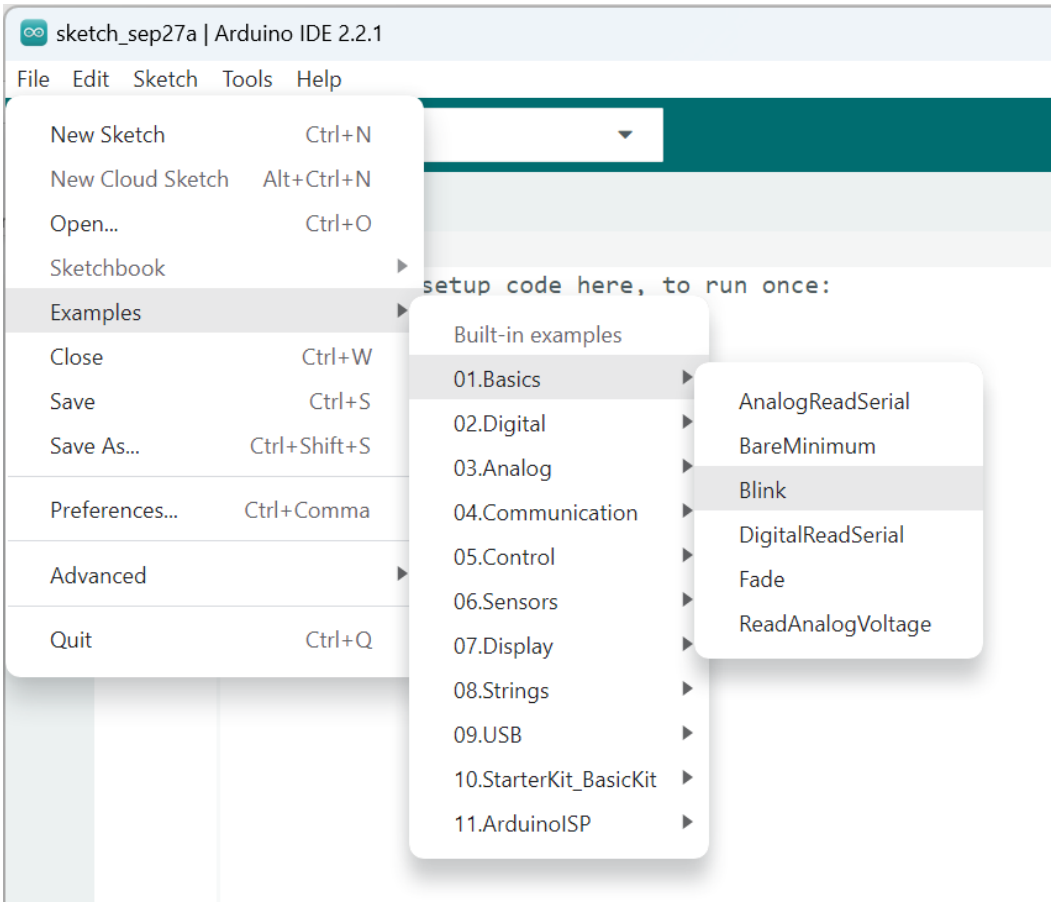
5. Pokretanje Arduino programa

Arduino program pokrećete dvostrukim klikom na Arduino ikonu 

koja se nalazi na radnoj površini računala.

6. Otvaranje primjera programa

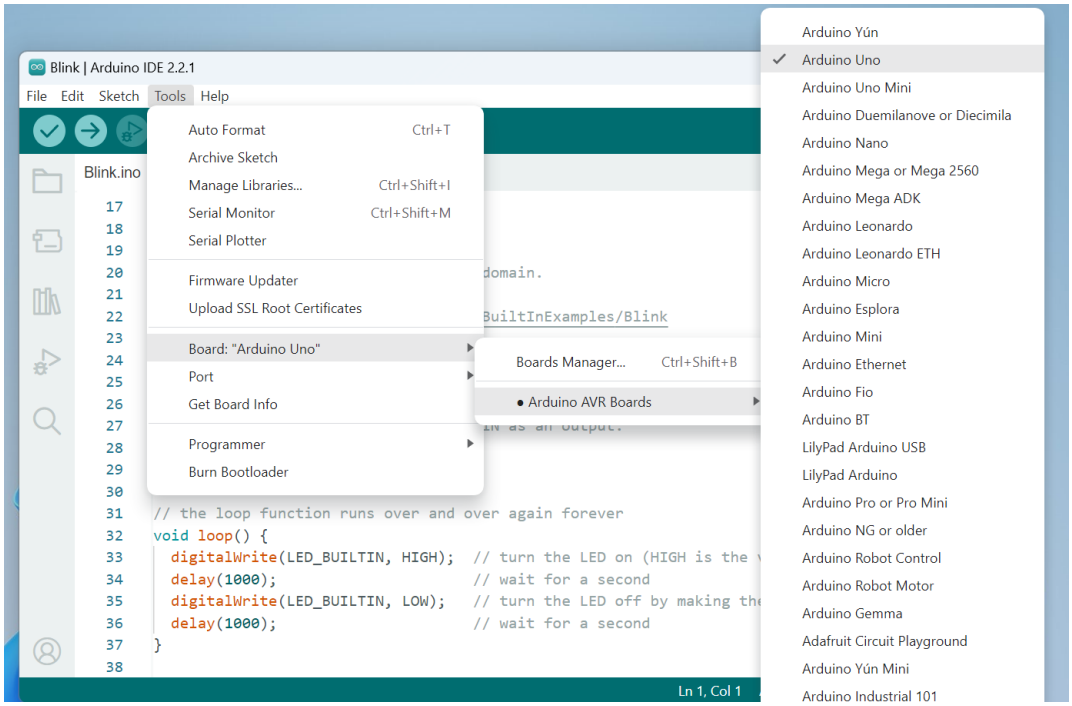
Otvorite jedan od primjera već napisanih programa koji se nalaze unutar Arduino aplikacije: **File / Examples / 1.Basics / Blink**.



Slika 3 File / Examples / 1.Basics / Blink.

7. Izaberite vašu Arduino pločicu

U izborniku **Tools/Bord/ArduinoAVR boards** izaberete svoju Arduino pločicu.



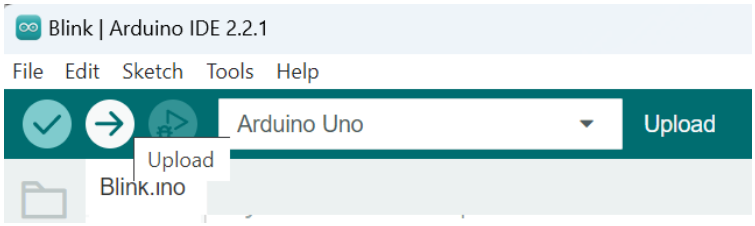
Slika 4 Tools/Bord/ArduinoAVR boards

8. Izaberete serijski COM port

Unutar Arduino aplikacije u padajućem izborniku izaberete ispravan COM port za vašu Arduino pločicu (onaj koji ste zapamtili prilikom instalacije Arduino programa). U većini slučajeva to budu veći brojevi poput COM3 ili COM4 ovisno o količini uređaja ili Arduino pločica koje imate instalirane na vašem računalu. Ukoliko niste sigurni koji je COM port od vaše Arduino pločice, isključite vašu Arduino pločicu te pogledate koji od portova više nije ponuđen. Tada znate koji je COM port od te pločice koju ste isključili. Ponovno uključite pločicu i izaberite taj port.

9. Upišite program u Arduino pločicu

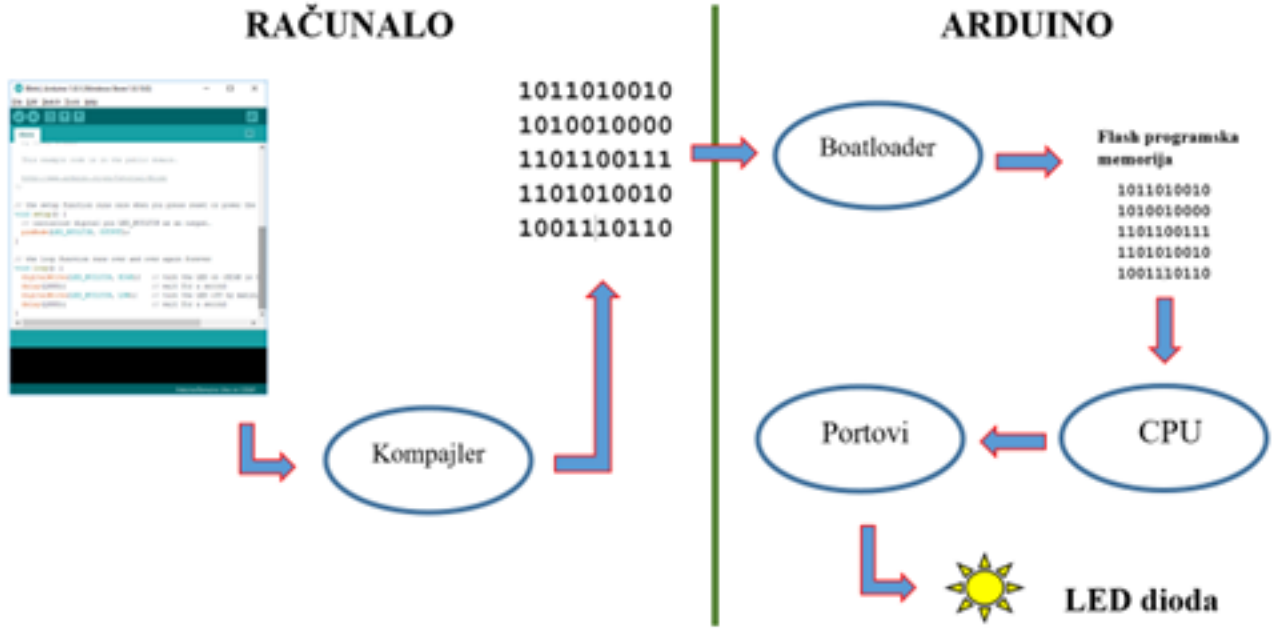
Jednim klikom na **"Upload"** tipku (strelica koja pokazuje desno) izvršit će te upisivanje programa u Arduino pločicu. Tijekom upisivanja RX i TX LED diode na ploči će ubrzano blinkati. Ako je upisivanje uspješno na dnu programskog dijaloga dobit će te poruku **"Done uploading."**



Slika 5 Učitavanje

10. Način rada Arduino programa.

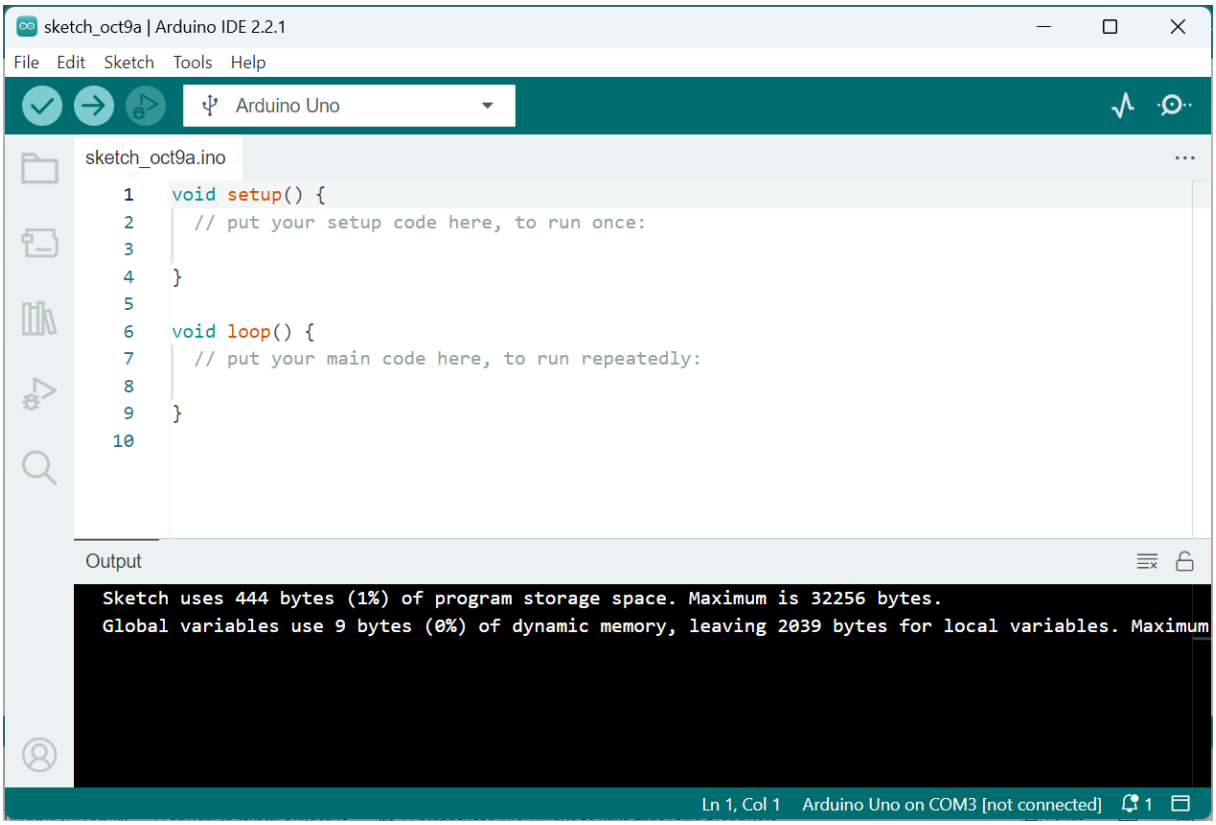
U ovom dijelu će te naučiti kako radi program, te što se događa kad pokrenete program. Put koji se prolazi od pisanja programa, prevođenja, učitavanja te pokretanja programa prikazan je na slici ispod.



Slika 6 Kompletne procedura izrade programa

Kada pritisnete "Upload" tipku na Arduino IDE tada pokrenete lanac događaja koji na kraju rezultiraju time da vaš kod bude instaliran na Arduino pločicu te pokrenut. U prvom koraku odvija se nešto što zovemo kompajliranje. Kompajliranje je pretvaranje koda koji ste vi napisali u strojni kod koji Arduino može razumjeti. Ako kliknete na ovu tipku  u Arduino programu vi zapravo kompajlirate kod bez slanja programa u Arduino pločicu. Na taj način vršite provjeru ispravnosti napisanog koda u skladu s pravilima C/C++ programskih jezika. Ukoliko u Aduino program napišete nešto što nije u skladu s C/C++ programskim jezicima kompajler će vam javiti grešku. Svaki program mora sadržavati dvije osnovne funkcije, a to su **void setup**() i **void loop**().

Ukoliko pokušate kompajlirati kod sa samo te dvije funkcije te izvršite provjeru, kompajler će vam poslati poruku da je kompajlirenje gotovo "Done compiling" i dati vam podatak o veličini programa koji ste kompajlirali. U ovom slučaju 444 bytes, Slika ispod.



Slika 7 Veličina programa u bajtovima

3. UVODNE VJEŽBE

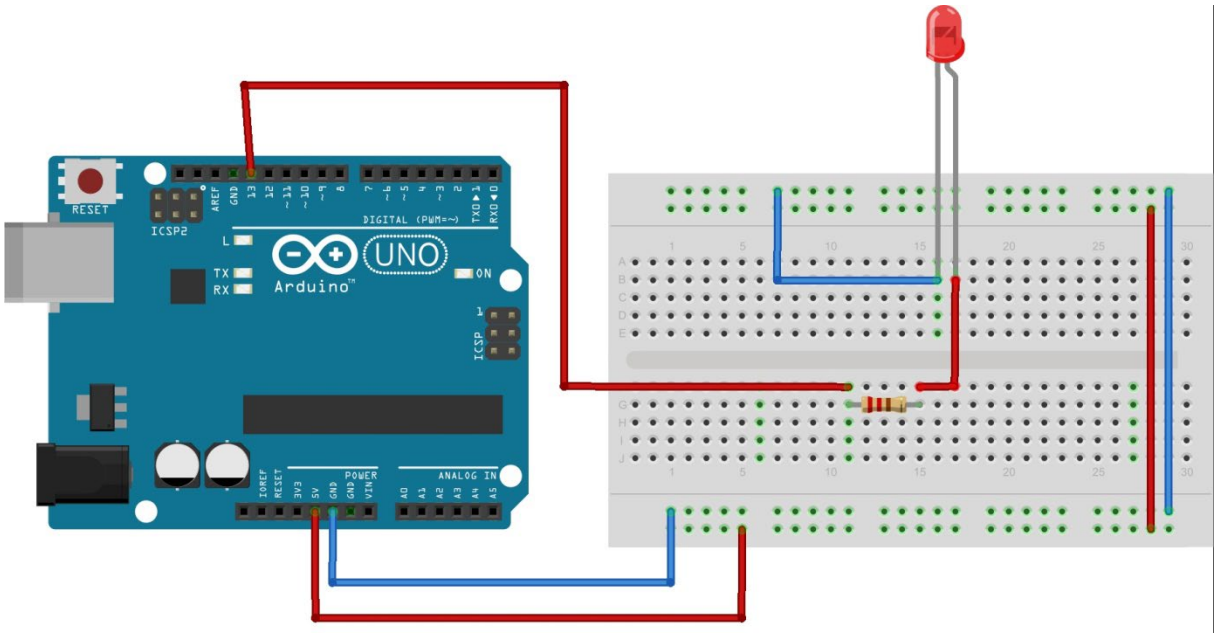
3.1 VJEŽBA br. 1 „Blink“

Zadatak 1.

Spojiti Arduino pločicu s računalom te učitati “Blink” program u mikrokontroler. Zatim spojiti LED diodu i napajanje Arduina kako je prikazano na slici te potom odgovoriti na pitanja.

Kod

```
/*
  Blink
Ovaj program uključuje LED diodu u vremenskom trajanju od jedne sekunde,
zatim se dioda gasi u vremenskom trajanju od jedne sekunde, nakon toga
program počinje ponovno s paljenjem LED diode.
 */
// Odabrati ćemo Pin 13 kao pin koji će davati napon na LED diodu.
// nazvat ćemo ga led
int led = 13;

// osnovna funkcija koja sadrži postavke programa
void setup()
{
  // postavljanje digitalnog pina 13 kao izlaznog.
  pinMode(led, OUTPUT);
}
// funkcija u kojoj se nalazi radnja programa koja se neprestano ponavlja.
void loop()
 {
  digitalWrite(led, HIGH);// upali LED (HIGH znači da ima napona) level)
  delay(1000);  // sačekaj jednu sekundu
  digitalWrite(led, LOW); // ugasi LED (LOW, znači da nema napona na pinu 13
)
delay(1000); // sačekaj jednu sekundu
}
```



Slika 8 „Blink“ shema spajanja

Pitanja?

1. Koje su dvije osnovne funkcije unutar „Blink“ programa i čemu služe?
2. Koja je uloga `pinMode` i `digitalWrite` funkcija?
3. Kako povećati dužinu svijetljenja diode za dvije sekunde, a smanjiti razmak između paljenja diode za pola sekunde?
4. Kako radi `delay` naredba?

Odgovori:

1. Dvije osnovne funkcie su void setup() i void loop() . Unutar void setup() funkcije se upisuju postavljanja koja vrijede u cijelom programu. Unutar void loop() programa se upisuje izvršni dio programa.
2. Uloga pinMode() funkije je definiranje pinova. U ovom slučaju definirali smo pin 13 kao izlazni. Uloga digitalWrite() funkcije je omogućavanje napona na određenoj varijabli. U ovom slučaju smo omogućili te zatim onemogućili napon na varijabli „led“ kojoj je pridružen pin 13.
3. Nakon linije koda `digitalWrite(led, HIGH);` povećati kašnjenje za 2000 milisekundi `delay(3000);`

te nakon linije koda `digitalWrite(led, LOW);` smanjiti kašnjenje za 500 milisekundi `delay(500);`

4. Naredba `delay()` zaustavlja izvršavanje programa te ostaje vrijediti posljednja naredba koja je bila prije `delay()` naredbe. Nakon što istekne vrijeme koje je zadano u naredbi program se nastavlja od iste točke.

Zadatak 2.

Napišite, kompajlirajte te učitajte sljedeći kod te odgovorite na pitanja:

```
void setup()
{
pinMode( 13, OUTPUT);
digitalWrite (13, HIGH);
}
void loop ()
{
}
```

Pitanja:

1. Kada se izvršava funcija `void setup()` i zašto LED dioda spojena na pin broj 13 stalno svijetli?
2. Što radi funkcija `pinMode( 13, OUTPUT);`?
3. Što radi funkcija `digitalWrite (13, HIGH);` ?

Odgovori:

1. Funkcija **setup()** se izvršava samo kod prvog pokretanja programa to je ujedno i razlog zašto pin broj 13 na Arduinu stalno svijetli.
2. Funkcija **pinMode(13, OUTPUT)** postavlja pin 13 kao izlaz (OUTPUT)
3. funkcija **digitalWrite(13, HIGH)** postavlja ovaj izlaz u visoku naponsku razinu (5V).

Zadatak 3.

Napišite, kompajlirajte te učitajte sljedeći kod te slijedite daljnje upute:

```
void setup()
{
pinMode( 13, OUTPUT);

}
void loop ()
{
digitalWrite (13, HIGH);
delay (500);
digitalWrite (13, LOW);
}
```

Unutar funkcije **loop()** kod se ciklički izvršava, program se vrti unutar petlje. Kada pokrenemo ovaj kod vidjeti ćemo da se ne događa točno ono što bi smo željeli. LED svijetli skoro cijelo vrijeme. Zašto?
Prođimo kroz kod liniju po liniju da bi smo vidjeli zašto je to tako.

1. Pokrećemo **setup** i namještamo pin 13 da bude izlaz (**OUTPUT**)
2. Pokrećemo **loop** i namještamo pin 13 da stalno svijetli (**HIGH**)
3. Kašnjenje u trajanju od pola sekunde
4. Namještamo pin 13 da bude ugašen bez napona (**LOW**)
5. Pokrećemo **loop** ponovno u drugom čitanju i opet namještamo pin 13 da ima napona (**HIGH**) te on opet svijetli.

Problem leži između 4. i 5. koraka. Učitavanje programa se događa tako brzo da se čini da LED dioda skoro stalno svijetli. Mikrokontrolerski čip na Arduinu izvodi 16 miliona instrukcija u sekundi. To ne znači 16 miliona komandi u C i C++ jeziku, ali je jako brzo. Iz tog razloga naša LED dioda će biti isključena samo nekoliko milijuntih dijelova sekunde.
Za ispravak ovog problema moramo u kod dodati još jedno kašnjenje nakon što ugasimo LED diodu.

Naš kod bi tada trebao izgledati ovako:

```
void setup()
{
pinMode( 13, OUTPUT);

}
void loop ()
{
digitalWrite (13, HIGH);
delay (500);
digitalWrite (13, LOW);
delay (500);
}
```

Kad ponovno učitate ovaj nadopunjeni kod vidjeti će te da vaša LED dioda blinka otprilike jednom u sekundi.

3.2 VJEŽBA br. 2 „Button“ & „millis()“

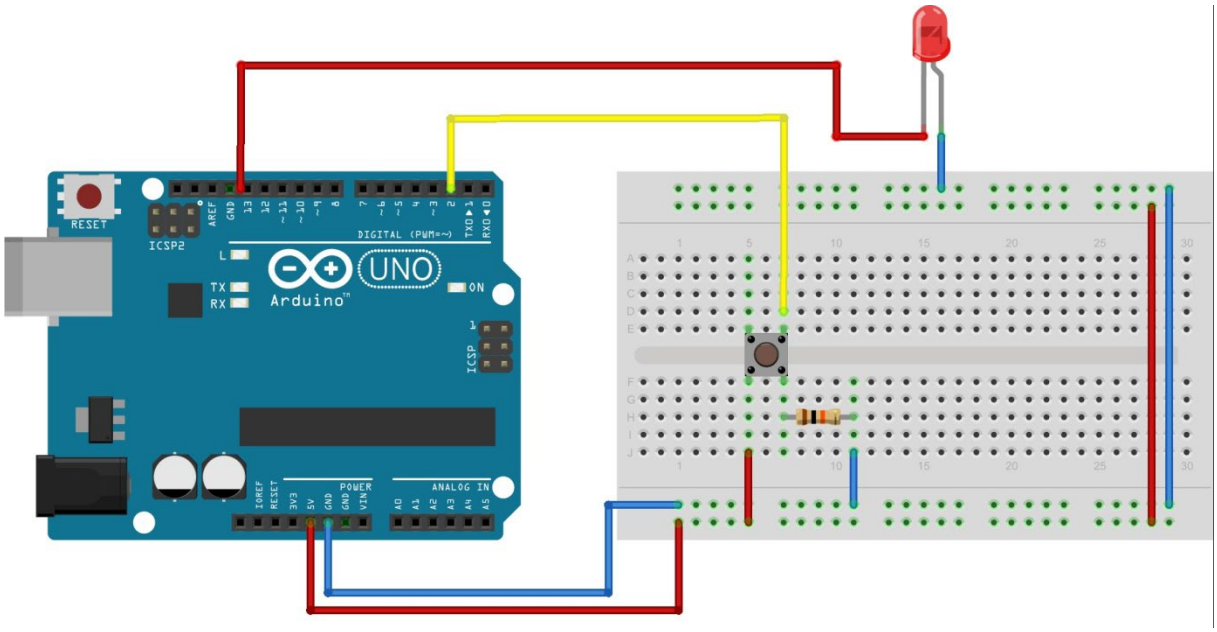
Zadatak 1

Spojiti Arduino pločicu s računalom te učitati “Button” program u mikrokontroler. Zatim spojiti LED diodu i napajanje Arduina kako je prikazano na slici te potom odgovoriti na pitanja.

Kod

```
/*
  Button

Pali i gasi LED diodu koja je spojena na digitalni pin 13,  u trenutku kad
se pritisne tipku koja je spojena na pin 2.


 Krug:
 * LED dioda spojena sa pina 13 na masu
 * Tipka spojena na pin 2 sa napona od +5V
 * 10K otpornik spojen između pina 2 i mase

 kreirao 2005
 DojoDave <http://www.0j0.org>
 modificirao 30 Aug 2011
  Tom Igoe

 Ovaj primjer je objavljen na:
 http://www.arduino.cc/en/Tutorial/Button
 */

// cons je varijabla koja je konstantna, nepromjenjiva i ovdje se koristi za
postavljanje pinova :
const int buttonPin = 2;     // postavljanje pina od tipke
const int ledPin =  13;      // postavljanje pina od LED diode

// int varijable su cjelobrojne promjenjive i ovdje je koristimo za
postavljanje stanja tipke:
int buttonState = 0;         // varijabla za čitanje statusa tipke

void setup() {
  // postavlja LED pin kao izlazni:
  pinMode(ledPin, OUTPUT);
  // postavljanje pina od tipke kao ulaza:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // čitanje stanja na pinu od tipke:
  buttonState = digitalRead(buttonPin);
```

```
  // provjera dali je tipka pritisnuta.
  // ako je onda je stanje tipke HIGH:
  if (buttonState == HIGH) {
    // upali LED diodu:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // ugasi LED diodu:
    digitalWrite(ledPin, LOW);
  }
}
```



Slika 9 „Button“ Shema spajanja

Pitanja?

1. Koji pin na Arduino pločici je postavljen kao ulazni?
2. Što se događa kad pritisnemo tipku?
3. Zašto između tipke i mase stavljamo otpornik od 10kΩ?
4. Kakvu liniju koda treba dodati u program i gdje, ako želimo da nam LED dioda svijetli neko vrijeme i nakon što pritisnemo i pustimo tipku?

Odgovori:

1. **Pin broj 2 je postavljen kao ulazni a to je onaj koji dobije signal s tipke.**
2. **Kad pritisnemo tipku napon od 5V sprovedemo na pin 2. Nakon toga mikrokontroler očita stanje na pinu 2 kao HIGH (što znači ima napona) te u skladu s programom pusti napon na izlazni pin 13.**
3. **Otpornik od 10kΩ stavljamo iz razloga da bi umanjili kratki spoj između mase i napona od 5V prilikom pritiskanja tipke. U slučaju kad ne bi bilo otpornika može doći do resetiranja mikrokontrolera na Arduino pločici.**
4. **Potrebno je dodati kašnjnje, delay() u „if" petlji neposredno nakon paljenja diode.**

```
if (buttonState == HIGH)
{
        // upali LED diodu:
        digitalWrite(ledPin, HIGH);
        delay(1000);

}
```

Zadatak 2

U zadatku broj 2 za razliku od zadatka broj 1, imamo dvije tipke. Također program mora omogućiti da prilikom pritiskanja tipke LED dioda svijetli neko zadano vrijeme i potom se gasi. Treba postojati mogućnost da se dvije LED diode upale u bilo kojem trenutku pa tako i u trenutku dok jedna od njih već svijetli. U tom slučaju više ne možemo koristiti naredbu `delay();` iz razloga što ona zaustavlja izvršavanje programa. Iz tog razloga ovdje je primijenjena funkcija `millis()`.

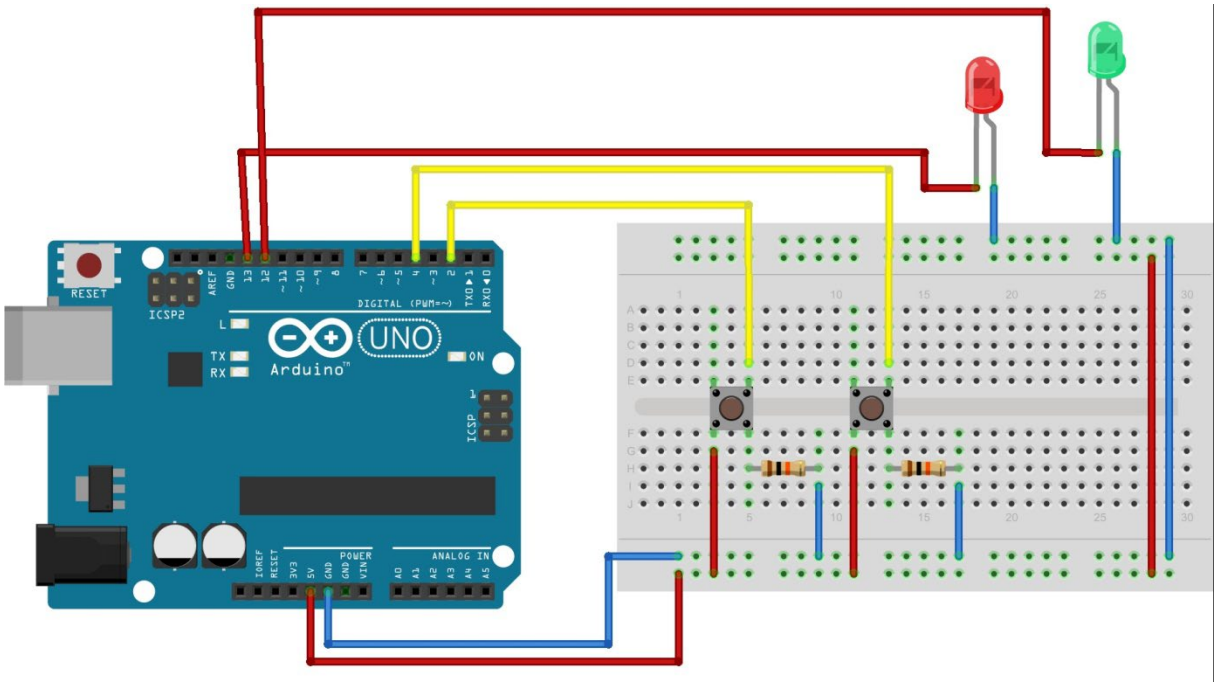
Spojiti Arduino pločicu s računalom te učitati "Button &millis()" program u mikrokontroler. Zatim spojiti LED diodu i napajanje Arduina kako je prikazano na slici te potom odgovoriti na pitanja.

Kod

```
/*Button & millis()
Program omogućuje paljenje dvije LED diode nevezano jedne s drugom i zadano
vremensko svijetljenje diode nakon pritiska tipke.
Krug:
 * LED dioda spojena sa pina 13 na masu
 * Tipka spojena na pin 2 sa napona od +5V
 * 10K otpornik spojen između pina 2 i mase
 * LED dioda spojena sa pina 12 na masu
 * Tipka spojena na pin 4 sa napona od +5V
 * 10K otpornik spojen između pina 4 i mase
kreirao 2014
Joško Smolčić */
// prva tipka i LED dioda ( pridruživanje pinova)
const int  buttonPin = 2;
const int ledPin = 13;

//druga tipka i LED dioda (pridruživanje pinova)
const int  buttonPin2 = 4;
const int ledPin2 = 12;

//varijable potrebne za korištenje millis() funkcije
unsigned long start; //za prvu tipku
unsigned long kraj;
unsigned long start2; //za drugu tipku
unsigned long kraj2;

//stanje prve tipke i prve LED diode
unsigned long buttonState = 0;
unsigned long ledState =0;

//stanje druge tipke i druge LED diode
unsigned long buttonState2 = 0;
unsigned long ledState2 =0;
```

```
void setup()
{
  Serial.begin(9600); //pokretanje serijske komunikacija

//za prvu tipku
  pinMode(buttonPin, INPUT);  //postavljanje varijable buttonPin kao ulaza
za prvu tipku
   pinMode(ledPin, OUTPUT); // postavljanje varijable ledPin kao izlaza za
prvu LED diodu
     buttonState = digitalRead(buttonPin); //stanje koje bude na buttonPin
varijabli pridružit će se buttonState varijabli.
     ledState = digitalRead(ledPin);//stanje koje bude na ledPin varijabli
pridružit će se ledState varijabli.

//za drugu tipku
  pinMode(buttonPin2, INPUT);  //postavljanje varijable buttonPin kao ulaza
za drugu tipku
  pinMode(ledPin2, OUTPUT); // postavljanje varijable ledPin kao izlaza za
drugu LED diodu
     buttonState2 = digitalRead(buttonPin2); //stanje koje bude na
buttonPin2 varijabli pridružit će se buttonState2 varijabli.
     ledState2 = digitalRead(ledPin2);//stanje koje bude na ledPin2
varijabli pridružit će se ledState2 varijabli.

}
    void loop()
{
 //_________PRVA TIPKA_____________________________

  buttonState = digitalRead(buttonPin);
  ledState = digitalRead(ledPin);

  if(buttonState == LOW)
  {
    start= millis();
  }
   if (buttonState == HIGH)
   {
  digitalWrite(ledPin, HIGH);
   delay(100);
   kraj = millis();
   delay(100);
   }
   if(ledState ==HIGH)
   if (start-kraj >= 5000UL)
   {
   digitalWrite(ledPin, LOW);
```

```
   }
   //prikaz na Serial Monitoru za prvu tipku
    Serial.print("Trenutno stanje millisa na startu =  ");
   Serial.println(start);
    Serial.print("Trenutno stanje millisa na kraju =  ");
   Serial.println(kraj);
   //_____________DRUGA TIPKA____________________
   buttonState2 = digitalRead(buttonPin2);
  ledState2 = digitalRead(ledPin2);

  if(buttonState2 == LOW)
  {
    start2= millis();
  }
   if (buttonState2 == HIGH)
   {
  digitalWrite(ledPin2, HIGH);
   delay(100);
   kraj2 = millis();
   delay(100);
   }
   if(ledState2 ==HIGH)
   if (start2-kraj2 >= 5000UL)
   {
   digitalWrite(ledPin2, LOW);
   }
   //prikaz na Serial Monitoru za drugu tipku
    Serial.print("Trenutno stanje millisa na startu2 =  ");
   Serial.println(start2);
    Serial.print("Trenutno stanje millisa na kraju2 =  ");
   Serial.println(kraj2);
}
```



Slika 10 „Button&millis()“ Shema spajanja

Piranja?

1. Čemu služi `millis()` funkcija
2. Kako pokrećemo i završavamo `millis()` funkciju
3. Kako ograničavamo rad `millis()` funkcije.
4. Koliko `millis()` funkcija pokrećemo u programu „Button&millis()“

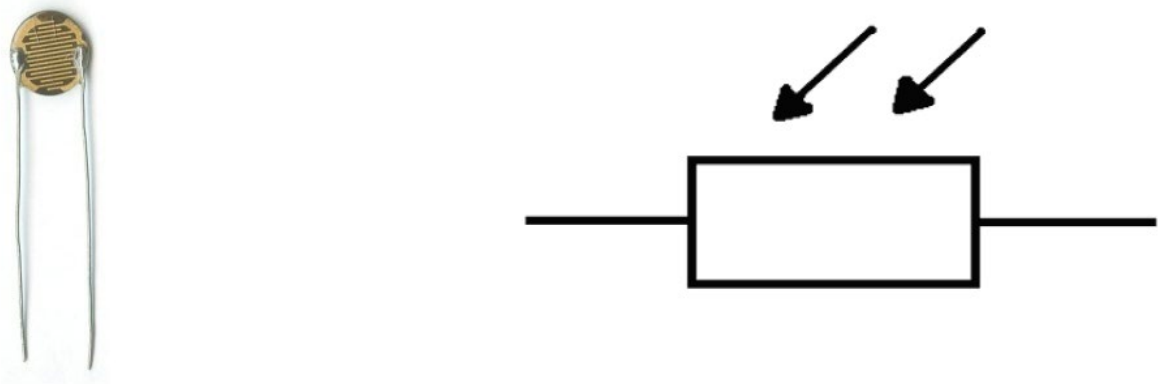
Odgovori:

1. Millis funkcija služi mjerenju proteka vremena pri izvršavanju programa a pri tome ne zaustavlja program.
2. Millis funkciju pokrećemo tako da varijablama start i kraj pridružimo millis funkciju. Tako svaka od tih varijabli dobije vrijednost millis funkcije koja je trenutna u tom trenutku. Millis funkcija počinje sa odbrojavanjem kad se prvi put pokrene program i odbrojava 49 dana neprekidno .
3. Nakon što dobijemo millis vrijednosti za start i kraj oduzmemo te vrijednosti i stavimo uvjet da razlika ne smije biti veća od željenog intervala vremena.
4. Unutar Button&millis programa pokrećemo dvije funkcije millis. Za svaku tipku posebno računamo protek vremena

3.3 VJEŽBA br. 3 „Svjetlosni prekidač i senzor udaljenosti“

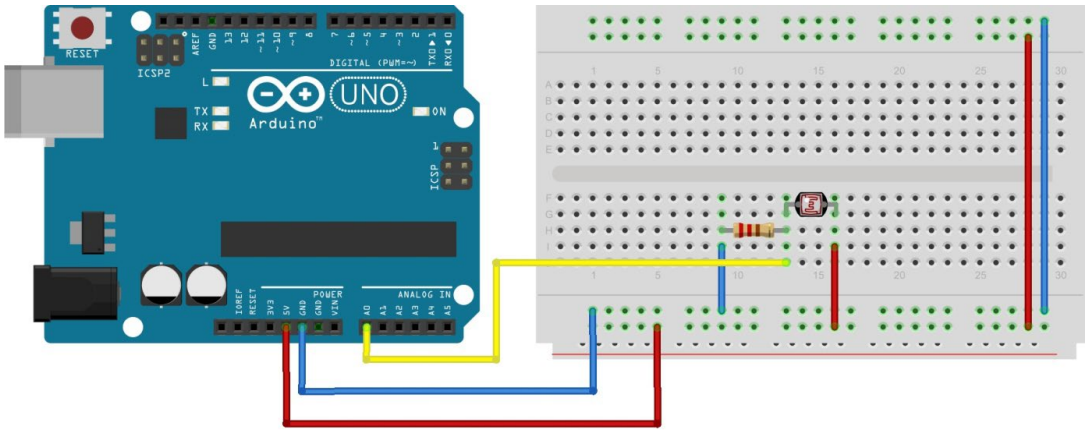
Svjetlosni prekidač (Fotootpornik)

Fotootpornik (eng. *photoresistor* ili *light dependent resistor* - *LDR*) je otpornik, čiji se električni otpor smanjuje s povećanjem intenziteta ulazne svjetlosti.



Slika 11 LDR Fotootpornik, izgled i znak

Fotootpornik se izrađuje od poluvodiča sa velikim električnim otporom. Ako svjetlo padne na fotootpornik, sa dovoljno velikom frekvencijom (granična frekvencija), poluvodič će upiti fotone svjetlosti i izbaciti elektrone, koji stvaraju električnu struju, u zatvorenom strujnom krugu.
Na Arduino pločicu spajamo LDR senzor na način da na jedan kraj senzora dovedemo napajanje a na drugi kraj stavimo otpornik 10kΩ koji je spojen sa svojim drugim krajem na masu. Tada sa točke gdje se spajaju otpornik i LDR napravimo spoj na analogni pin na Arduinu što je prikazano na slijedećoj slici



Slika 12 „LDR“ Način spajanja

Otpornik od 10 kΩ i otpor LDR-a su dva serijski spojena otpora koji se zbrajaju i tvore zajednički otpor. Što znači da promjenjivi otpor LDR-a smanjuje ukupni zbrojeni otpor od 10kΩ otpornika i preko 600kΩ LDR-a, ovisno o svjetlu koje obasjava LDR na minimalno 10kΩ. Istovremeno padom otpora rastu struja i napon u krugu. U slijedećoj tablici su prikazana kretanja otpora, struje i napona u ovisnosti o svjetlini i izabranom otporu.

Vrsta svijetla (opisno)	Količina u luxima	Otpor LDR-a	Zajednički otpor	Struja kroz zajednički otpor	Napon kroz otpor od 10kΩ
Mračan hodnik	**0.1 lux**	600KΩ	610 KΩ	0.008 mA	0.1 V
Noć sa mjesečinom	**1 lux**	70 KΩ	80 KΩ	0.07 mA	0.6 V
Tamna soba	**10 lux**	10 KΩ	20 KΩ	0.25 mA	2.5 V
Mračan oblačan dan	**100 lux**	1.5 KΩ	11.5 KΩ	0.43 mA	4.3 V
Oblačan dan	**1000 lux**	300 Ω	10.03 KΩ	0.5 mA	5V

Zadatak 1

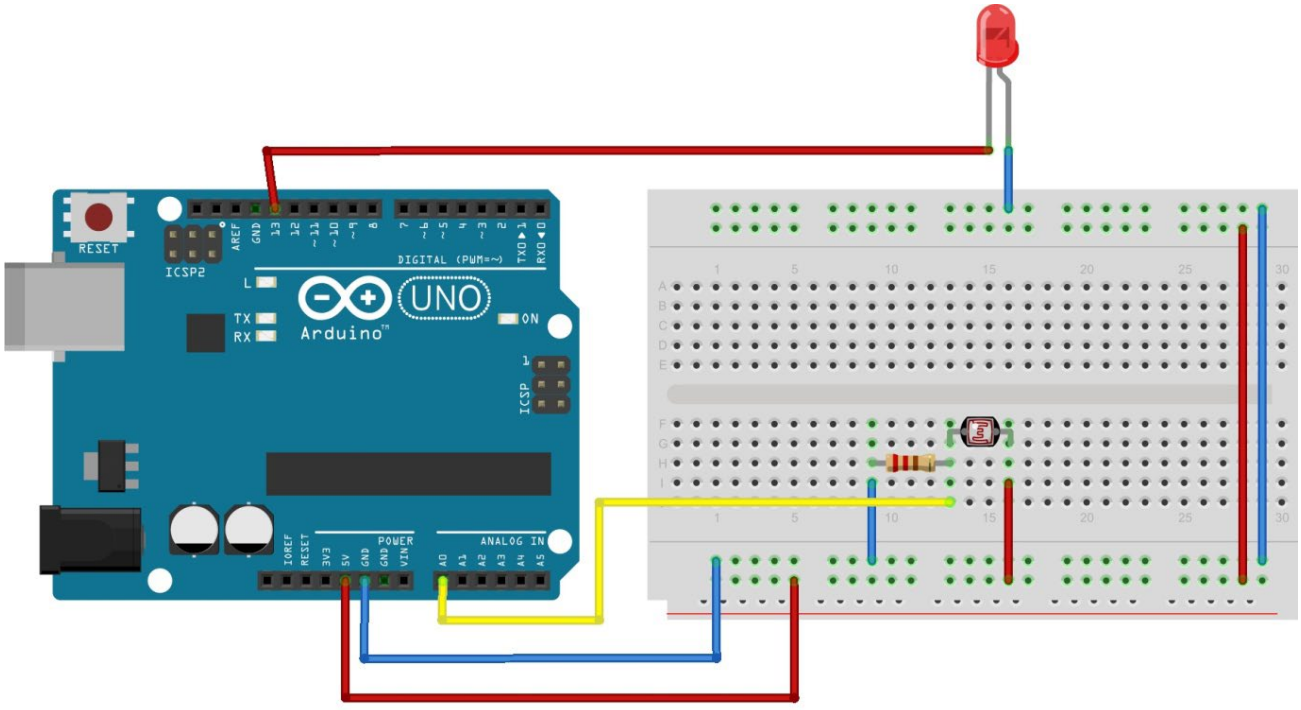
Zadani program pali LED diodu u ovisnosti o postignutom uvjetu svjetline koja pada na fotootpornik LDR.

Spojiti Arduino pločicu s računalom te učitati „Svjetlosni prekidač“ program u mikrokontroler. Zatim spojiti LED diodu, LDR i napajanje Arduina kako je prikazano na slici te potom odgovoriti na pitanja.

Kod

```
/*
SVJETLOSNI PREKIDAC
Ovaj program pali i gasi LED diodu u ovisnosti od postignutog
uvjeta  svjetline koja pada na foto-otpornik LDR.
Krug:
 * LED dioda spojena sa pina 13 na masu
 * masa spojena na otpornik od 10kΩ koji je spojen na LDR koji je spojen na
napon od +5V
 * spoj otpornika od 10kΩ i LDR-a spojen na analogni pin A0
Kreirao 2014
Joško Smolčić
*/

int LDR_Pin = A0; //analogni pin 0
int izlaz_LED = 13;

void setup(){
```

```
  Serial.begin(9600); //pokretanje serijske komunikacije
  pinMode(izlaz_LED, OUTPUT); //Definiranje varijable    izlaz_LED kao
izlaza
}

void loop()
{
  int LDRstanje = analogRead(LDR_Pin); // postavljanje te dodjeljivanje
varijabli LDRstanje vrijednosti sa LDR_Pin analognog ulaza A0

Serial.print("Trenutna osvijetljenost  =  "); //ispis na Serial Monitoru
  Serial.println(LDRstanje); // ispis stanja
  delay(10); // kašnjenje radi stabilnosti programa

if(LDRstanje >= 900) // uvjet paljenja LED diode
  {
    digitalWrite(izlaz_LED, HIGH);
  }
    else
    {
    digitalWrite(izlaz_LED, LOW);
  }
}
```



Slika 13 „Svjetlosni prekidač" Shema spajanja

Pitanja?

1. Na koji način LDR fotootpornik spajamo u strujni krug.
2. Što se mjeri na A0 analognom pinu.
3. Na koji način možemo promijeniti osjetljivost svjetlosnog prekidača.
4. Što je prikazano na Serial Monitoru

Odgovori:

_____

_____

_____

1. LDR spajamo serijski sa otporom od 10kΩ.
2. Na analognom pinu mjerimo napon koji prolazi kroz zajednički otpor od LDR-a i 10kΩ otpora. To je napon od 5V.
3. Osjetljivost svjetlosnog prekidača možemo promjeniti korištenjem drugačijeg otpornika koji se spaja u seriju s LDR-om ili programski izmjenom uvjeta koji se nalaze u „ if“ petlji programa.
4. Na Serial Monitoru prikazane su digitalne vrijednosti od 0 do 1023 koje odgovaraju vrijednostima napona od 0 do 5V.

Senzor udaljenosti

Za mjerenje udaljenosti koristiti ćemo senzor HC-SR04.

Slika 14 „HC-SR04 modul“

Ovaj senzor omogućava bez kontaktno mjerenje udaljenosti u opsegu od 2cm do 400cm uz preciznost mjerenja od oko 3mm . Najbolje rezultate senzor ima u kutu mjerenja do 30 stupnjeva. Senzor ima potrošnju struje manju od 2 mA. Modul se sastoji od ultrazvučnog predajnika, ultrazvučnog prijemnika i kontrolne elektronike te radi na sljedeći način:

1. modul se aktivira slanjem kontrolnog impulsa dužine najmanje 10µS
2. modul zatim automatski šalje osam ultrazvučnih impulsa frekvencije 40kHz
3. kada detektira povratne ultrazvučne impulse, generira izlazni signal čija je dužina proporcionalna udaljenosti.

Spajanje modula :

- pin 1: VCC - napajanje modula, 5V
- pin 2: Trig - okidanje/aktiviranje mjerenja
- pin 3: Echo - povratni signal, dužina impulsa proporcionalna udaljenosti
- pin 4: GND

Udaljenost se mjeri na osnovu dužine trajanja povratnog impulsa te se dobija po formuli:

Udaljenost = (Trajanje povratnog impulsa u sekundama) * (brzina zvuka 340 m/s) / 2

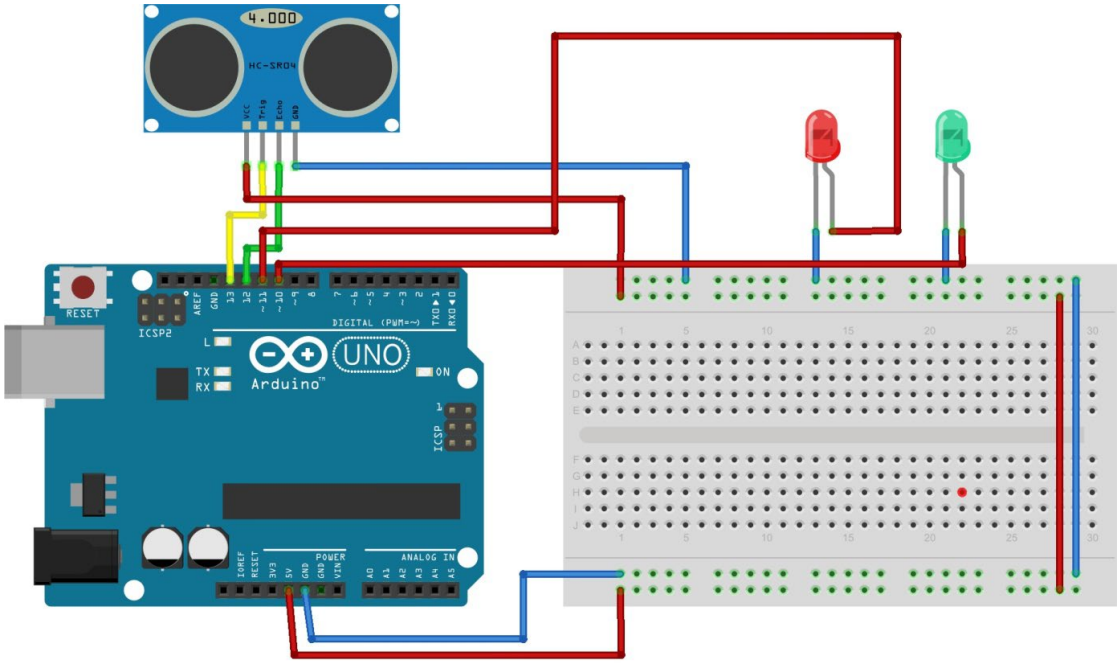
Zadatak 2

Spojiti Arduino pločicu s računalom te učitati program „Senzor udaljensti“ u mikrokontroler. Zatim spojiti shemu i napajanje Arduina kako je prikazano na slici te potom odgovoriti na pitanja.

Kod

```
/*
SENZOR UDALJENOSTI HC-SR04

  Pali se crvena LED dioda ako je predmet bliže od 10 centimetara,
  a ako je predmet dalje od 10 centimetara
  gasi se crvena i pali se zelena LED dioda.
  Ukoliko predmet nije u dosegu od 50 centimetara obe LED su ugašene,
  na Serial Monitoru se ispisuje "Nema predmeta u dosegu od 50 centimetara".

 Krug
 * VCC na Arduino 5v GND na Arduino GND
 * Echo na Arduino pin 13 Trig na Arduino pin 12
 * Crvena LED  na Arduino pin 11
 * Zelena LED to Arduino pin 10

 Kreirao 2014
 Joško Smolčić
 */

#define trigPin 13  //definiranje pinova
#define echoPin 12
#define led 11
#define led2 10

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT); //definiranje izlaza i ulaza na pinovima
  pinMode(echoPin, INPUT);
  pinMode(led, OUTPUT);
  pinMode(led2, OUTPUT);
}
void loop() {
  long trajnost;    //stvaranje varijabli long vrste
  long udaljenost;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, L OW);
  trajnost = pulseIn(echoPin, HIGH);
  udaljenost = (trajnost/2) / 29.1;//izračunavanje centimetara
  if (udaljenost < 10) {  // Uvjet za paljenje LED dioda
    digitalWrite(led,HIGH); // kad se ispune uvjeti crvena LED se pali a
zelena LED se gasi.
  digitalWrite(led2,LOW);
}
```

```
  else {
    digitalWrite(led,LOW);
    digitalWrite(led2,HIGH);
  }
  if (udaljenost >= 50 || udaljenost <= 0){ //uvjet za gašenje LED ako nisu
u dosegu
    Serial.println("Nema predmeta u dosegu od 50 centimetara");
    digitalWrite(led,LOW);
    digitalWrite(led2,LOW);
  }
  else {
    Serial.print(udaljenost);
    Serial.println(" cm");
  }

  delay(500);
}
```



Slika 15 „Senzor udaljenosti" shema spajanja

Pitanja?

1. Na koji način funkcionira senzor udaljenosti.
2. Zbog čega se koristi kašnjenje u mikro sekundama u programu, `delayMicroseconds(10);`.
3. Čemu služi funkcija `pulseIn(echoPin, HIGH);`
4. Gdje imamo dvostruki uvjet u programu.

Odgovori:

1. Modul se sastoji od ultrazvučnog predajnika, ultrazvučnog prijemnika i kontrolne elektronike te mjeri udaljenost na osnovu dužine trajanja povratnog impulsa koja se dobija po formuli: Udaljenost = (Trajanje povratnog impulsa u sekundama) * (brzina zvuka 340 m/s) / 2
2. Modul se aktivira slanjem kontrolnog impulsa dužine najmanje 10µS
3. Funkcija `pulseIn` služi čitanju vrijednosti na pinovima (dali je HIGH ili LOW). U slučaju da je vrijednost definirana kao HIGH pulseIn() čeka da stanje na pinu pređe u HIGH i tada počinje mjerenje koje završi kad se stanje na pinu promjeni u LOW. Funkcija vraća dužinu pulsa u mikro sekundama.
4. Dvostruki uvjet je postavljen u drugoj „if“ petlji iz razloga definiranja svijetljenja dioda kada su izvan opsega

4. SOFTVERSKE VJEŽBE

4.1 VJEŽBA br. 4 „Ispis teksta u Serial Monitoru“

Zadatak 1.

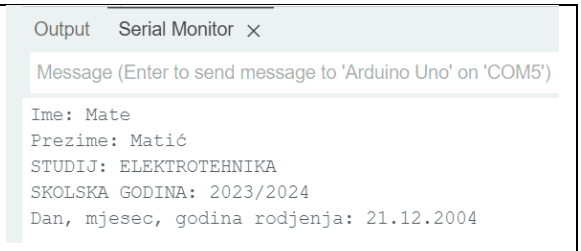
Koristeći Arduino IDE napiši program koji u Serial Monitoru ispisuje na zaslon slijedeći tekst:

Ime: Mate
Prezime: Matić
STUDIJ: ELEKTROTEHNIKA
SKOLSKA GODINA: 2023/2024
Dan, mjesec, godina rodjenja: 21.12.2004
Broj polozenih ispita: 1 od 5

Tekst se treba ispisati samo jednom u Serial Monitoru. Za unos i ispis podataka u Arduino IDE-u potrebno je koristiti Serial Monitor. Serial Monitoru možete pristupiti klikom na ikonu u toolbar traci Arduino programa  ili kombinacijom tipki Ctrl+Shift+M. Njegova svrha je da djeluje kao komunikacijski kanal između vašeg računala i Arduina. Možete utipkati poruku u polju za unos teksta na vrhu Serial Monitora i kada pritisnete Send poruka će biti poslana Arduinu. Također ako Arduino ima kakvu poruku ona će biti prikazana na Serial Monitoru

Rješenje:

```
void setup() {
  Serial.begin(9600);
 Serial.println ("Ime: Mate");
 Serial.println ("Prezime: Matić");
 Serial.println ("STUDIJ:
ELEKTROTEHNIKA");
 Serial.println ("SKOLSKA GODINA:
2023/2024");
 Serial.println ("Dan, mjesec, godina
rodjenja: 21.12.2004");
  Serial.println("Broj polozenih ispita:
1 od 5");

}

void loop() {

}
```



Zadatak 2.

Preurediti gornji kod da se ponaša na sljedeći način:

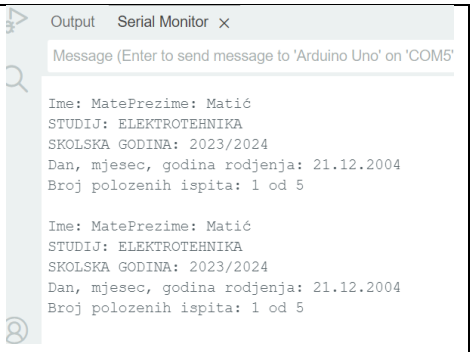
a) ime i prezime unosi zasebno ali ispisuje u istom redu.

b) ispisuje sve navedeno u razmaku od 5 sekundi.

c) između svakog ispisa postoji prazna linija (razmak).

Rješenje:

```
void setup() {
  Serial.begin(9600);

}

void loop() {
 Serial.print("Ime: Mate");
  Serial.println("Prezime: Matić");
  Serial.println("STUDIJ: ELEKTROTEHNIKA");
  Serial.println("SKOLSKA GODINA:
2023/2024");
  Serial.println("Dan, mjesec, godina
rodjenja: 21.12.2004");
  Serial.println("Broj polozenih ispita: 1 od
5");
  Serial.println("  ");

  delay(5000);

}
```



4.2 VJEŽBA br. 5 „Unos vrijednosti uz korištenje Serial Monitora“

Zadatak 1.

Napisati sljedeći program:

a) Deklarirati dvije cjelobrojne varijable a i b (varijabli „b“ pridruži vrijednost „3“ u trenutku deklaracije).

b) Koristiti Serijski Monitor za unos vrijednosti varijable a (unijeti vrijednost = 2). Za unos koristiti **if** naredbu, te **Serial.available(), i Serial.read()funkcije.**

c) Izračunati **a+b, a-b, a*b, a/b** te ispisati na Serijskom monitoru.

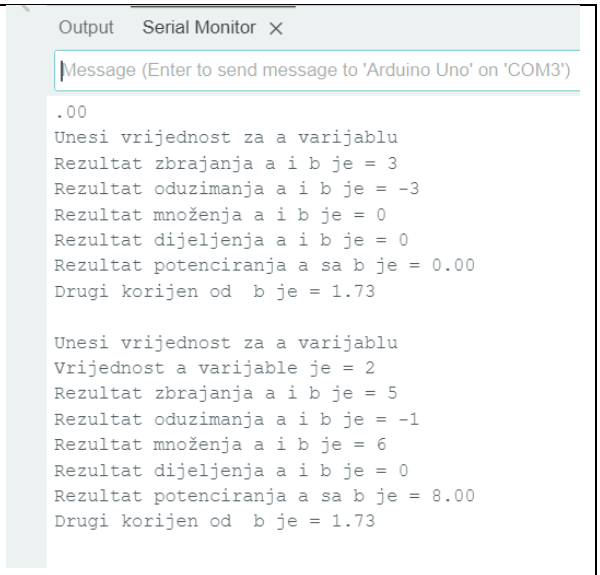
d) Izračunati potenciju a sa b koristeći **pow(x,y)** funkciju te ispisati na Serijskom monitoru.

e) Izračunati drugi korijen od b koristeći **sqrt(x)** funkciju te ispisati na Serijskom monitoru.

f) Sve ispisivati iz **void loop()** funkcije te napraviti razmak između ispisa od jednog praznog reda i vremenski od 5 sekundi.

Rješenje:

```
int a ;
int b = 3;
int c = 0;
double d = 0 ;

void setup()
{
 Serial.begin(9600);
}

void loop()
{
  Serial.println("Unesi vrijednost za
a varijablu");
  if (Serial.available() > 0) {    //
ako je znak dostupan?
   a =  Serial.read() ;       //
preuzmi znak

  a = a - 48;

    Serial.print("Vrijednost a
varijable je = ");

    Serial.println(a);
  }
```



```
c= a+b;
Serial.print("Rezultat zbrajanja a i
b je = ");
Serial.println(c);
delay(500);

c= a-b;
Serial.print("Rezultat oduzimanja a i
b je = ");
Serial.println(c);
delay(500);

c= a*b;
Serial.print("Rezultat množenja a i b
je = ");
Serial.println(c);
delay(500);

c= a/b;
Serial.print("Rezultat dijeljenja a i
b je = ");
Serial.println(c);
delay(500);

d= pow(a,b);
Serial.print("Rezultat potenciranja a
sa b je = ");
Serial.println(d);
delay(500);

d= sqrt(b);
Serial.print("Drugi korijen od  b je
= ");
Serial.println(d);
delay(10000);
Serial.println("");

}
```

4.3 VJEŽBA br. 6 „For petlja“

Osim izvršavanja različitih naredbi pod različitim okolnostima, također često ćete htjeti pokrenuti određeni niz naredbi nekoliko puta u programu. Već znate da je jedan od načina za to korištenje **loop** funkcije. Tako dugo dok naredbe u **loop** funkciji budu pokrenute one će se izvršavati automatski. Ponekad ipak trebate više kontrole. Na primjer, recimo da želimo napraviti kod tako da LED dioda treperi 20 puta te potom pauzira 3 sekunde te opet nakon toga počinje iz početka. To možete napraviti jednostavno ponavljajući isti kod opet i opet unutar vaše **loop** funkcije ovako:

```
int ledPin = 13;
int delayPeriod = 100;

void setup()
{
pinMode( ledPin, OUTPUT);
}
void loop ()
{
digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);

digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);

digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);
// ponoviti gornje četiri linije koda još 17 puta
delay (3000);
}
```

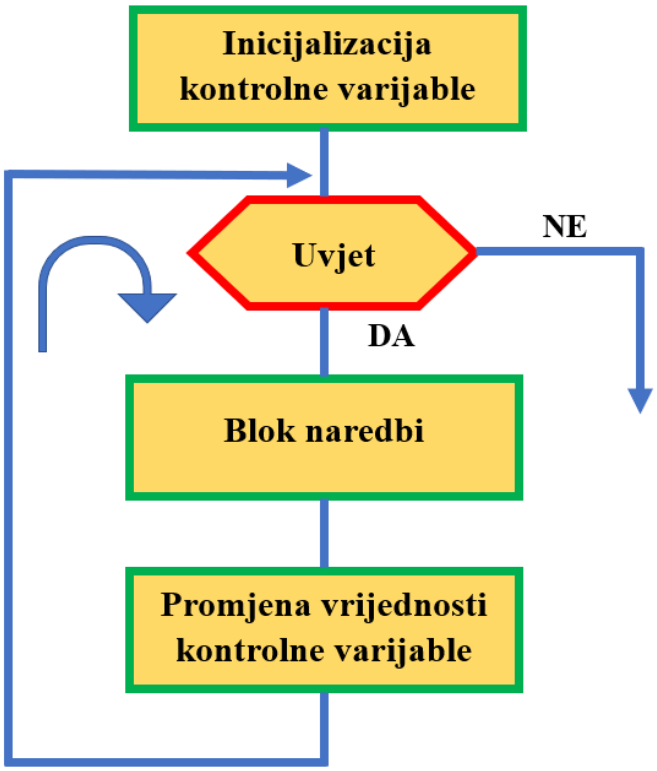
Ovakav način nije dobar za primjenu! Zašto?

Postoji nekoliko boljih načina da se napiše ovaj isti program s mnogo manje linija koda. Pogledajmo kako to možemo napraviti na drugi način uz pomoć **for** petlje.

for petlja (naredba) izgleda kao funkcija koja ima tri argumenta koja su međusobno razdvojena sa točkom i zarezom, umjesto s uobičajenim zarezom.

```
for (int i=0; i<20; i++)
```

Prva stvar nakon **for** unutar zagrada je deklarirana varijabla. Ovdje je određena varijabla koja će se koristiti kao varijabla za kontrolu izvršavanja petlje (brojač u ovom primjeru) **i** te je inicijalizirana na **0**. Drugi dio je uvjet koji mora biti istinit da bi se **for** petlja ponavljala. U ovom slučaju petlja će se izvršavati sve dok **i** varijabla ne poprimi vrijednost 20 te se tada napušta **for** petlja. Inkrement operator **i++** znači da će svaki prolaz kroz petlju uvećavati vrijednost varijable i za jedan. Kad vrijednost varijable **i** bude veća program će prestati izvršavati **for** naredbu unutar **loop** funkcije.



Slika 16 Opis rada "For" petlje

Napomena: Svako izvršavanje **for** petlje uvećava varijablu **i** za vrijednost **1**. Prednost ovakvog koda je što ga možemo lako promijeniti te staviti uvjet da varijabla **i** bude na primjer manja od 100. Nedostatak ovakvog pristupa je što izvršavanje **for** naredbe unutar **loop** funkcije okupira procesor za određeno vrijeme. To i nije problem kada naš kod kao u ovom primjeru samo pali i gasi LED diode. Međutim, često unutar **loop** funkcije treba prepoznati i neki drugi događaj osim izvršavanja petlje, primjerice poput provjere je li pritisnuta određena tipka ili je možda zaprimljena kakva serijska komunikacija. Ako je naš procesor zauzet izvršavanjem **for** petlje on neće biti u mogućnosti prepoznati takve signale u momentu dok se izvršava **for** petlja. U osnovi, pri pisanju koda potrebno je voditi računa da napisani kod unutar **loop** funkcije bude što kraći i da se učitava što brže.

Zadatak 1.

Preuredi prethodni „Blink“ program uz korištenje **for** petlje na način da imamo pet treptaja u trajanju od **500** milisekundi te nakon toga pauzu od **2000** milisekundi. Nakon toga odgovoriti na pitanje.
Napomena: Kada se kaže treptaj u trajanju od 500 ms to se misli da je LEDica svijetli 500 ms i da je nakon toga ugašena za vrijeme od 500 ms.

Rješenje:

```
int ledPin = 13;
int delayPeriod = 100;

void setup()
{
pinMode( ledPin, OUTPUT);
}
void loop ()
{
for (int i=0; i<5; i++)
{
digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);
}
delay (2000);
}
```

Pitanja:

1. Objasni funkcioniranje **for** petlje u navedenom primjeru.

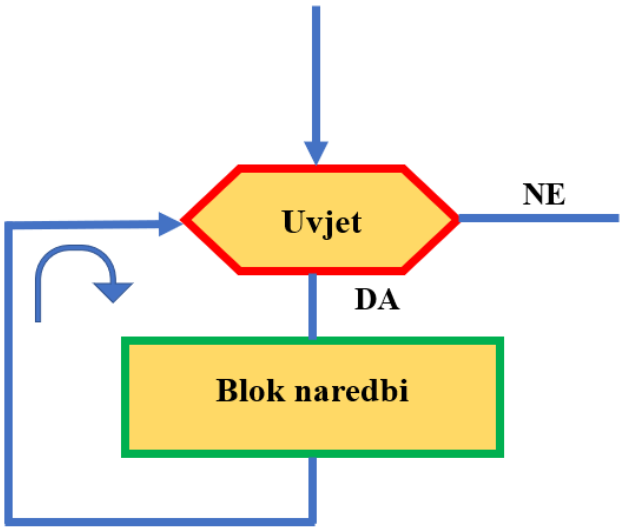
Prva što imamo nakon **for,** unutar zagrada, je deklarirana varijabla. Ovdje je određena varijabla koja će se koristiti kao varijabla za brojenje „**i“** te joj je dana početna vrijednost **0**. Drugi dio je uvjet koji mora biti istinit da bi se ostalo u **for** naredbi. U ovom slučaju naredba će se izvršavati sve dok **i** varijabla ne dobije vrijednost veću od 20. Kad vrijednost varijable **i** bude veća program će prestati izvršavati **for** naredbu unutar **loop** funkcije.

4.4 VJEŽBA br. 7 „While“ i „do while“ petlja

Drugi način kontrole toka programa u C i C++ jeziku je koristeći **while** petlju umjesto **for** petlje. Koristeći **while** petlju možete postići potpuno isti efekt kao i pri korištenju **for** naredbe. Sintaksa **while** petlje je sljedeća:

```
while (izraz) naredba;
```

Izraz u zagradama nakon **while** mora biti istinit da bi se **while** naredba nastavila izvršavati. Kada to više nije slučaj tada se prestaje izvršavati kod unuta **while** petlje.



Slika 17 Opis rada "While" petlje.

Primjenu **while** petlje možete vidjeti na slijedećem primjeru:

```
void setup()
{

}
void loop ()
{
int i=0;  // inicijalizacija varijable koja je potrebna za brojač
while (i<20)// postavljanje uvjeta
{
Xxxxxx // set naredbi koje se izvršavaju dok je uvjet ispunjen

i++;  //povećavanje kontrolne varijable
}
Yyyyyyy  // naredbe koje se izvršavaju nakon što uvjet nije ispunjen
}
```

Zadatak 1.

Napisati „Blink“ program uz korištenje **while** petlje na način da imamo pet treptaja u trajanju od **500** milisekundi te nakon toga pauzu od **2000** milisekundi.

Rješenje:

```
int ledPin = 13;
int delayPeriod = 100;

void setup()
{
pinMode( ledPin, OUTPUT);

}
void loop ()
{
int i=0;
while (i<5)
{
digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);
i++;
}

delay (2000);
}
```

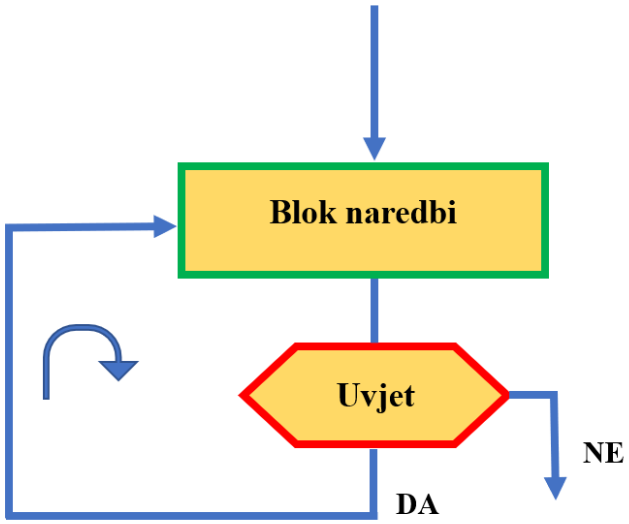
Pitanja:

1. Objasni funkcioniranje **while** petlje u navedenom primjeru.
   Kod while naredbe izraz u zagradama nakon **while** mora biti istinit da bi se **while** naredba nastavila izvršavati. Kad to više ne bude istina, tada se prestaju učitavati linije između dvije vitičaste zagrade **while** naredbe.

Do while petlja.

Petlja do-while je slična while petlji, ali se relacijski test sada nalazi na kraju petlje. Ovo osigurava izvođenje petlje barem jednom. Sintaksa je sljedeća:

```
do{
   Xxxxxx // set naredbi koje se izvršavaju dok je uvjet ispunjen
} while (uvjet);
```



Slika 18 Opis rada "Do-while" petlje

Petlja do-while se obično upotrebljava kada je uvjet koji određuje kraj izvršavanja naredbi unutar petlje definiran unutar algoritma u samoj petlji

Zadatak 2.

Napisati program koji ispisuje brojke od 1 do 5 na Serial monitoru koristeći do-while petlju. Brojke se moraju ispisati samo jednom.

Rješenje:

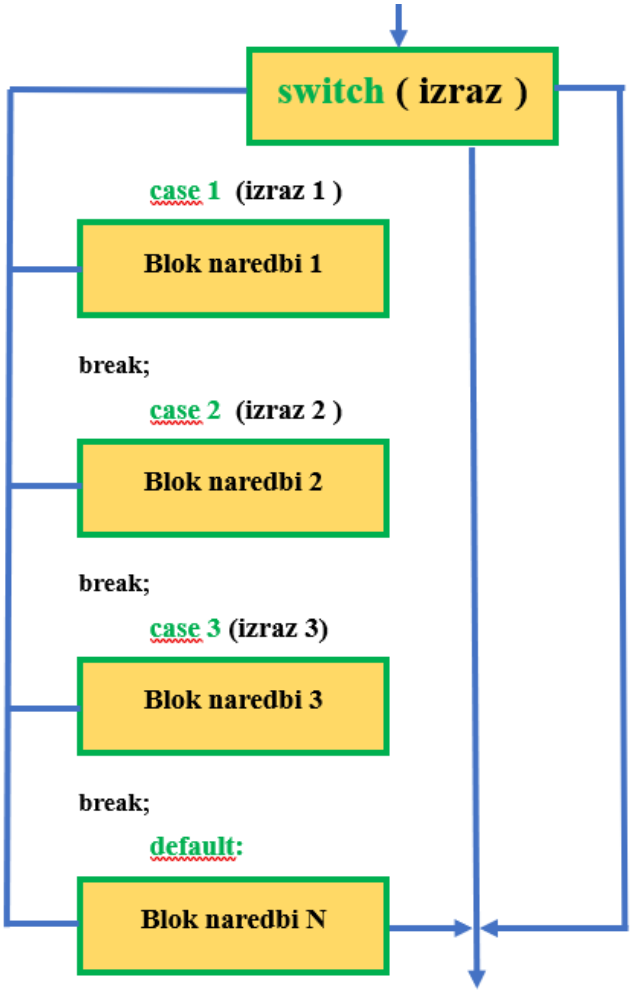
```
void setup() {
   // put your setup code here, to run once:
   Serial.begin(9600);

   //Serial.println();

   int i = 1;
   do{

      Serial.println(i);
     delay(500);

      i++;
   } while(i < 6);
}

void loop()
 {
}
```

4.5 VJEŽBA br. 8 „Switch petlja“

Ova naredba je najpogodnija za realiziranje višeznačne odluke na temelju konstantnih cjelobrojnih znakova ili znakovnih vrijednosti. Sintaksa **switch** naredbe je sljedeća:

```
switch (izraz){
case konst.izraz 1:
blok naredbi 1;
break;
case konst.izraz 2:
blok naredbi 2;
break;
default:
blok naredbi;
}
```

Ako vrijednost izraza odgovara jednom od konstantnih izraza, iza **case** ključne riječi, tada se vrši grananje na pripadajući blok naredbi. Ovaj blok naredbi završava naredbom **break** za trenutni izlaz iz **switch** naredbe. Posljednji element naredbe je default i predstavlja slučaj koji se izvodi ako nijedan od prethodnih slučajeva nije ispunjen (on se može izostaviti).



Slika 19 Opis rada "Switch" petlje.

Zadatak 1.

Primjenom „switch“ petlje napiši program koji se ponaša na sljedeći način:

1. Ako korisnik unese 1 na tipkovnici neka se upali LEDica i ispiše na Serial monitor: LED is ON
2. Ako korisnik unese 2 na tipkovnici neka se ugasi LEDica i ispiše na Serial monitor: LED is OFF
3. Ako korisnik unese 3 na tipkovnici neka se upali LEDica i ispiše na Serial monitor:
   ------- MENU -------
   1. Switch LED on.
   2. Switch LED off.
   3. This menu.
   ----
4. U slučaju unosa bilo kojih vrijednosti neka se ispiše na Serial monitor: "Invalid option"

Rješenje:

```
int led=13;
void setup()
{
  pinMode(led,OUTPUT);
  Serial.begin(9600);
}

void loop()
{ int unos;
  if(Serial.available()>0)
    {
        unos=Serial.read();

         switch(unos)
        {
         case'1':
             digitalWrite(led,HIGH);
             Serial.println("LED is ON");

         break;

         case'2':
            digitalWrite(led,LOW);
            Serial.println("LED is OFF");

         break;

         case'3':
```

```
            digitalWrite(led,HIGH);
            Serial.println("------- MENU -------");
            Serial.println("1. Switch LED on.");
            Serial.println("2. Switch LED off.");
            Serial.println("3. This menu.");
            Serial.println("--------------------");

          break;

          default:
          Serial.println("Invalid option");
          break;
      }
   }
}
```

4.6 VJEŽBA br. 9 „Funkcije“

Složeniji programski zadaci zahtijevaju stotine i tisuće linija izvornoga koda. Vrlo se često isti dijelovi koda, koji čine jednu logičku cjelinu, ponavljaju na više mjesta u programu. Ove dijelove koda dobro je izdvojiti, dodijeliti im ime i zatim ih odgovarajućim pozivima uključivati na potrebna mjesta u programu. Ovaj način rješavanja složenijih programskih zadataka vrlo je dobro podržan u C++ programskom jeziku. Jedinstvene logičke cjeline izdvajaju se u ***funkcije***.

Najvažnije prednosti uporabe funkcija su sljedeće:
1. kompaktnost programa,
2. bolja preglednost izvornog koda,
3. olakšano pisanje programa (ne ponavljaju se isti dijelovi koda) i
4. jednostavnije pronalaženje i otklanjanje grešaka.

Funkcija se može zamisliti kao ***crna kutija*** (*black box*) s jednim ili više ulaza i jednim ili više izlaza. Za ostale dijelove programa je samo bitno što funkcija zahtijeva na ulazu, način djelovanja funkcije i izlazni rezultat.

Funkcije u C++ programskom jeziku komuniciraju preko ***argumenata*** funkcije (ulazni podaci u funkciju) i preko vrijednosti koju pozvana funkcija vraća u nadređenu funkciju (izlazni podatak koji je rezultat izvršavanja funkcije). Za razliku od drugih programskih jezika (Pascal, Basic i sl.), gdje funkcije vraćaju jednu vrijednost, a procedure niz vrijednosti, funkcije u C++ mogu vratiti jednu ili nijednu vrijednost (što je određeno tipom funkcije) ili pak niz vrijednosti preko argumenata funkcije. Definicija druge funkcije unutar jedne funkcije nije dozvoljena.

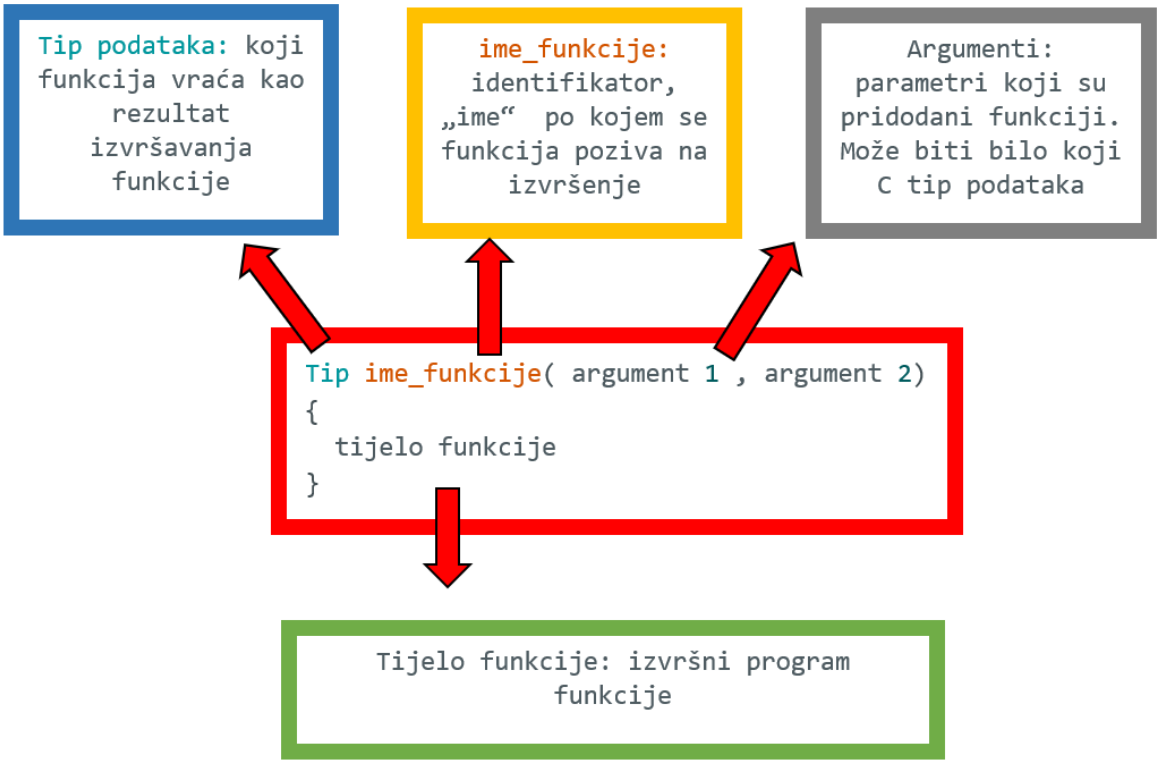
Za rad s funkcijama bitno je shvatiti sljedeće:
1. definiciju i deklaraciju funkcije,
2. način pozivanja funkcije,
3. prijenos podataka u funkciju i
4. prototip funkcije.

Funkcija predstavlja blok naredbi koji se izvršavaju kada se funkcija pozove iz nekog dijela programa.
Njen format je sljedeći:

```
tip ime (argument1, argument2, …) naredba
```

gdje je:

1. **tip (*type*)** odreduje tip podataka koji pozvana funkcija vraća u nadređenu funkciju (funkciju iz koje je pozvana)
2. **ime (*name*)** je identifikator preko kojega se funkcija poziva
3. **argumenti (*arguments*)** predstavljaju podatke koji se iz nadređene funkcije prenose u pozvanu funkciju. Argumenata može biti onoliko koliko želimo, a svaki se sastoji od tipa podataka i identifikatora. Različiti argumenti se razdvajaju zarezom.
4. **naredba (*statement*)** je tijelo funkcije. Tijelo funkcije se može sastojati od jedne naredbe ili bloka naredbi. U ovom drugom slučaju obavezno se moraju upotrijebiti {} za početak i kraj bloka.



Slika 20 Opis rada funckije.

Dok je u C++ programu obavezna samo jedna funkcija – **main()**,u Arduino inačici C jezika imamo dvije funkcije a to su: `void setup()` i `void loop()`. Funkcija `void setup()` se učitava samo jednom, dok se funkcija `void loop()`, učitava kontinuirano te je ona glavna funkcija programa.

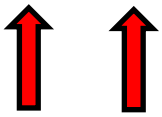
Pozivom ove funkcije označava se početak izvršavanja programa (ona komunicira s operacijskim sustavom).

Prijenos podataka pri pozivu funkcija možemo podijeliti u dvije skupine:

1. prijenos po **vrijednosti** ili sadržaju (*by value*) i
2. prijenos po **referenci** ili adresi (*by reference*).

Prilikom prijenosa po **vrijednosti** često se deklariraju varijable unutar funkcije. Varijable koje se deklariraju unutar funkcije vidljive su samo u toj funkciji (*scope of variables*). To znači da one egzistiraju ("žive") samo unutar svoje funkcije. Ove varijable nazivaju se *lokalne varijable*.

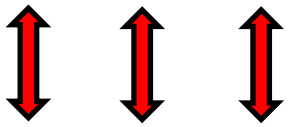
```
int zbroj( int a, int b);
```



```
   z = zbroj(  5   ,  3);
```

Pri prijenosu podataka po ***referenci* ili *adresi*** u funkciju se prenosi adresa varijable odnosno stvarna vrijednost, te ako se sada izvrši neka promjena na toj varijabli, ona će imati tu vrijednost i izvan te funkcije. Kod prijenosa po **referenci** svaki argument funkcije ima ispred identifikatora "*ampersand*" & znak, koji označava da imamo prijenos argumenata po referenci umjesto po vrijednosti. Svaka promjena varijable unutar funkcije rezultira i promjenom te iste varijable van funkcije:

```
void udvostruci( int& a, int& b, int& c );
```

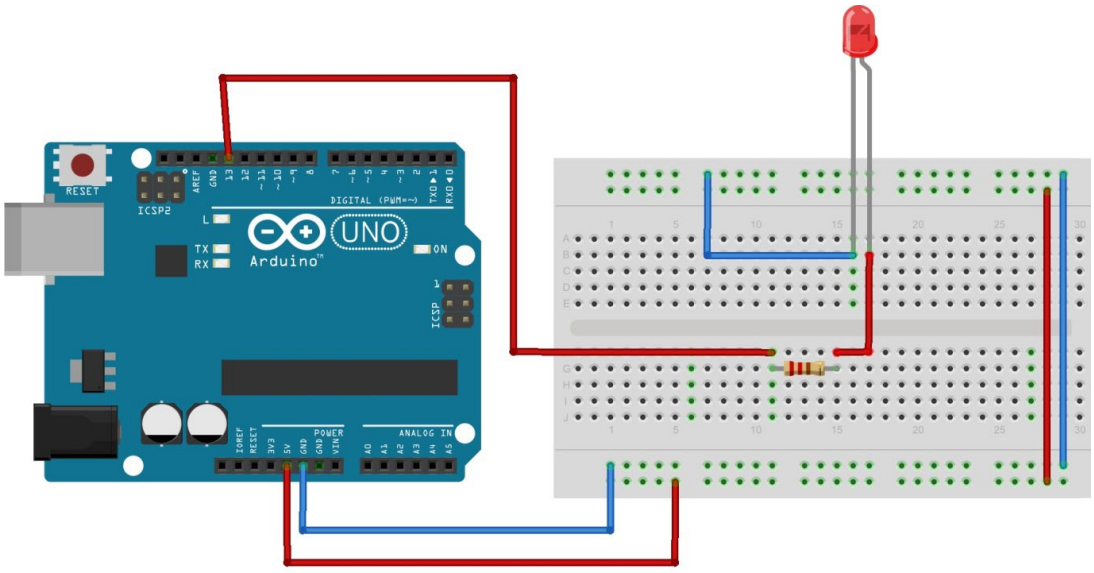


```
    udvostruci(      x,      y,      z );
```

Ovaj način prijenosa podataka u funkciju po referenci primjenom "*ampersand*" znaka moguć je samo u C++. U C jeziku za realizaciju ovoga morali bi upotrijebiti pokazivače. Prijenos podataka po referenci je vrlo efikasan način kada nam funkcija treba vratiti više vrijednosti.

Zadatak 1.

Umjesto navedene sheme u programu će se koristi ugrađena LEDica.



Slika 21 **„Blink“ shema spajanja**

„Blink“ program je primjer programa koji je pogodan za pisanje funkcija. Stoga prepravite sljedeći primjer „Blink“ programa tako da se u njega ugradi funkcija koja će se nazvati **flash** i koja nema argumenata. Funkcija **flash** će nakon svog pozivanja izazvati treptanje diode na način da razmaci između svijetljenja i ugašenosti LED iznose 500 milisekundi. U tu svrhu također napravite globalnu varijablu **delayPeriod** te joj pridružite potrebne vrijednosti. Treperenje treba se ponoviti 20 puta s razmakom od 3 sekunde za što treba upotrijebiti **for** petlju.

```
int ledPin = 13;

void setup()
{
pinMode( ledPin, OUTPUT);

}
void loop ()
{
digitalWrite (ledPin, HIGH);
delay (500);
digitalWrite (ledPin, LOW);
delay(500);
}
```

Rješenje:

```
int ledPin = 13;
int delayPeriod = 250;
```

```
void setup()
{
pinMode( ledPin, OUTPUT);

}
void loop ()
{

for (int i=0; i<20; i++)
{
flash ();
}

delay (3000);
}

void flash ()
{
digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);
}
```

Pitanja:

1. Zbog čega su u funkciji **flash ()** zagrade prazne?

Zagrade u funkciji flash () su prazne iz razloga što funkcija ne koristi nikakve parametre.

2. Prilikom definicije, deklariranja i pozivanja funkcije gdje sve stavljamo točku-zarez a gdje ne?

Prilikom deklariranja funkcije pišemo tip, ime te argumente funkcije, ako ih ima, te stavljamo na kraju točku-zarez. U slučaju ako iza deklariranja funkcije stoji odmah i definicija funkcije (tijelo funkcije) tada ne stavljam točku-zarez iza tipa, imena i argumenata. Prilikom pozivanja funkcije ne pišemo tip funkcije samo ime i argumente ako ih ima te stavljamo točku-zarez.

Zadatak 2.

Koristeći rješenje iz prethodnog primjera napravite funkciju u kojoj će te u pozivu funkcije definirati dvije stvari:

1. koliko puta da LED dioda zatreperi
2. koliko da budu dugački ili kratki ti treptaji.

U tu svrhu napravite dvije lokalne varijable (**numFlashes** i **d**) koje će biti korištene samo unutar **flash** funkcije. Prijenosom vrijednosti u funkciju varijabli **d** pridružite razmak između treptaja (**delayPeriod**) koji može ostati 500 milisekundi, a varijabli **numFlashes** broj treptaja. Na taj način možete jednostavno mijenjati broj treptaja u funkciji promjenom vrijednosti prilikom pozivanja funkcije. Nadopuni funkciju tako da se svaki put na Serial monitor ispiše kada je dioda upaljena:

"LEDica je upaljena!"

Te kada je ugašena:

"LEDica je ugašena!"

Rješenje:

```
int ledPin = 13;
int delayPeriod = 250;

void setup()
{
pinMode( ledPin, OUTPUT);

}
void loop ()
{
flash (20, delayPeriod);
delay (3000);
}

void flash (int numFlashes, int d)
{
for (int i=0; i< numFlashes; i++)
{
digitalWrite (ledPin, HIGH);
delay (d);
digitalWrite (ledPin, LOW);
delay (d);
}
}
```

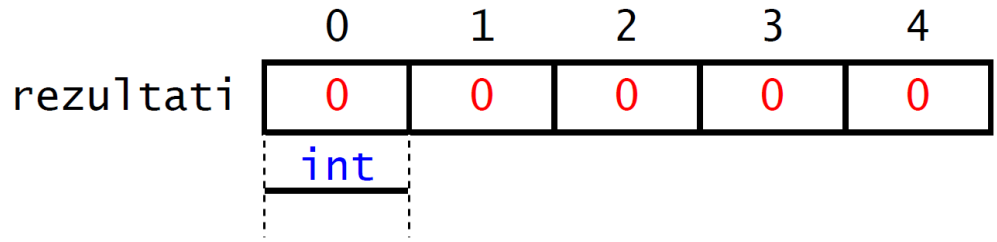
4.7 VJEŽBA br. 10 „Polja podataka“

Često u programu trebamo koristiti skupove istovrsnih podataka. Za označavanje i rukovanje ovakvim podacima bilo bi vrlo nespretno upotrijebiti zasebne varijable x_1, x_2, x_3, ….
Da bi ovo shvatili pretpostavimo obradu rezultate nekoga mjerenja koje sadrži 100 podataka, pri čemu svaki podatak treba proći isti algoritam (niz računskih operacija). Ako bi deklarirali i upotrijebili 100 različitih varijabli, tada bi se program sastojao od 100 istih blokova naredbi koje bi se razlikovale samo u imenu varijable koja se u dotičnome bloku naredbi obraduje. Očito je da je ovakav način pisanja programa neučinkovit. Puno bolje rješenje je sve podatke smjestiti pod isto ime, a pojedini rezultat dohvaćati pomoću brojčanoga indeksa. Ovakav način organizacije podataka omogućuju nizovi, **polja** (*arrays*) podataka.

Polje podataka je niz konačnoga broja istovrsnih tipova podataka, koji predstavljaju č*lanove* (*elemente*) *polja*. Ti podaci mogu biti bilo kojega tipa:

- ugrađenoga (**int, float, double, char** i sl.) ili
- korisnički definiranoga.

Pojedinim članovima polja može se pristupiti pomoću cjelobrojnoga indeksa te ih mijenjati neovisno o ostalim članovima polja. Primjerice polje od 5 cjelobrojnih vrijednosti imenom **rezultati** može se prikazati na sljedeći način:



gdje svaki prazan pravokutnik predstavlja jedan element polja, koji su u našem primjeru vrsta cjelobrojnih podataka. Elementi polja su numerirani od 0 do 4, što znaci da prvi element polja uvijek ima indeks 0 neovisno o dužini polja.

Tipična deklaracija polja u C++ je:

```
tip ime[broj_elemenata];
```

gdje je **tip** valjani tip podataka, **ime** je valjani identifikator, a unutar zagrada dolazi broj elemenata polja. To znaci da bi za naš prethodni primjer ispravna deklaracija bila:

```
int rezultati[5];
```

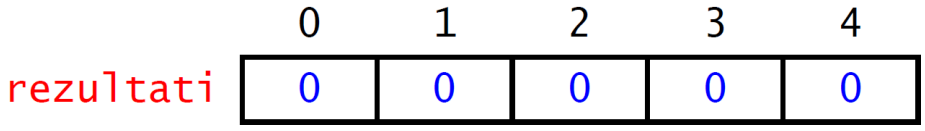
Napomena:
Broj elemenata polja unutar pravokutnih zagrada *mora biti konstantna vrijednost*, budući da su polja blokovi statičke memorije određene veličine, te stoga prevoditelj mora točno odrediti koliko memorije treba pridružiti danome polju prije nego što će se bilo koja naredba izvrši.

Inicijalizacija polja

Kada *deklariramo polje* unutar neke *funkcije* (lokalna varijabla), *sadržaj elemenata polja* je neodređen u trenutku deklaracije (ako se drugačije ne specificira, kao i za sve druge lokalne varijable). Ako deklariramo globalno polje (izvan svih funkcija), vrijednost svih elemenata polja bit će 0, tako, ako je rezultat globalno polje:

```
int rezultati[5];
```

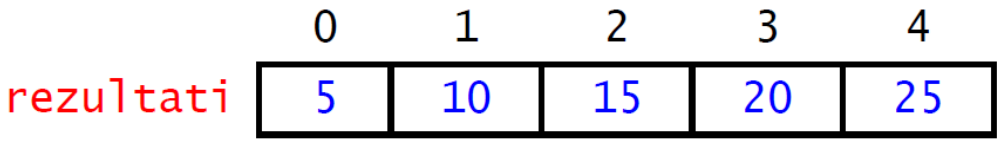
svakome elementu polja pridružit će se 0 kao početna vrijednost:



Također, kod same deklaracije imamo mogućnost pridružiti početnu vrijednost svakome elementu polja unutar vitičastih zagrada {}. Primjerice:

```
int rezultati[5] = {5, 10, 15, 20, 25};
```

što znaci da smo kreirali sljedeće polje:



Broj elemenata polja kojima smo pridružili vrijednost unutar vitičastih zagrada, mora odgovarati broju elemenata polja unutar pravokutnih zagrada, kao što se vidi iz prethodnoga primjera. Budući da ovo predstavlja ponavljanje iste informacije (*broj elemenata polja*), C++ ostavlja mogućnost izostavljanja broja elemenata polja unutar pravokutnih zagrada [], kada inicijaliziramo polje na ovaj način:

```
int rezultati[] = {5,10,15,20,25};
```

Pristupanje elementima polja

Bilo gdje u programu gdje je polje vidljivo, možemo pristupiti pojedinome elementu polja kako bi mu pročitali pridruženu vrijednost ili je promijenili na način:
ime[indeks]

Tako za naš prethodni primjer, pojedinim elementima polja može se pristupiti na sljedeći nacin:

Tako, ako želimo pridružiti vrijednost 45 drugome elementu to možemo učiniti na način:

```
rezultati[1] = 45;
```

i ovu vrijednost sada možemo pridružiti nekoj varijabli na nacin:

```
a = rezultati[1];
```

To znaci da izraz **rezultati[1]** ima potpuno isto značenje i svojstva kao i svaka druga Cjelobrojna varijabla.

Napomena:
Broj indeksa elementa polja je uvijek manji za jedan (1) od broja elementa. To znaci da kada želimo pristupiti primjerice ***trećemu*** elementu polja, to radimo na sljedeći način:

```
rezultati[2];
```

Također u C++ možemo prekoračiti deklarirani broj elemenata polja, ali to nije preporučljivo, jer može izazvati neke logičke pogreške u programu koje je teško otkriti, budući da prevoditelj neće otkriti nikakve pogreške ali program će se čudno ponašati. Razlog zašto je ovo dozvoljeno postat će jasan kada se obrade **pokazivači**. Također, vrlo je bitno uočiti razliku u primjeni pravokutnih zagrada kod polja. Jedna primjena predstavlja navođenje veličine polja kod deklaracije, dok druga mogućnost uporabe indeksa preko kojega se pristupa pojedinome elementu polja:

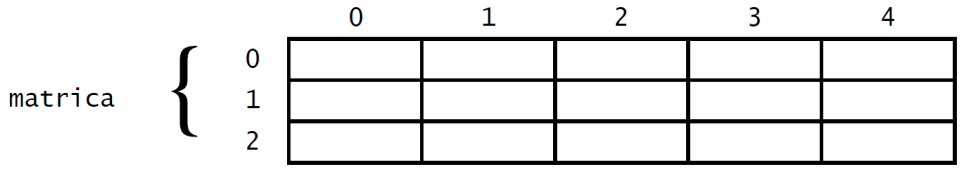
```
int rezultati[5]; // deklaracija polja
rezultati[1] = 45; // pridruživanje vrijednosti 2. elementu polja
```

Druge valjane operacije s poljima su:

```
rezultati[0] = a;
rezultati[a] = 75;
b = rezultati[a+2];
rezultati[rezultati[a]] = rezultati[2] + 5;
```

Višedimenzijska polja

Do sada smo razmatrali jedno dimenzijska polja. Međutim često nam trebaju vise dimenzijska polja, primjerice matrice. Tako se dvo-dimenzijsko polje može shvatiti kao matrica:

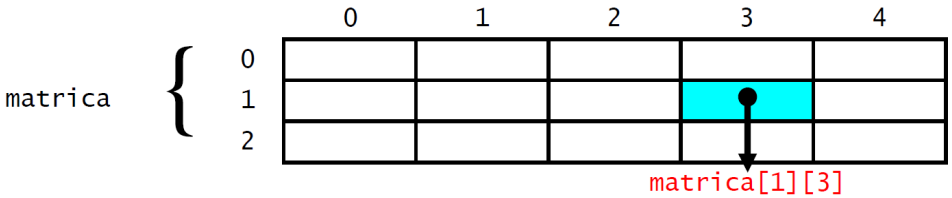


gdje ***matrica*** predstavlja dvo-dimenzijsko polje dimenzija 3 x 5 i svaki element je cjelobrojnoga tipa. Ovo polje deklarira se na sljedeći način:

```
int matrica[3][5];
```

te ako želimo pristupiti elementu u drugome retku i četvrtome stupcu to možemo učiniti na sljedeći način:

```
matrica[1][3]
```



Napomena:
Indeksi počinju od 0!. Vise dimenzijska polja nisu ograničena samo na dvije dimenzije. Ona mogu imati onoliko dimenzija koliko je potrebno, ali rijetko će se upotrebljavati polja s više od tri dimenzije. Jedan od glavnih razloga za ovo jest potrebna količina memorije, primjerice:

```
char stoljece_sekunde[100][365][24][60][60];
```

gdje pridružujemo jedan znak za svaku sekundu u jednom stoljeću, što rezultira:
100 x 365 x 24 x 60 x 60 = 3.153.600.000 znakova.

Kako jedan znak predstavlja 1 byte to znaci da je potrebno više od 3 GB RAM-a da bi mogli deklarirati ovo polje, a to je za 32-bitne sustave računala izvan raspoloživoga adresnog prostora. Više dimenzijska polja nisu ništa drugo nego jedna apstrakcija, budući da jednostavnim množenjem dimenzija dvo-dimenzijskoga polja, možemo dobiti iste rezultate primjenom jedno dimenzijskoga polja:

```
int matrica[3][5];
```

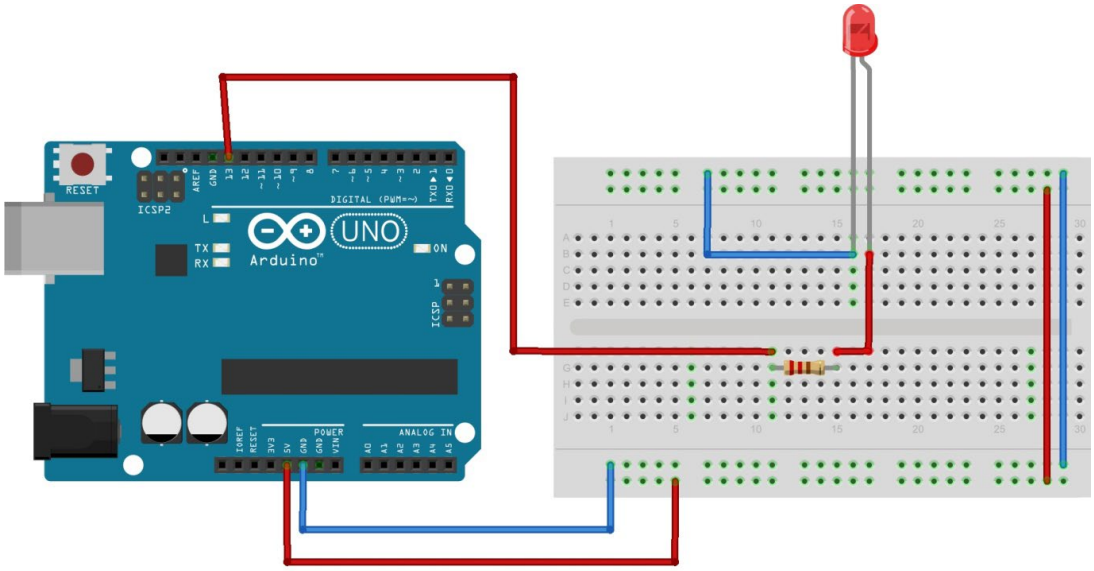
je ekvivalentno (3 * 5 = 15)

```
int matrica[15];
```

s tom razlikom što u prvome slučaju prevoditelj pamti za nas “dubinu“ svake imaginarne dimenzije, a u drugome slučaju mi moramo voditi računa o tome.

Zadatak 1.

Koristeći Arduino spojiti sljedeću shemu:



Slika 22 **„Blink“ shema spajanja**

Uz pomoć polja podataka preraditi sljedeći primjer „Blink“ programa tako da napravite S.O.S. poziv u pomoć na način da manipulirate s vremenom kašnjenja. U tu svrhu kreirajte funkciju sa sljedećim imenom te argumentom: `void flash(int delayPeriod).` Također kreirajte polje podataka naziva: `int durations[] = {},` unutar kojeg će te smjestiti podatke koje će te pozivajući ih iskoristiti za kašnjenje unutar funkcije **`flash`**. Za pozivanje podataka iz polja podataka koristite `for` petlju. Za točku koristite 200 a za crticu 500 milisekundi vrijeme kašnjenja. Nakon što S.O.S. završi treptati ostavite jednu sekundu razmaka.

```
int ledPin = 13;

void setup()
{
pinMode( ledPin, OUTPUT);

}
void loop ()
{
digitalWrite (ledPin, HIGH);
delay (500);
digitalWrite (ledPin, LOW);
delay(500);
}
```

Pitanja:

1. Koja je prednost ovog S.O.S programa ako prilikom izrade koristimo polja podataka?

Očita prednost ovog pristupa je u tome što je veoma lako mijenjati poruku jednostavno mijenjajući vrijeme trajanja unutar polja podataka.

Rješenje

```
int ledPin = 13;
int durations [] = {200,200,200, 500,500,500, 200,200,200,};

void setup()
{
pinMode( ledPin, OUTPUT);

}
void loop ()
{
for(int i =0; i<9; i++)
{
flash(durations[i]);
}
delay (1000);
}

void flash(int delayPeriod)
{
digitalWrite (ledPin, HIGH);
delay (delayPeriod);
digitalWrite (ledPin, LOW);
delay (delayPeriod);
}
```

Zadatak 2.

Unutar „Blink“ programa iz prethodnog primjera kreirati sljedeće polje podataka**:**

```
int durations [] = {0,3,6,9,12,15,18,21};
```

Polje sadrži broj treptaja naše LED diode. Program treba napisati tako da parametrima polja podataka pristupamo preko Serijskog monitora na način ako upišemo u Serijski monitor broj 2 to je u našem polju podataka treća pozicija s vrijednošću 6 te će naša LED dioda treptati 6 puta. Nakon toga možemo pristupiti sljedećem parametru koji želimo.
U tu svrhu kreiraj **flash** funkciju koja će vršiti treperenje. Razmak između treptaja treba biti 200 milisekundi. Također izvrši ispis unesene vrijednosti sa Serijskog monitora, kao i ispis dobivene količine treptaja (podatak iz polja podataka). U tu svrhu potrebno je kreirati jednu cjelobrojnu varijablu **int a;** te joj pridružiti vrijednosti sa Serijskog monitora. Za to će te koristiti funkcije **Serial.available();** i **Serial.read();**. Također potrebno je nakon što je pročitana sa

Serijskog monitora umanjiti vrijednost **int a** varijable na način **int a = a – 48;**. Nakon što se treptanje dogodi potrebno je završiti serijsku komunikaciju s funkcijom **Serial.end();** te ponovno pokrenuti serijsku komunikacju s funkcijom **Serial.begin(9600);.** Nakon svega odgovoriti na pitanja ispod.

Pitanja:

2. Zbog čega umanjujemo vrijednost varijable **a**?

Za mikrokontroler znak 2 koji smo unijeli preko Serijskog monitora ima vrijednost 50 po ASCII kod (American Standard Code for Information Interchange). S obzirom da smo našu varijablu **a** deklarirali kao **int** oduzimanjem 48 smo dobili njenu vrjednost kao broja 2.

3. Iz kojeg razloga završavamo serijsku komunikaciju te ju ponovno pokrećemo?

Serijsku komunikaciju završavamo, te je ponovno pokrećemo iz razloga da bi očistili memoriju te omogućili eventualni unos sljedećeg znaka preko Serijskog monitora.

Rješenje:

```
int ledPin = 13;
int durations [] = {0,3,6,9,12,15,18,21};
int a;

void flash()
{
digitalWrite (ledPin, HIGH);
delay (200);
digitalWrite (ledPin, LOW);
delay (200);

}

void setup()
{
pinMode( ledPin, OUTPUT);
Serial.begin(9600);
}

void loop ()
{

if(Serial.available() > 0)
 {
  a = Serial.read();
  a = a-48;
  Serial.println(" ");
```

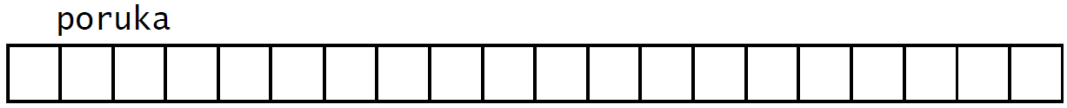
```
  Serial.print("Unesena je brojka  ");
  Serial.println(a);

for(int i =0; i<durations[a]; i++)
{
flash();

}

delay (1000);
Serial.print("LED lampica je treptala ");
Serial.print(durations[a]);
Serial.print(" puta");

Serial.end();    // Završava serijsku komunikaciju  kada je podatak primljen
     Serial.begin(9600);  // Ponovno pokreće serijsku komunikaciju i briše
sve što je ostalo u bufferu od prije.
}
}
```

4.8 VJEŽBA br. 11 „Znakovni stringovi i pretvorba znakova“

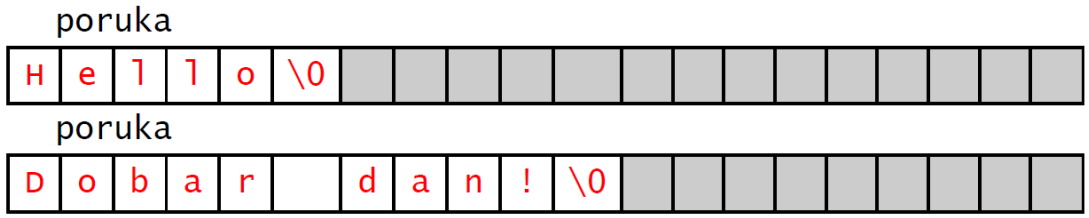
U C++ ne postoji znakovni niz (*string*) kao osnovni tip podataka. Do sada smo susreli znakovne nizove jedino kao konstante kod određenih ispisa. Međutim, u programiranju često se ukaže potreba da varijable sadrže znakovne nizove. Rješenje za ovo predstavlja ***polje znakova***:

```
char poruka[20];
```

Ovo polje znakova nazvano poruka može spremiti string dužine do 20 znakova.



Maksimalna dužina znakovnog niza ne mora se uvijek upotrijebiti. Tako primjerice u znakovno polje poruka možemo spremiti znakovne nizove “Hello“ ili “Dobar dan!“ i sl. na sljedeći način:



gdje sivi kvadratići predstavljaju neiskorištena mjesta znakovnoga niza. Kao što je iz slike vidljivo, svaki znakovni niz mora završiti nul znakom **'\0'** (*nullcharacter*), koji predstavlja kraj stringa.

Inicijalizacija znakovnoga niza

Znakovni nizovi predstavljaju polja podataka pa stoga moraju zadovoljiti sva pravila koja se odnose na ovaj izvedeni tip podataka. Stoga, ako želimo inicijalizirati unaprijed definiranim vrijednostima, to možemo učiniti kod same deklaracije znakovnoga niza na slijedeći način:

```
char poruka [] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

U ovome slučaju deklarirali smo znakovni niz dužine 6 elemenata znakovnoga tipa, inicijaliziran skupom znakova koji predstavljaju riječ **Hello**, te zaključen s **'\0'**. Osim ovoga već smo upoznali način inicijalizacije stringa uporabom **konstantnih stringova.** Ovi stringovi su zatvoreni između znakova navođenja "**…**":

```
"rezultat je: "
```

te ovi stringovi imaju nul znak (\0) automatski dodan na svome kraju. Zbog ovoga gore napisanoga, možemo inicijalizirati string poruka na jedan od ova dva načina:

```
char poruka [] = {'H', 'e', 'l', 'l', 'o', '\0'};
char poruka [] = "Hello";
```

U oba slučaju string se sastoji od 6 elemenata: 5 elemenata sačinjava riječ **Hello** dok je šesti element **'\0'** koji se automatski dodaje na kraj stringa u drugome slučaju.
Pridruživanje vrijednosti stringu na ovaj način je moguće samo u trenutku deklaracije, dok u samome programu to nije dozvoljeno na ovaj način:

```
poruka = "Hello"; // nije dozvoljeno
poruka [] = "Hello"; // nije dozvoljeno
poruka = {'H', 'e', 'l', 'l', 'o', '\0'}; // nije dozvoljeno
```

Pridruživanje vrijednosti stringu

Zbog toga što **lvalue** prilikom pridruživanja može biti *samo jedan element polja*, a ne cijelo polje, vrijednosti stringu možemo pridružiti na sljedeći način:

```
poruka[0] = 'H';
poruka[1] = 'e';
poruka[2] = 'l';
poruka[3] = 'l';
poruka[4] = 'o';
poruka[5] = '\0';
```

Kao što se može vidjeti ovo nije baš praktična metoda pogotovo za duže stringove. Za pridruživanje vrijednosti stringu postoji funkcija **strcpy_s (*stringcopy*)**koja je definirana u biblioteci **cstring(string)**i u biblioteci **iostream,** a poziva se na sljedeći način:

```
strcpy_s (string1, string2);
```

Ova funkcija jednostavno kopira sadržaj **string2**u **string1**, gdje **string2**može biti *polje*, *pokazivač* ili *konstantan string*:

```
strcpy_s (poruka, "Hello");
```

Zadatak 1

Napiši Arduino program u kojem ćeš definirati string, imena **string** te mu pridružiti: **"Ovo je moj string"**. Zatim deklariraj drugi string naziva **drugi_string[40],** te jednu cjelobrojnu varijablu **int a = 0;**. Neka program obavi sljedećih sedam operacija:

1. Ispiši string **string**

   ```
   Serial.println(string);
   ```

2. Izmjeri veličinu stringa (isključuje zadnji znak nule)

```
a = strlen(string);
```

3. Izmjeri veličinu cijelog polja stringova (uključujući znak nule)

```
 a = sizeof(string);
```

4. Kopiraj string, **string,** u drugi string, **drugi_string**

```
strcpy(drugi_string, string);
```

5. Na kraju teksta od **drugi_string** dodaj dodatak teksta **"dodatak".**

```
strcat(drugi_string, " dodatak.");
```

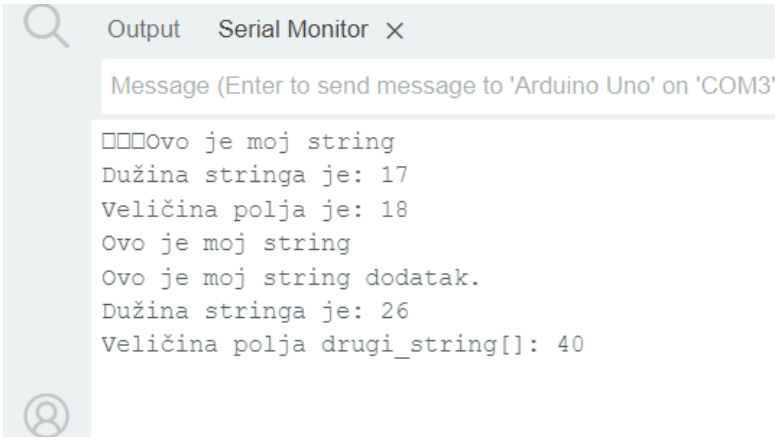
6. Sad opet izmjeri veličinu stringa **drugi_string** (isključuje zadnji znak nule)

```
a = strlen(drugi_string);
```

7. Sad opet izmjeri veličinu cijelog stringa, **drugi_string** s korištenim i nekorištenim poljima

```
a = sizeof(drugi_string);
```

Neka ispis na Serijskom monitoru izgleda ovako:



Slika 23 Ispis na Serial monitoru

Neka se sve ovo ispiše samo jednom.

Rješenje:

```
void setup() {
  char string[] = "Ovo je moj string";  // deklaracija stringa stringa
  char drugi_string[40];                  // deklaracija drugog stringa
  int a;                            // cjelobrojna varijabla a

  Serial.begin(9600);

  // (1) ispis stringa
  Serial.println(string);
```

```
  // (2) izmjeri veličinu stringa (isključuje zadnji znak nule)
  a = strlen(string);
  Serial.print("Dužina stringa je: ");
  Serial.println(a);

  // (3) izmjeri veičinu polja znakova (uključujući znak nule)
  a = sizeof(string);  // sizeof() is not a C string function
  Serial.print("Veličina polja je: ");
  Serial.println(a);

  // (4) kopiraj string
  strcpy(drugi_string, string);
  Serial.println(drugi_string);

  // (5) dodaj string na kraj stringa (dodatak)
  strcat(drugi_string, " dodatak.");
  Serial.println(drugi_string);
  //(6) izmjeri veličinu stringa drugi_string (isključuje zadnji znak nule)
  a = strlen(drugi_string);
  Serial.print("Dužina stringa je: ");
  Serial.println(a);
  //(7) izmjeri veličinu cijelog stringa, drugi_string s korištenim i
nekorištenim poljima
  a = sizeof(drugi_string);
  Serial.print("Veličina polja drugi_string[]: ");
  Serial.println(a);
}

void loop() {
}
```

4.9 VJEŽBA br. 12 „Čuvanje podataka koristeći Flash i EEPROM memoriju“

Kada varijablama date vrijednosti, Arduino će zapamtiti te vrijednosti tako dugo dok je uključen.Kada isključite vaš Arduino, tada ste izbrisali i te vrijednosti i svi podaci su izgubljeni. U ovoj vježbi ćete naučiti neke načine kako da zadržite podatke.
Ako se podaci koje želite sačuvati ne mijenjaju, tada jednostavno možete učitati podatke svakiput kada pokrenete Arduino. Ipak mnogo je bolje sačuvati ove podatke u 32K Flash memorije, nego u 2K RAM memorije. Postoji način za to napraviti. To je direktiva nazvana PROGMEM; ona je zapisana u biblioteci i malo je nezgodna za korištenje.

PROGMEM naredba

Da bi sačuvali vaše podatke u flash memoriji, morate uključiti PROGMEM biblioteku na sljedeći način:

```
#include <avr/pgmspace.h>
```

Svrha ove naredbe je da kaže kompajleru da koristi **`pgmspace`** biblioteku u ovom kodu. U ovom slučaju, biblioteka je skup funkcija koje je netko napisao i koju možete koristiti u vašem kodu bez potrebe da razumijete svaki detalj kako funkcionira.

Iz razloga što koristite ovu biblioteku, **`PROGMEM`** ključna riječ i **`pgm_read_word`** funkcije su vam dostupne. U kodu koji slijedi koristimo oboje.

Ova biblioteka je sastavni dio Arduino softvera. Dobra kolekcija takvih službenih biblioteka je dostupna u mnogim neslužbenim bibliotekama, koje su razvijene od ljudi kao što ste i vi koji su to napravili za druge i sve objavili dostupno na internetu. Takve neslužbene biblioteke moraju biti instalirane u vaše Arduino okružje.

Kada koristite PROGMEM, morate biti sigurni da koristite specijalne PROGMEM frendly podatkovne tipove. Nažalost, to ne uključuje polje znakova (char arrey). U sljedećm primjeru ako želite u memoriju pohraniti znakove Morseove abecede vi morate definirati varijable za svaki string koristeći PROGMEM string tip i onda ga staviti u PROGMEM polje na ovakav način:

```
PROGMEM prog_char sA[] =  ".-" ;
PROGMEM prog_char sB[] =  "-..." ;
// i tako dalje za sva slova
PROGMEM const char* letters []=
{
sA, sB, sC, sD, sE, sF, sG, sH, sI,sJ, sK, sL, sM,
sN, sO, sP, sQ, sR, sS, sT, sU,sV, sW, sX, sY, sZ,
};
```

Uz pisanje podataka na poseban način, također morate pročitati te podatke na poseban način. Vaš kod za dobivanje nizova znakova za Morseovu abecedu iz polja mora biti izmijenjen tako da izgleda ovako:

```
strcpy_P(buffer,(char*)pgm_read_word(&(letters[ch - 'a'])));
```

Ovdje se koristi buffer varijabla unutar koje je kopiran PROGMEM string tako da se on može koristiti kao obično polje znakova (char array). Ovdje je potrebno definirati globalnu varijablu kao u sljedećem slučaju:

```
char buffer [6];
```

Ovakav pristup djeluje samo ako su podaci konstantni, dakle ako ih ne želite mijenjati dok je kod pokrenut. U sljedećem dijelu će te naučiti više o korištenju EEPROM memorije čija je namjena pohranjivanje stalnih podataka koji se mogu mijenjati.

EEPROM memorija

Mikročip ATMega 328 je samo srce Arduina Uno, te posjeduje kilobajt električne izbrisive samo za čitanje namijenjene memorije EEPROM (Electrical Erasable Read- only Memory). EEPROM memorija je napravljena da zapamti podatke u svom sadržaju za duži niz godina. Unatoč svom imenu ona i nije memorija samo za čitanje (Read-only). Također je moguće vršiti i upisivanje.

Arduino naredba za čitanje i pisanje EEPROM memorije je jednako nezgodna za korištenje kao i korištenje PROGMEM naredbe. Morate čitati i pisati u, i iz, EEPROM-a jedan po jedan bajt.

Primjer koda koji omogućuje unos jednoznamenkastih slovnih oznaka iz Serial Monitora. Kod zatim pamti znamenku i više puta je ispisuje na Serial Monitor.

```
#include <EEPROM.h>
int addr = 0;
char ch ;
void setup()
{
Serial.begin (9600);
ch = EEPROM.read (addr);

}

void loop ()
{
if (Serial.available() > 0)
{
ch = Serial.read ();
```

```
EEPROM.write (0, ch);
Serial.println (ch);
}
Serial.println(ch);
delay (1000);
}
```

Za provjeriti ovaj kod, otvorite Serial Monitor te upišite neki novi znak. Tada isključite Arduino te ga opet priključite. Kada ponovno otvorite Serial monitor, vidjeti će te da je znak koji ste unijeli zapamćen.

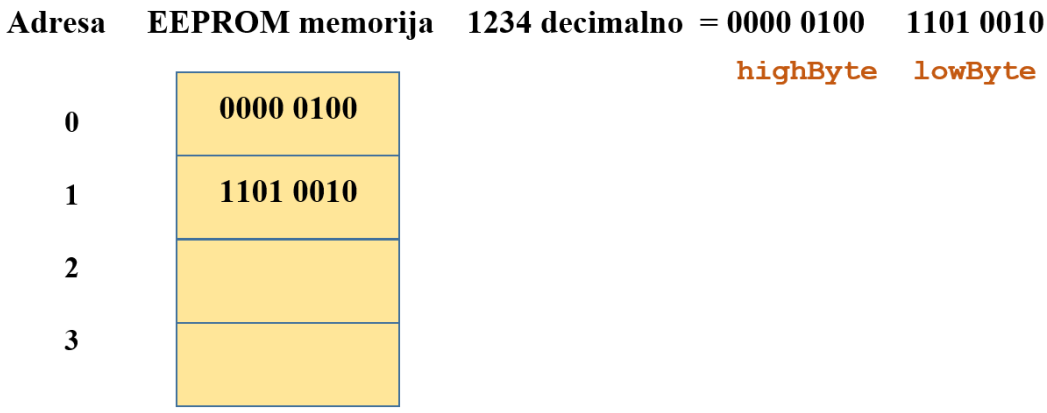
Funkcija **EEPROM.write** koristi dva argumenta. Prvi je adresa, koja je lokacija memorije u EEPROM-u i trebala bi biti između 0 i 1023. Drugi argument je podatak koji će se upisati na lokaciju. Ovo mora biti jedan bajt. Znak je predstavljen sa osam bitova, dakle to je uredu i moguće je napraviti, ali nije moguće direktno pohraniti 16 –bitovni **int**.

Pohranjivanje int u EEPROM-u

Da bi ste pohranili dva bajta **int** na lokaciji 0 i 1 unutar EEPROM-a, morate napraviti sljedeće:

```
int x = 1234;
EEPROM.write(0,highByte(x));
EEPROM.write(1,lowByte(x));
```

Funkcije **highByte** i **lowByte** su korisne za dijeljenje **int** u dva bajta. Slika pokazuje kako je ovaj **int** zapravo sačuvan u EEPROM-u.



Slika 24 pohranjivanje 16-bitnog integer-a u EEPROM

Da bi ste pročitali **int** iz EEPROM-a, trebate pročitati oba bajta iz EEPROM-a i rekonstruirati **int** na sljedeći način:

```
byte high = EEPROM.read(0);
byte low = EEPROM.read (1);
int x = (high << 8) + low;
```

Operator << je operator pomaka koji pomiče osam (**high**) bitova na vrh **int**-a te tada dodaje ostatak (**low**) bitova.

Pohranjivanje float u EEPROM (Unions)

Pohranjivanje **float** u EEPROM je posebno na svoj način. Za napraviti to morate koristiti značajku C jezika koja se zove „**unions**“. Ove su strukture podataka zanimljive jer se mogu smatrati načinom s kojim bi isto područje memorije bilo dostupno za više od jedne varijable. Štoviše, ove varijable mogu biti različite vrste sve dok su iste veličine u bajtovima.

Sljedeća „**union**“ definicija omogućuje oboma, **float** i **int** da koriste ista dva bajta memorije:

```
union data
{
float f;
int i;
} convert;
```

Tada možete staviti **float** u **union** na sljedeći način;

```
float f = 1.23;
convert.f = f;
```

Tada možete razdvojiti integer u njegova dva bajta za spremanje u EEPROM na sljedeći način:

```
EEPROM.write (0, highByte(convert.i));
EEPROM.write(1, lowByte(convert.i));
```

Ako želite vratiti **float** nazad, to zahtjeva obrnuti postupak. Prvo pridružite dva bajta u jedan **int**, te tada stavite **int** u **union** i povučete ga opet kao **float**.

```
byte high = EEPROM.read(0);
byte low = EEPROM.read (1);
int x = (high << 8) + low;
float.f = convert.f;
```

Pohranjivanje stringa u EEPROM

Pisanje i čitanje znakovnih nizova (stringova) u EEPROM je prilično jednostavno; ono što trebate je samo pisati svaki znak posebno, kao u sljedećem primjeru:

```
char *test = "Hello";
int i = 0;
while (test [i] != '0')
{
EEPROM.write(i, test[i]);
i++
}
```

Da bi ste pročitali string natrag u polje znakova morate napraviti sljedeće:

```
char test = [10];
int i = 0;
char ch;
ch = EEPROM.read (i);
while (ch != '0' && 1 < 10))
{
test[1] = ch;
ch = EEPROM.read (i);
i++
}
```

Čišćenje sadržaja EEPROM-a

Kada pišete u **EEPROM**, zapamtite da čak ni upisivanje novog koda u Arduino neće izbrisati **EEPROM**, stoga morate voditi računa o tome jesu li vam ostale neke stare vrijednosti u **EEPROM**-u od prethodnog koda. Sljedeći kod pretvara (briše) sav sadržaj **EEPROM**-a u nule:

```
#include <EEPROM.h>

void setup()
{
Serial.begin(9600);
Serial.println("Clearing EEPROM");
for (int i=0; i<1023; i++)
{
EEPROM.write(i, 0);
}
```

```
Serial.println("EEPROM Cleared");
}
void loop()
{

}
```

Također budite svjesni da možete upisivati i brisati na **EEPROM** lokaciju samo otprilike 100 000 puta prije nego ona postane nekorisna. Dakle samo pišite podatke u **EEPROM** kada to stvarno morate. **EEPROM** je također i prilično spor, treba mu otprilike oko 3 milisekunde da upiše jedan bajt.

Zadatak 1.

Ispitaj sadržaj cijele EEPROM memorije. Za lokaciju EEPROM memorije koja je različita od 0, zatim ispisati sadržaj i adresu na Serial monitoru, a u EEPROM memoriju upisati nule (ne pisati 0 na lokacije u kojimaje već pohranjena 0).

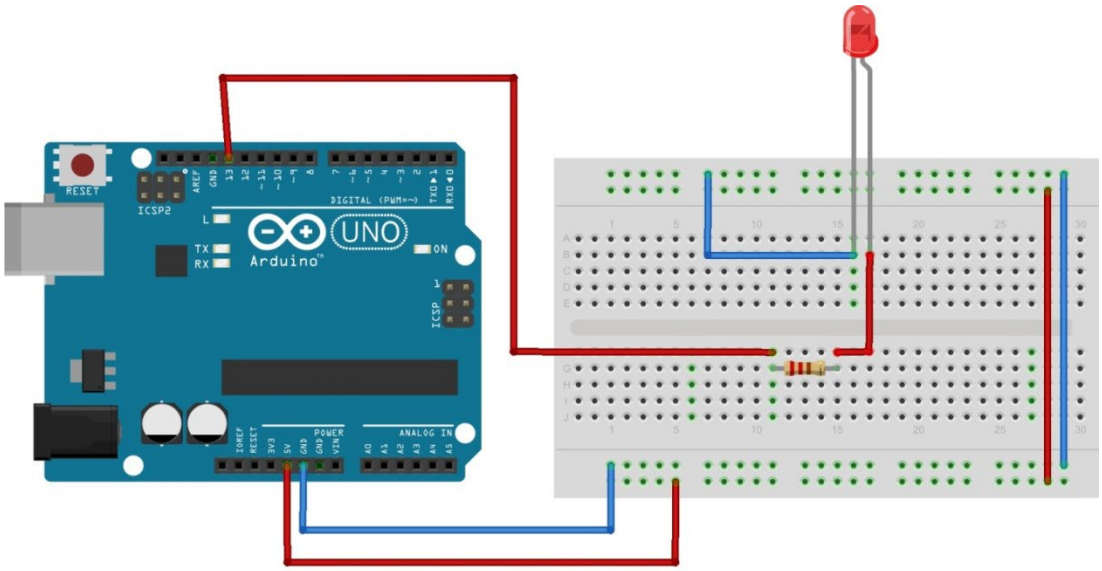
Rješenje

```
#include <EEPROM.h>   //Uključivanje EEPROM biblioteke, EEPROM je memorija koja ostane nakon gašenja napajanja

int vrijednost;   //Varijabla u koju ćemo spremati vrijednosti iz EEPROM-a kad ga budemo čitali

void setup()    //Početak setup() funkcije
{
  Serial.begin(9600);   //Naredba s kojom palimo serijsku komunikaciju kako bi mogli ispisivat sadržaj na Serial monitor
  Serial.println("Pocetak citanja EEPROM-a");   //Ispis da započinjemo citanje EEPROM-a
  for (int a = 0; a < 1024; a++)    //for petlja koja obuhvaća sve vrijednosti memorije, od 0 do 1023, to možemo iskoristit da prodamo kroz sve vrijednosti na svim adresama
  {
    vrijednost = EEPROM.read(a);     //Pridruživanje varijabli vrijednost sadrzaja EEPROM memorije na adresi a
    if (vrijednost != 0) {           //Provjera jeli vrijednost nula ili ima nekakav sadržaj, ako sadrži nešto ulazimo u petlju i ispisujemo taj sadržaj
      Serial.print("Adresa: ");      //Ispis podataka u formatu "Adresa: 'vrijednost adrese(0-1023)'   Vrijednost: 'vrijednost pohranjena na toj adresi'
```

```
      Serial.print(a);              //Za ispis koristimo Serial.print()
kako bi ispisali tekst/vrijednost i ostali u toj liniji ispisa
      Serial.print("\t");           //Serial.print("\t") ispisuje tab
razmak
      Serial.print("Vrijednost: ");
      Serial.println(vrijednost);    //Nakon Serial.println(vrijednost)
prelazimo u sljedeci red ispisa

      EEPROM.write(a, 0);           //Na adresu a EEPROM memorije upisuje
vrijednost 0, smatramo da time praznimo memoriju

      delay(100);                   //Pauza od 100ms, i nalazi se unutar
if petlje, kako bi se izvršavala samo kad imamo nešto za ispisat
    }
  }
  Serial.println("EEPROM memorija je prazna");    //Ispis da je memorija
prazna nakon sto ispišemo sav njen sadržaj ako je popunjena
}

void loop()   //Pocetak loop() funkcije, loop funkcija mora bit u programu
iako ostaje prazna
{
    //loop funcija je prazna
}
```

4.10 VJEŽBA br. 13 „C++ i Arduino biblioteke“

Koristeći Arduino spojiti sljedeću shemu:



Slika 25 „Blink“ shema spajanja

Arduino je jednostavan mikrokontrotler. Arduino kodovi su uglavnom prilično mali, te uz korištenje C programskog jezika rade savršeno. U svakom slučaju programski jezik za Arduino je ustvari C++ prije nego samo C. C++ je nastavak C programskog jezika, koji dodaje nešto što se zove **objektna orijentacija**.

Objektna orijentacija

Glavni cilj i razlog korištenja **objektne orjentacije** je podići mogućnost enkapsluacije vaših programa. Enkapsulacija drži bitne stvari skupa, te je to nešto što C++ jezik čini vrlo prikladnim za pisanje biblioteka kao što su one koje ste do sada već koristili u ranijim programima.

Klase i metode

Objektna orijentacija koristi koncept zvan „**klase**“ kako bi potpomogao enkapsulaciju. U svojoj osnovi klasa je kao skup programa koji uključuju obje varijable koje zovemo članske varijable (memebers variables) i metode, koje su kao funkcije ali primijenjene u klasama. Ove funkcije mogu biti javne, te u tom slučaju metode i funkcije se mogu koristiti u drugim klasama, ili privatne, u kojem slučaju metode mogu biti pozivane samo od strane druge metode unutar same klase.

Dok je Arduino kod sadržan u jednoj datoteci, kada radite u C + +, obično upotrebljavate više datoteka. Zapravo, za svaku klasu obično postoje dvije datoteke: datoteka **zaglavlje**, (heder file) koja ima dodatak **.h** i **datoteku implementacije**, koja ima dodatak **.cpp**

Zadatak 1.

U sljedećem zadatku 1, kreirat ćemo jednostavan primjer biblioteke koja bi trebala staviti koncepte iz biblioteke u određeni kontekst. Za tu svrhu koristiti ćemo od ranije poznat „Blink“ primjer programa.

Pisanje biblioteka

Dakle u ovom zadatku ćemo pretvoriti u biblioteku funkciju bljeskalice koja uzrokuje da LED dioda treperi određeni broj puta. Da biste stvorili C + + datoteke koje su potrebne za to, trebat će vam uređivač teksta poput Notepad-a ili Wordpad-a.

Datoteka zaglavlja

Započnite s izradom mape koja sadrži sve dijelove biblioteke. Morate napraviti ovu mapu unutar Arduino mape gdje su sadržane sve mape ostalih biblioteka. U sustavu Windows, mapa s vašim bibliotekama će biti u **My Documents \ Arduino\libraries**. a na Linuxu, bit će u imeniku sketchbooka, vašem kućnom imeniku. Ako u Arduinu nema mape s bibliotekama, stvorite je sami. Ova „Libraries“ mapa je mapa gdje stavljate bilo koju biblioteku koju napišete sami ili gdje bilo kakva „neslužbena“ biblioteka mora biti instalirana.

Otvorite tek napravljenu mapu „Flasher“. Pokrenite uređivač teksta (Notepad) te utipkajte u njega sljedeće:

```
//LED Flashing library
#include "Arduino.h"
class Flasher
{
public:
      Flasher (int pin, int duration);
      void flash (int times);
private:
      int _pin;
      int _d;
};
```

Spremite ovu datoteku u mapu **Flasher** s imenom **Flasher.h.** Ovo je datoteka zaglavlja za klasa biblioteke. Ova datoteka određuje različite dijelove klase. Kao što možete vidjeti, ona je podijeljena na javne i privatne dijelove.

Javni dio sadrži nešto što izgleda kao početak dviju funkcija. To su metode i razlikuju se od funkcija samo u mjeri u kojoj su povezane s klasom. One se mogu koristiti samo kao dio klase, te za razliku od funkcija, se ne mogu koristiti same.

Prva metoda, **Flasher**, počinje s velikim slovom, to je različito od funkcija. Ima isto ime kao i klasa. Ova metoda zove se **konstruktor**, koji možete primijeniti za izradu novog Flasher objekta koji će se koristiti u kodu.

Imamo sljedeći primjer:

```
Flasher slowFlasher (13, 500);
```

Tako ćemo stvoriti novi **Flasher** nazvan **slowFlasher** koji će izazvati treptanje na pinu **D13** s a trajanjem od 500 milisekundi.

Druga metoda u razredu naziva se **flash**. Ova metoda uzima jedan argument čiji je broj onoliko puta koliko trebamo treptaja. Budući da je povezan s klasom, kada ga želite nazvati morate se pozvati na objekt koji ste ranije stvorili, kako slijedi u nastavku:

```
slowFlasher.flash(10)
```

Ovo će uzrokovati da LED dioda trepti 10 puta u periodu koji ste odredili u **konstruktoru** za **Flasher** objekt.

Privatni dio klase sadrži definicije dviju varijabli: jednu za pin i jednu za vrijeme trajanja, koja je jednostavno nazvana „**d**“. Svaki put kada kreirate objekt klase **Flasher**, on će imati ove dvije varijable. Ovo omogućava da se zapamti pin i vrijeme trajanja, kada je novi **Flasher** objekt kreiran.

Ove varijable se zovu članske varijable, iz razloga što su one članovi klase. Njihova imena su neobična po tome što počinju sa znakom **donje crte**; u svakom slučaju to je samo uobičajena praksa, te nije obavezno u programskom smislu. Druga je praksa koja se koristi je korištenje malog slova **m** (za članove) kao prvog znaka u imenu varijable.

Datoteka implementacije

Datoteka zaglavlja samo određuje kako će klasa izgledati. Sada trebate drugu datoteku koja zapravo obavlja posao. Ona se zove **datoteka implementacije** i ima ekstenziju **.cpp**. Dakle stvorite novu datoteku koja sadrži sljedeće, te je spremite i imenujte kao **Flasher.cpp** untar Flasher Datoteke

```
#include "Arduino.h"
#include "Flasher.h"

Flasher::Flasher(int pin, int duration)
{
  pinMode(pin,OUTPUT);
  _pin = pin;
  _d = duration / 2;
}
```

```
void Flasher::flash(int times)
{
  for (int i = 0; i < times; i++)
  {
    digitalWrite (_pin, HIGH);
    delay(_d);
    digitalWrite(_pin, LOW);
    delay(_d);
  }
}
```

U ovoj datoteci postoji neka nepoznata sintaksa. Nazivi metoda (oba naziva) su unaprijed označeni sa **Flasher::**. To znači da te metode pripadaju klasi Flasher. **Konstruktor** metode (Flasher) jednostavno dodjeljuje svaki od njegovih parametara prikladnoj privatnoj članskoj varijabli. Parametar trajanja podijeljen je s dva prije dodjeljivanja na člansku varijablu _d. To je zato što se **delay** poziva dvaput, a čini se više logično za trajanje bude ukupno trajanje treptaja (dok svjetli LED) i vremena između treptaja.

Flash funkcija zapravo obavlja posao treptanja; obavlja ponavljanje odgovarajući broj puta, te tako uključivanje i isključivanje LED diode uz odgovarajuće kašnjenje.

Dovršavanje vaše biblioteke

Sada ste vidjeli sve bitne elemente za izradu biblioteke. Sada biste mogli upotrijebiti ovu biblioteku i ona bi radila sasvim u redu. Međutim, postoje još dva koraka koje bi ste vi trebali poduzeti kako bi dovršili svoju biblioteku. Jedno je da definiramo ključne riječi koje se koriste u knjižnici tako da ih Arduino IDE može prikazati u odgovarajućoj boji kada korisnici uređuju kod. Drugi je da uključite neke primjere kako koristiti biblioteku.

Ključne riječi (Keywords)

Da bi ste definirali ključne riječi, morate stvoriti datoteku koja se zove keywords.txt, koji se nalazi u Flasher direktoriju. Ova datoteka sadrži samo dvije sljedeće linije:

```
Flasher KEYWORD 1
Flasher KEYWORD 2
```

To je u biti tablica s dva stupca u tekstualnoj datoteci. Lijevi stupac je ključna riječ a desni stupac označava vrstu ključne riječi. Nazivi klase trebaju biti KEYWORD1 a metode trebaju

biti KEYWORD2. Nije važno koliko prostora ostavite između stupaca, ali svaka ključna riječ treba početi u novoj liniji.

Primjeri

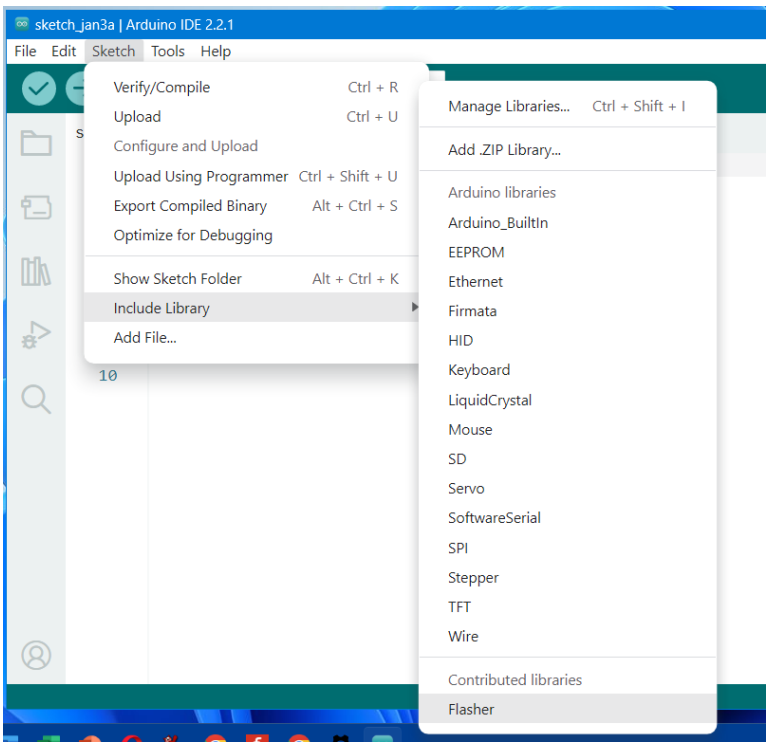
Sljedeća stvar koju biste, kao dobar Arduino programer, trebali uključiti kao dio biblioteke je mapa primjera. U ovom slučaju, biblioteka je toliko jednostavna da je svaki primjer suvišan.

Svi primjeri moraju biti postavljeni u mapu koja se zove primjeri (examples), unutar mape Flasher. Primjer je zapravo samo običan Arduino kod, tako da ga možete napraviti koristeći Arduino IDE. Ali prvo, trebate zatvoriti i ponovo otvoriti Arduino IDE kako bi nova biblioteka bila učitana.

Nakon ponovnog pokretanja Arduino IDE-a, na izborniku Arduino IDE odaberite, **File**, a zatim, **New**, za pisanje novog koda. Zatim iz izbornika odaberite **Sketch** i **Include Library** opciju. Postupak možete vidjeti na slici 1.

Biblioteke iznad linije su službene biblioteke; ispod su složene neslužbene biblioteke među koima su i one koje ste sami napisali kao što je naša Flasher biblioteka. Ako ste napravili sve kako treba vidjeti će te je na listi.

Ako Flasher biblioteka nije na listi, veoma je vjerojatno da Flasher mapa nije u mapi Libraries na putanji Document/Arduino/Libraries. Stoga provjerite.



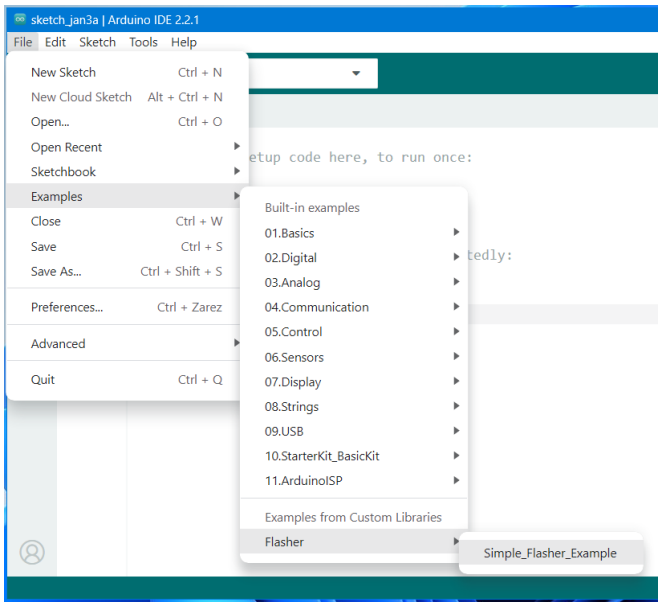
Slika 26 Uključivanje Flasher biblioteke

Utipkajte slijedeće u vaš prazan list za pisanje koda koji ste upravo kreirali:

```
#include <Flasher.h>

int ledPin = 13;
int slowDuration = 300 ;
int fastDuration = 100;

Flasher slowFlasher (ledPin, slowDuration);
Flasher fastFlasher (ledPin, fastDuration);


void setup()
{
}

void loop()
{
  slowFlasher.flash(5);
  delay(1000)
  fastFlasher.flash(10);
  delay(2000);
}
```

Arduino IDE vam neće dozvoliti da sačuvate kod **Primjer** direktno u **Libraries** mapu, stoga sačuvajte je negdje drugdje pod imenom **Simple Flasher Example** te tada maknite tu cijelu **Simple Flasher Example** mapu koju ste upravo napravili i sačuvali u u **examples** mapu u vašoj biblioteki.

Ako isključite te ponovno pokrenete vaš Arduino IDE, vidjeti će te da ste sada u mogućnosti otvoriti example kod iz izbornika kao što je prikazano na slici 2.



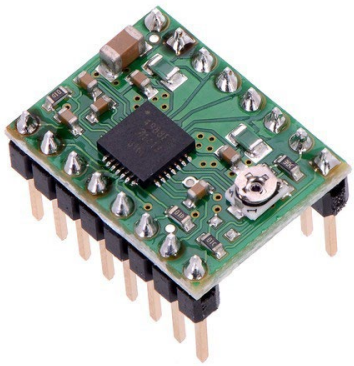
Slika 27 Otvaranje examples koda

5. HARDVERSKE VJEŽBE

5.1 VJEŽBA br. 14 „Upravljanje NEMA 17 koračnim motorom sa upravljačem Pololu A4988“

Pololu A4988 upravljač (driver)

A4988 je upravljač (driver) koji služi za upravljanje bipolarnih koračnih motora poput NEMA17 motora. Ovaj upravljač može provoditi jednostavne funkcije upravljanja na način da kontrolira motor sa samo dva pina. Jedan pin se koristi za upravljanje smjerom vrtnje a drugi pin za kontrolu broja koraka.



Slika 28 „Pololu A4988 upravljačka pločica“

Pololu A4988 moguće je ugoditi za upravljanje pet različitih veličina pokreta (koraka). Tako možemo koristiti:

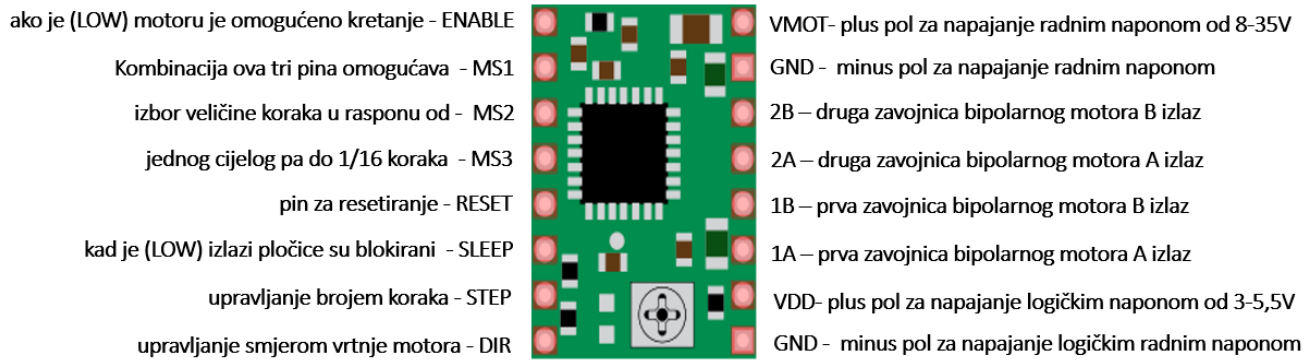
- Puni korak 200 koraka za jedan okret motora
- Pola koraka 400 koraka za jedan okret motora
- Četvrtinu koraka 800 koraka za jedan okret motora
- Osminu koraka 1600 koraka za jedan okret motora
- Šesnaestinu koraka 3200 koraka za jedan okret motora

Upravljačku pločicu A4988 ima logičke naponske razine od 3V- 5,5V, moguće je imati 2A jakosti struje po jednoj fazi ako imamo hlađenje pločice, bez hlađenja moguće je 1A po fazi, sama pločica se napaja radnim naponom od minimalno 8V do maksimalno 35V.

Minimalna logička naponska razina	3V
Maksimalna logička naponska razina	5,5V
Stalna struja po fazi	1A
Maksimalna (uz hlađenje) struja po fazi	2A
Minimalni radni napon	8V
Maksimalni radni napon	35V

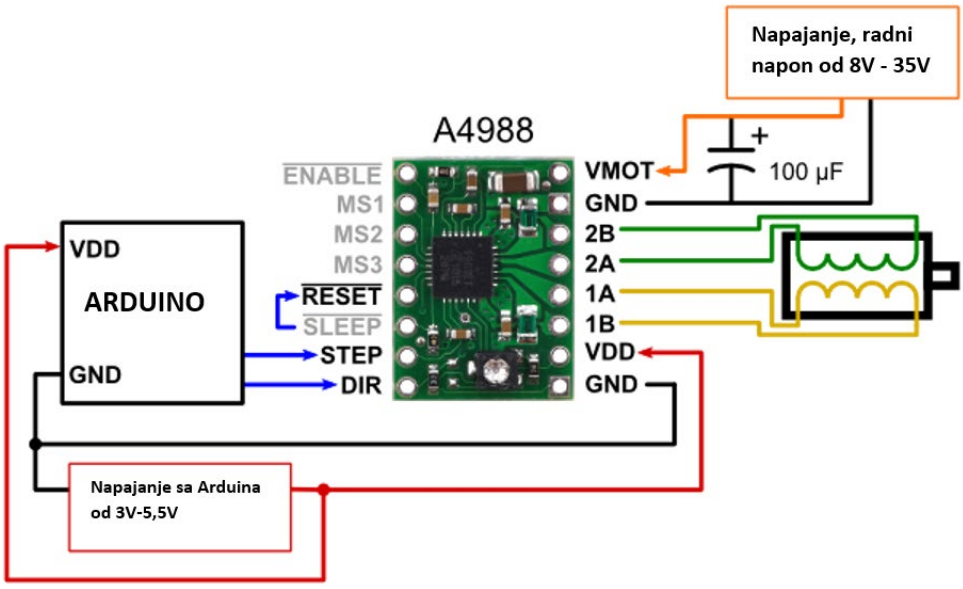
Pololu A4988 raspored pinova

Na slici 10 ispod prikazano je značenje pinova kao i spajanje pločice sa arduino mikrokontrolerskom pločicom te sa NEMA 17 bipolarnim koračnim motorom.



Slika 29 „Raspored pinova na A4988 upravljačkoj pločici“

Prilikom spajanja radnog napona potrebno je paralelno spojiti i kondenzator od bar 47µF iz razloga zaštite A4988 pločice od vršnih napona, prema slijedećem nacrtu spajanja.



Slika 30 „Dijagram spajanja A4988 pločice sa Arduinom i koračnim motorom“

Za rad A4988 pločice potrebno je spojiti slijedeće:

- Koračni motor
- Arduino upravljačku pločicu
- Radno napajanje sa zaštitom

- Pinove za odabir veličine koraka (ako ne koristimo puni korak)
- Spojiti RESET i SLEEP pinove

Koračni motor spajamo na pinove 1A, 1B, 2A, 2B na način da jednu zavojnicu koračnog motora spajamo na 1A i 1B a drugu zavojnicu koračnog motora na 2A i 2B pinove.

Arduino upravljačku pločicu spajamo na način da logičko napajanje ovisno o naponu od 3V ili 5,5V spajamo na pinove GND i VDD. Pinove za upravljanje motorom (digitalni pinovi, ovisno o odabiru pri pisanju programa) spajamo na pinove, STEP-za određivanje broja koraka i DIR – za odabir smjera vrtnje.

Radno napajanje u iznosu od 8V do 35 V spajamo na pinove VMOT i GND. Ovdje je preporučljivo paralelno spojiti i kondenzator radi zaštite A4988 pločice od vršnih napona jer ponekad vršni naponi (ovisno o priključenom naponu i radu motora) mogu preći granicu od 35V. Preporučljivi iznos kondenzatora je od minimalno 47 µF ili 100 µF.

Pinove SLEEP i RESET potrebno je međusobno spojiti iz razloga što tako omogućavamo rad pločice. U slučaju da imamo namjeru koristiti SLEEP funkciju tada na taj pin je potrebno spojiti logički (LOW) i u tom trenutku pločica upada u „sleep“ mod smanjuje potrošnju kad se motor ne koristi. Isto tako spajanjem logičkog (LOW) signala na RESET pin postavljamo A4988 pločicu u osnovno definirano početno stanje. Iz tog razloga direktnim spajanjem RESET i SLEEP pina, time na SLEEP pin spajamo logičko (HIGH) stanje te tako omogućavamo rad pločice.

MS1, MS2, i MS3 pinove koristimo za odabir vrste koraka kojim ćemo upravljati s našim motorom. Ako želimo koristiti puni korak tada sve ove pinove ostavljamo nespojene. U slučaju da želimo mijenjati veličinu koraka potrebno je na ove pinove spojiti slijedeća logička stanja prema slijedećoj tablici:

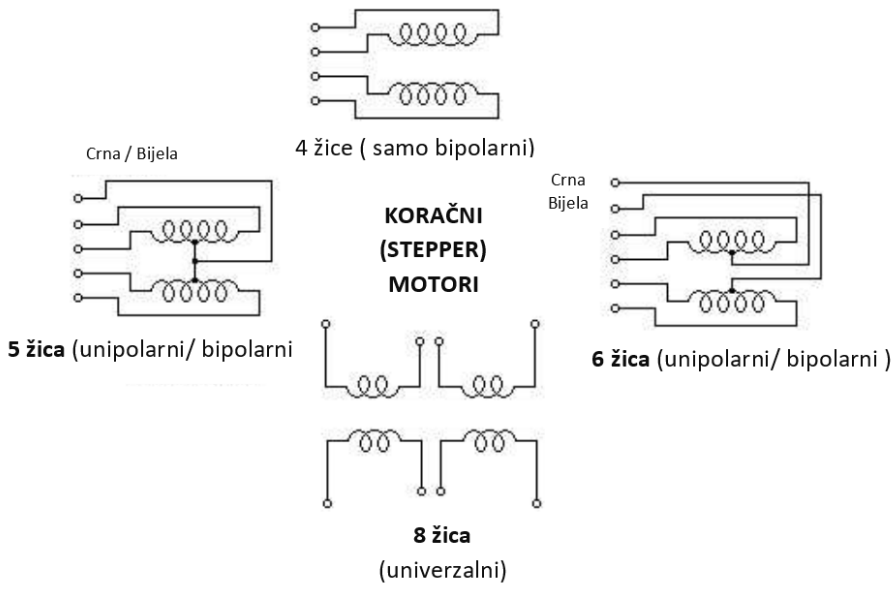
MS1	MS2	MS3	VELIČINA KORAKA	BROJ KORAKA U JEDNOM OKRETU
LOW	LOW	LOW	Puni korak	200
HIGH	LOW	LOW	1/2 koraka	400
LOW	HIGH	LOW	1/4 koraka	800
HIGH	HIGH	LOW	1/8 koraka	1600
HIGH	HIGH	HIGH	1/16 koraka	3200

Preostali ENABLE pin, služi za moguće isključivanje pločice. Ukoliko na ENABLE pin dovedemo logičko (HIGH), izlazi pločice biti će blokirani te će time pločica biti isključena za upotrebu.

NEMA 17 koračni bipolarni motor

Koračni motori u svojoj osnovi su pretvornici električne energije (impulsa) u odgovarajući mehanički pomak. Prilikom uzbude namotaja rotor se pokreće u željenom smjeru za predviđeni kut. Smjer vrtnje mijenjamo promjenom impulsnog slijeda (okretanjem polariteta) a brzinu

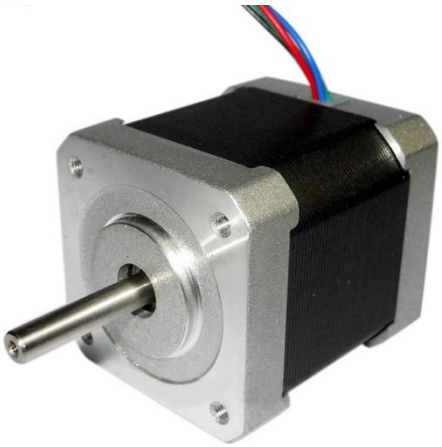
rotacije promjenom frekvencije impulsa. Prijeđeni kut ovisi o broju koraka odnosno o broju zaprimljenih impulsa.
Postoji nekoliko načina podjele koračnih motora. Jedan od načina razlikovanja je prema konstrukciji namotaja, napajanju. Na taj način razlikujemo bipolarne i unipolarne koračne motore.
Koračni motori mogu imati 4,5,6 ili 8 priključnih žica. Bipolarni koračni motori koriste za svoj rad isključivo 4 žice te zahtijevaju i bipolarni upravljač (driver, kontroller). Unipolarni koračni motori imaju dodatne žice, jednu ili više njih, za izmjeničnu pobudu jedne ili druge polovice namotaja za postizanje promjene struje a time i magnetskog polja. Kod bipolarnih to se radi promjenom vrijednosti napona na oba kraja namota.
Iz tog razloga unipolarne koračne motore možemo koristiti kao bipolarne. Za tu namjenu potrebno je samo isključiti (zanemariti) određeni broj žica (srednji statorski izvod).



Slika 31 „Način izvedbe bipolarnih i unipolarnih koračnih motora“

NEMA 17 označava standard koji je napisan od strane američkog udruženja proizvođača elektronike (National Electical Manufacturers Association). S tim standardom označen je promjer prednjeg kućišta motora. Broj 17 označava veličinu od 1,7“ (inch).

Postoje mnoge izvedbe s različitim karakteristikama NEMA 17 motora. U našim vježbama koristit ćemo NEMA 17 motore sa slijedećim karakteristikama:



Slika 32 „NEMA 17 koračni motor“

Postoje mnoge izvedbe s različitim karakteristikama NEMA 17 motora. U našim vježbama koristit ćemo NEMA 17 motore sa slijedećim karakteristikama ili slične:

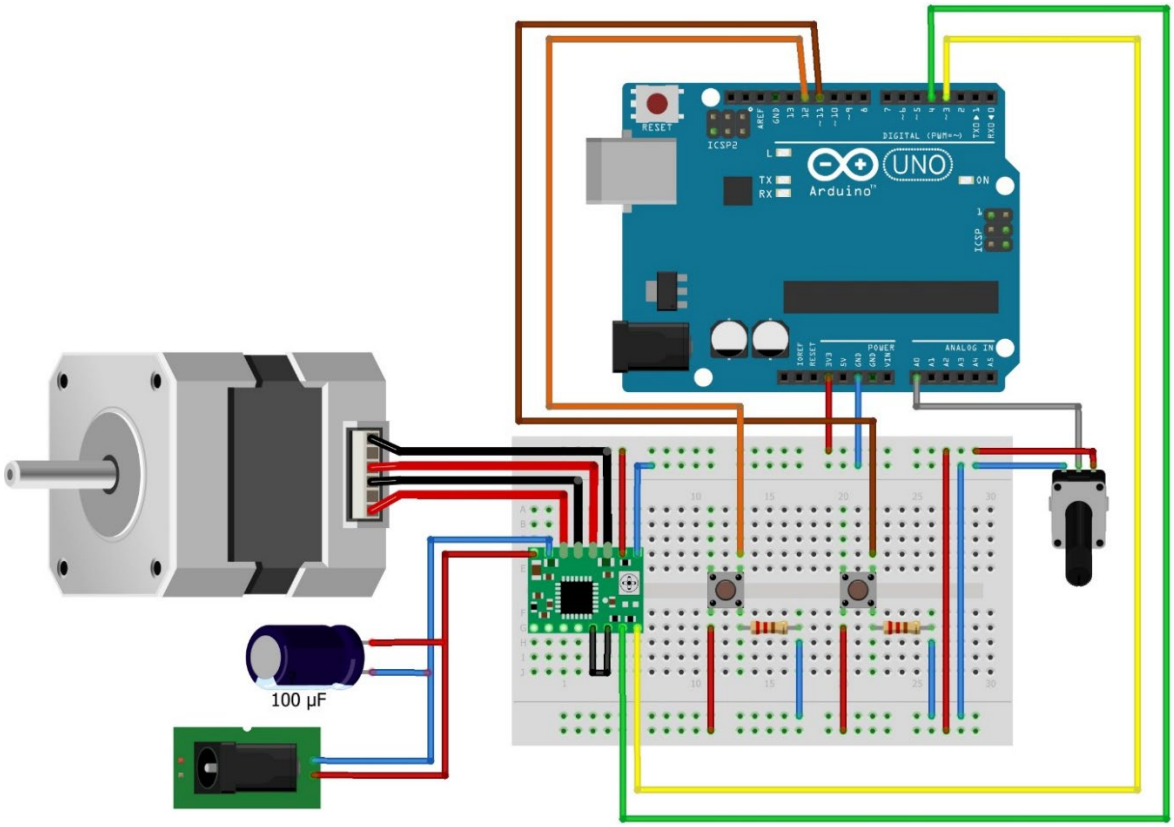
Model	17HS4417
Kut koraka	1,8°
Dužina motora	40 mm
Broj žica	4
Napon	2,55 V
Struja	1,7 A
Fazni otpor	1,5 Ω
Fazna indukcija	2,8 mH
Težina motora	288 g
Moment	60 oz-in ili 0,423 Nm

Raspored priključivanja vodova NEMA 17, 17HS4417 motora ovisno o boji žice na upravljačku pločicu A4988 prikazan je u slijedećoj tablici:

1A		crna
1B		zelena
2A		crvena
2B		plava

Zadatak 1

Spojiti Arduino pločicu s računalom te učitati program u mikrokontroler. Zatim spojiti shemu i napajanje Arduina kako je prikazano na slici te potom odgovoriti na pitanja.



Slika 33 „Shema spajanja NEMA 17 koračnog motora sa A4988 upravljačem“

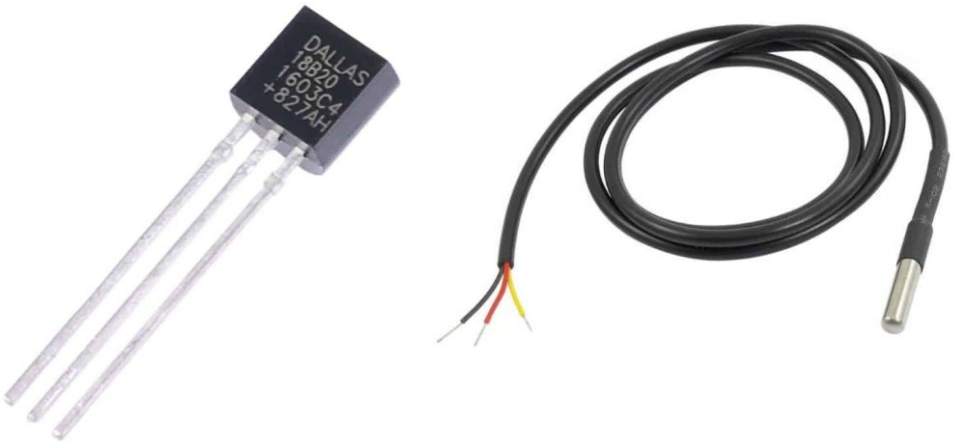
Kod

```
/*
„Upravljanje NEMA 17 koračnim motorom sa upravljačem Pololu A4988“

  Pritiskom na lijevu tipku, koračni motor se okreće u lijevu stranu, pritiskom na desnu tipku, okreće se u desnu stranu. Promjenom položaja potenciometra mijenja se brzina vrtnje motora.

 Krug
 * A4988 VCC na Arduino 3,3v, GND na Arduino GND
 * A4988 STEP na Arduino pin 3, DIR na Arduino pin 4
 * A4988 RESET I SLEEP spojiti međusobno.
 * A4988 1A, 1B, 2A, 2B, spojiti koračni motor.
 * A4988 VMOT i GND spojiti napajanje motora (8-35V) + kondenzator 100 µF
 * Tipka 1, na pin 11 Arduino, Tipka 2, na pin 12 Arduino.
 * Potenciometar signal na Arduino A0,


 Kod kreirali 05/2017
 Joško Smolčić & Marija Jelović
 */

//definicija varijabli i pinova
```

```
const int stepPin = 3;
const int dirPin = 4;

const int buttonPinL = 11;
const int buttonPinR = 12;

int buttonStateL = 0;
int buttonStateR = 0;

int delay_namjestanje,delay_mapiranje;

//definiranje ulaza i izlaza
void setup() {
  pinMode(buttonPinL, INPUT);
  pinMode(buttonPinR, INPUT);

  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
}


void loop() {

buttonStateL = digitalRead(buttonPinL);
buttonStateR = digitalRead(buttonPinR);

delay_mapiranje = ubrzanje();
// Pridružuje delay_mapiranje varijabli  vrijednosti dobivene iz
ubrzanje() funkcije te izrađuje prilagođeno kašnjenje, ovisno o
potenciometru, od kojeg ovisi brzina motora.

if(buttonStateR == HIGH){
  digitalWrite(dirPin,LOW);
  digitalWrite(stepPin,HIGH);
  delay(delay_mapiranje);
  digitalWrite(stepPin,LOW);
  delay(delay_mapiranje);
  }
  if(buttonStateL == HIGH){
  digitalWrite(dirPin,HIGH);
  digitalWrite(stepPin,HIGH);
  delay(delay_mapiranje);
  digitalWrite(stepPin,LOW);
  delay(delay_mapiranje);
  }}

  int ubrzanje()
{
  int delay_namjestanje = analogRead(A0); // Čitanje vrijednosti
potenciometra
  int nova_vrijednost = map(delay_namjestanje, 0, 1023, 1,30);
//Pretvaranje pročitanih vrijednosti sa potenciometra od 0 do 1023 u
željene vrijedosti delay-a između 1 i 30
  return nova_vrijednost;
}
```

5.2 VJEŽBA br. 15 Senzor temperature DS18B20

U raznim sustavima često se javlja potreba za mjerenje temperature. Jedan od najjednostavnijih i najjeftinijih načina je senzor DS18B20.



Slika 34 „DS18B20 TO-92 i vodonepropusna izvedba“

DS18B20 je jednožični (1-Wire) temperaturni senzor koji proizvodi tvrtka Dallas Semiconductor (preuzeta od strane Maxim Integrated). Budući da je to „1-wire“ uređaj, potreban mu je samo jedan digitalni pin za komunikaciju s mikrokontrolerom.

Senzor je dostupan u dva oblika. Jedan dolazi u pakiranju **TO-92**, koje nalikuje jednostavnom tranzistoru. Drugi dolazi u obliku vodootporne sonde, koja se uglavnom koristi kada trebate mjeriti temperaturu pod vodom, ispod zemlje ili u uvjetima izloženosti vanjskim vremenskim utjecajima. U ovoj vježbi ćemo koristiti ovaj drugi vodonepropusni model.

Senzor temperature DS18B20 prilično je precizan i ne zahtijeva nikakve vanjske komponente za rad. Ima temperaturni raspon od **-55°C** do **+125°C** i točnost od **±0,5°C**.

Razlučivost senzora temperature može se postaviti na 9, 10, 11 ili 12 bita. Međutim, zadana razlučivost pri uključivanju je 12-bitna (tj. preciznost od **0,0625°C**).

Senzor radi na napajanju od **3 V** do **5,5 V** i troši samo **1 mA** tijekom aktivne pretvorbe temperature.

DS18B20 senzor temperature	Tehničke karakteristike
Napajanje	3V do 5.5V
Potrošnja struje	1mA
Temperaturni opseg	-55 to 125°C
Točnost	±0.5°C
Rezolucija	9 do 12 bitova (ovisno o odabiru)

Vrijeme pretvorbe (izračuna)	< 750ms

DS18B20 senzor radi na principu izravnog pretvaranja temperature u digitalnu vrijednost. Njegova glavna značajka je mijenjanje broja bitova u skladu s promjenom temperature.

DS18B20 raspored pinova

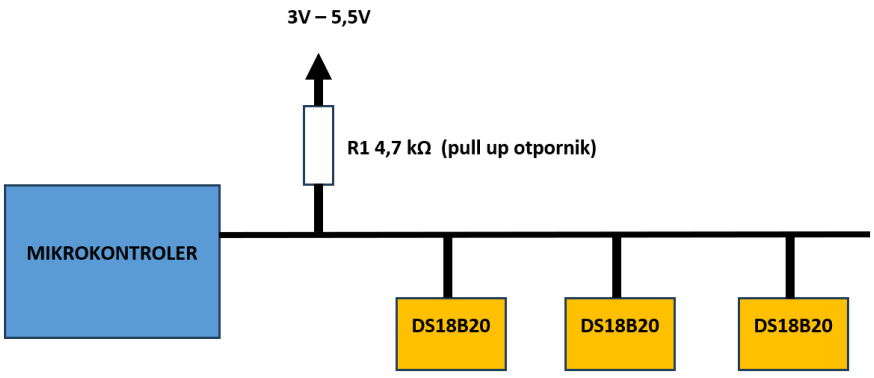
Slika 35 DS18B20 raspored pinova

Vcc	**Napajanje senzora u opsegu od 3,3 do 5 V**
Podatkovni kabel	**Podatkovni „1-Wire“ kabel koji se spaja na digitalni pin na mikrokontroleru**
Masa	**Spoj na masu**

Višestruki DS18B20 senzor na jednoj sabirnici

Jedna od bitnih značajki DS18B20 senzora je da više DS18B20 senzora može biti spojeno na istoj „1-Wire“ sabirnici. Budući da je svaki DS18B20 senzor unaprijed programiran s jedinstvenim 64-bitnim serijskim kodom, više senzora mogu se međusobno razlikovati.

Ova značajka može biti izuzetno korisna kada trebate kontrolirati više DS18B20 senzora ili izvršiti mjerenje temperature na velikom području, na način da rasporedite veći broj senzora na to područje mjerenja.



Slika 36 37 Način spajanja višestrukih DS18B20 senzora.

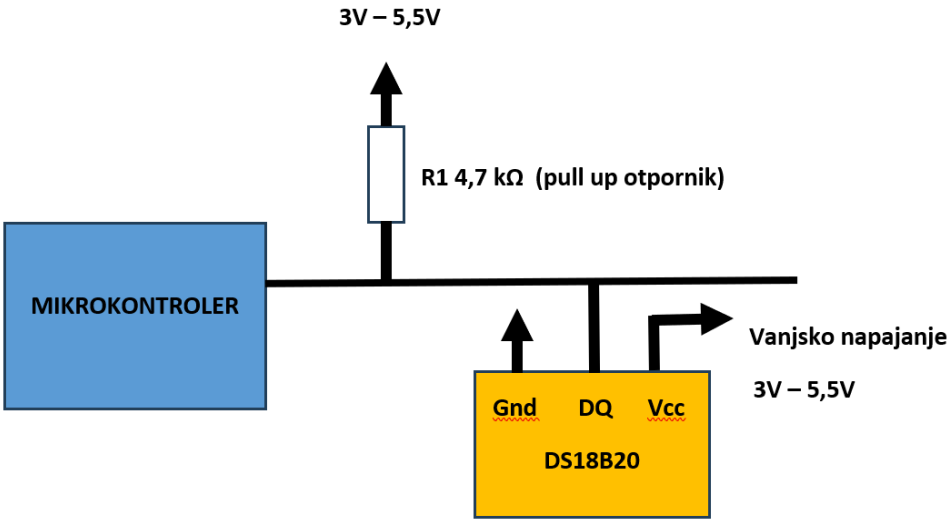
Vrste napajanja

Postoje dvije vrste napajanja DS18B20 senzora temperature, a to su:

- Vanjsko napajanje
- Parazitno napajanje

Kod metode **vanjskog napajanja**, osiguravamo napajanje DS18B20 senzora konvencionalnom metodom, tj. baterijom ili naponskim adapterom. Ova metoda je primjenjiva za temperature ispod **100** stupnjeva Celzijusa.

Prednost ove metode je to što nema dodatnog opterećenja na „pull up“ otporniku koji se koristi u ovoj metodi stoga ova metoda osigurava i veliku točnost mjerenja.

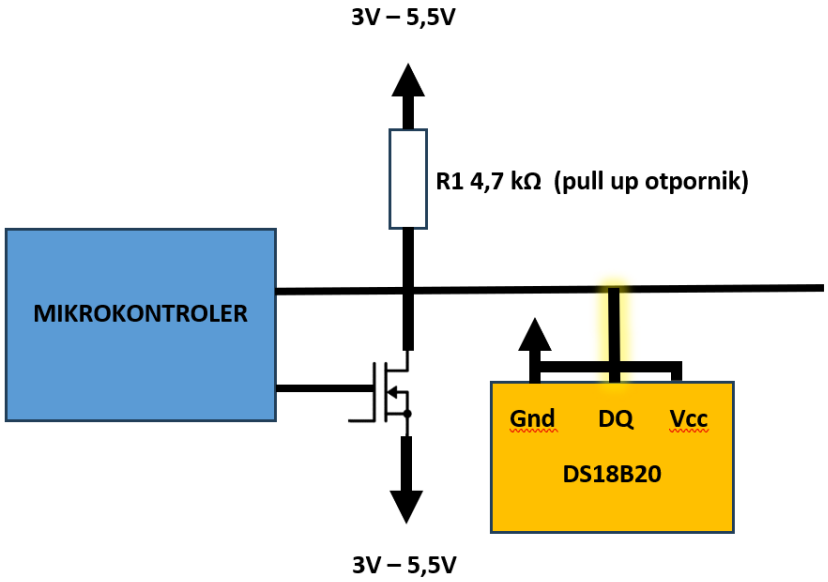


Slika 37 Vanjsko napajanje

Kod metode **parazitnog napajanja** ne trebamo posebno napajanje. Ova metoda se koristi za temperature veće od 100 Celzija.

U normalnoj situaciji, ova metoda osigurava dovoljnu struju i napon za DS18B20 senzor. Ali, u posebnom radu, kada DS18B20 pretvara temperaturnu vrijednost u digitalnu, trenutna vrijednost struje se povećava na takvu vrijednost struje koja može oštetiti otpornik. Za ograničenje struje i osiguravanje ispravnog mjerenja DS18B20 senzora, potrebno je kod ovakvog načina napajanja koristiti pull up mosfet tranzistor.

Budući da se ovakva vrsta napajanja koristi samo za određene vrijednosti temperature iznad 100 stupnjeva Celzija, najčešće se ipak primjenjuje vanjsko napajanje.

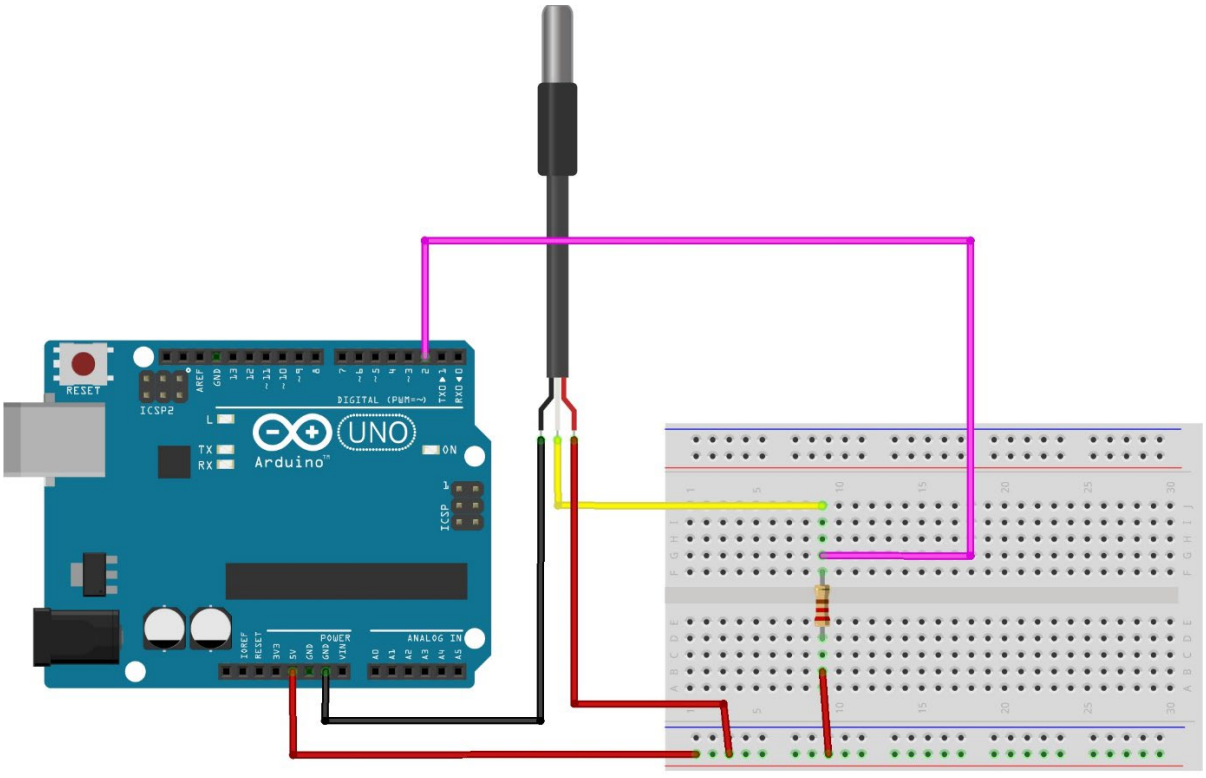


Slika 38 Parazitno napajanje

Zadatak 1.

Slijedeći nacrt sklopa potrebno je izvršiti spajanje DS18B20 senzora sa Arduino mikrokontrolerom uz korištenje breadboard pločice.

Započnite spajanjem crvene VDD žice sa senzora na Arduino 5V pin, pa zatim povežite crnu žicu sa senzora sa GND pinom na Arduinu. Zatim spojite žuti signalni pin DQ na Arduino digitalni pin broj 2.



Slika 39 Shema spajanja DS18B20 senzora

Kako bi prijenos podataka bio stabilan, morat ćete spojiti i **4,7k** (pull-up) otpornik između signalnih i energetskih pinova. Kako biste izbjegli pregrijavanje i oštećenje, provjerite je li DS18B20 ispravno spojen.

Učitavanje programa u Arduino pločicu.

Nakon što ste izvršili spajanje senzora sa Arduino pločicom potrebno je Arduino pločicu spojiti sa računalom, pokrenuti Arduino IDE program te izvršiti učitavanje koda a prije toga učitavanje biblioteka koje ovaj kod koristi.

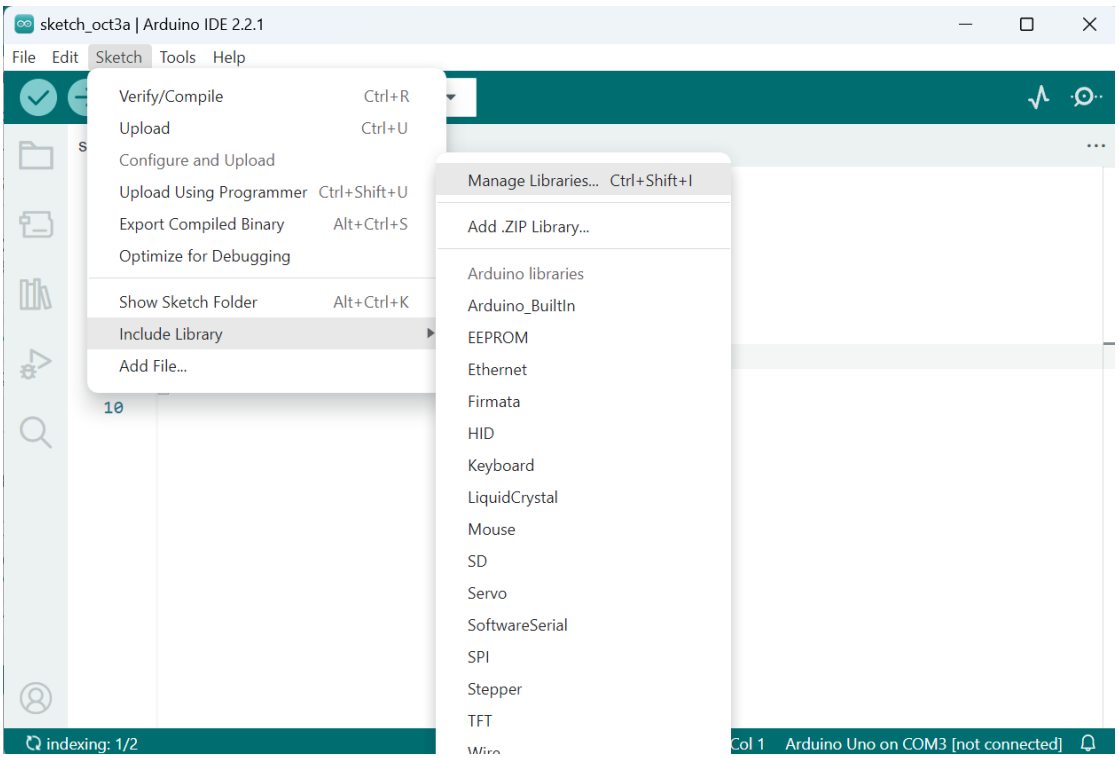
Za rad ovog koda potrebno je učitati dvije biblioteke. To su:

- OneWire.h
- DallasTemperature.h

Protokol „1-Wire“ je složen i potrebno je opsežan kod da bi ispravno funkcionirao. DallasTemperature.h biblioteka je napisana kako bilo moguće pojednostavniti kod, dopuštajući nam izdavanje jednostavnih naredbi za dobivanje očitanja temperature sa senzora.

Dallas Temperature je biblioteka specifična za hardver te ona obrađuje funkcije niže razine. Mora biti uparena s bibliotekom One Wire kako bi dobili mogućnost komunikacije više senzora s jednom žicom, a to se ne odnosi samo na naš DS18B20 senzor. To je razlog zašto instaliramo i ovu biblioteku.

Da biste instalirali biblioteku, idite na **Sketch** > **Include Library** > **Manage Libraries**…

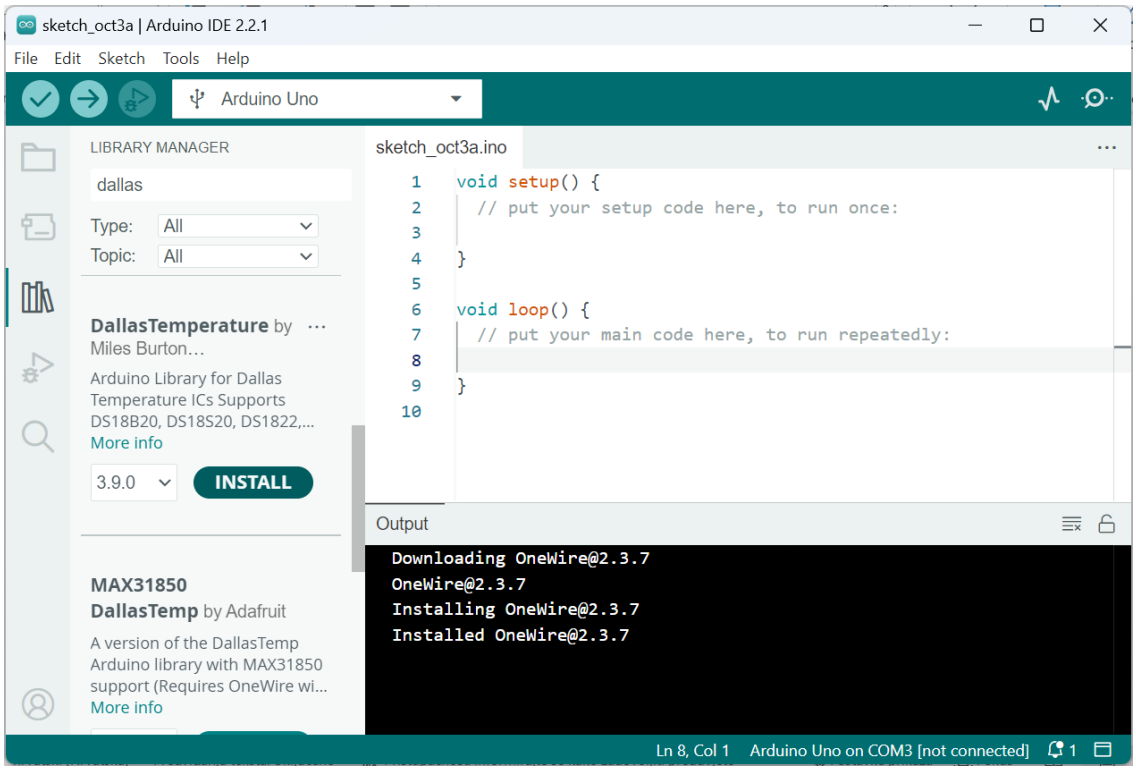


Slika 40 Sketch > Include Library > Manage Libraries…

Pričekajte da se **Libary Manager** potpuno učita i ažurira popis instaliranih biblioteka. Zatim u pretraživač upišite onewire te kada se pokažu biblioteke instalirajte OneWire biblioteku Jima Studa i Toma Pollarda.

Slika 41 OneWire biblioteke Jima Studa i Toma Pollarda.

Isti postupak ponovite i za učitavanje DallasTemperature biblioteke. U pretraživaču biblioteka upišite DallasTemperature te kada se biblioteka pojavi učitajte biblioteku od Miles Burtona



Slika 42 DallasTemperature biblioteka od Miles Burtona

Kod

```
/*********
Senzor temperature DS18B20
Program očitava temperaturu pomoću vodonepropusnog DS18B20 senzora u
stupnjevima Celzijusa. Podaci se primaju na pinu broj 2 od Arduino
pločice. Za tu svrhu se koriste dvije biblioteke: OneWire i
DallasTemperature.
Joško Smolčić 03.10.2023
*********/
//Učitavanje biblioteka
#include <OneWire.h>
#include <DallasTemperature.h>

// Pin na Arduinu na koji je spojen podatkovni vod sa DS18B29 senzora
const int oneWireBus = 2;

// Postavljanje OneWire reference za  omogućavanje jednožilne
komunikacije.
OneWire oneWire(oneWireBus);

// Proslijeđivanje našu oneWire reference senzoru temperature u
DallasTemperature biblioteci
DallasTemperature sensors(&oneWire);

void setup() {
  // Pokretanje serijske komunikacije za  Serial Monitor
  Serial.begin(9600);
  // Pokretanje DS18B20 senzora
  sensors.begin();
}

void loop() {
  //Zahtjev za očitanjem temperature sa senzora
  sensors.requestTemperatures();

  // Kreiranje varijable kojoj pridružujemo vrijednost temperature
  float temperaturaC = sensors.getTempCByIndex(0);

  //Ispisivanje vrijednosti varijable temperature
  Serial.print(temperaturaC);
  //Ispisivanje znaka za Celzijuse
  Serial.println("ºC");

  // Kašnjenje 5 sekundi
  delay(5000);
}
```

U `void loop()` funkciji pozivamo funkciju **requestTemperatures()**, koja upućuje sve senzore na sabirnici da izvrše konverziju temperature.

Zatim pozivamo funkciju **getTempCByIndex(deviceIndex)**, gdje je **deviceIndex** lokacija senzora na sabirnici. Ova funkcija čita očitanje temperature s odgovarajućeg senzora i vraća ga. Ako imate samo jedan DS18B20 na sabirnici, postavite **deviceIndex** na **0**.

Korisne funkcije u biblioteci DallasTemperature.h

Postoje mnoge korisne funkcije unutar biblioteke koje možete koristiti s objektom DallasTemperature. Ovo su neke najvažnije funkcije:

setResolution() - postavlja razlučivost internog ADC-a DS18B20 na 9, 10, 11 ili 12 bita, što odgovara 0,5°C, 0,25°C, 0,125°C i 0,0625°C.

bool getWaitForConversion() - vraća vrijednost oznake **waitForConversion**, koja je korisna kada se utvrđuje je li pretvorba temperature dovršena ili nije.

setHighAlarmTemp() i **setLowAlarmTemp()** - konfiguriraju unutarnje alarme za visoku i nisku temperaturu uređaja u stupnjevima Celzija. Prihvatljivi temperaturni raspon je od -55 do 125°C.

bool hasAlarm() - funkcija vraća **true** kada temperatura premaši bilo visoku ili nisku postavljenu točku alarma.

5.3 VJEŽBA br. 16 Plastični plutajući prekidač senzor nivoa tekućine.

Plastični plutajući prekidač razine tekućine je vrlo jednostavan senzor koji radi kao prekidač. Ovaj se senzor postavlja okomito na tekućinu čiji nivo pratimo. Radi na jednostavnom i pouzdanom principu, koristi se plutajući plovak koji se diže i spušta s razinom tekućine. Plovak je pričvršćen na mehanizam prekidača koji djeluje na temelju magnetskog polja kada razina tekućine dosegne određenu točku.

Neki od uobičajenih načina primjene ovog senzora nivoa tekućine su:

- kontrola pumpi
- indikacija razine vode u spremniku
- alarmiranje nivoa
- uparivanje s drugim uređajima za praćenje i kontrolu

Slika 43 Plastični plutajući prekidač senzor nivoa tekućine.

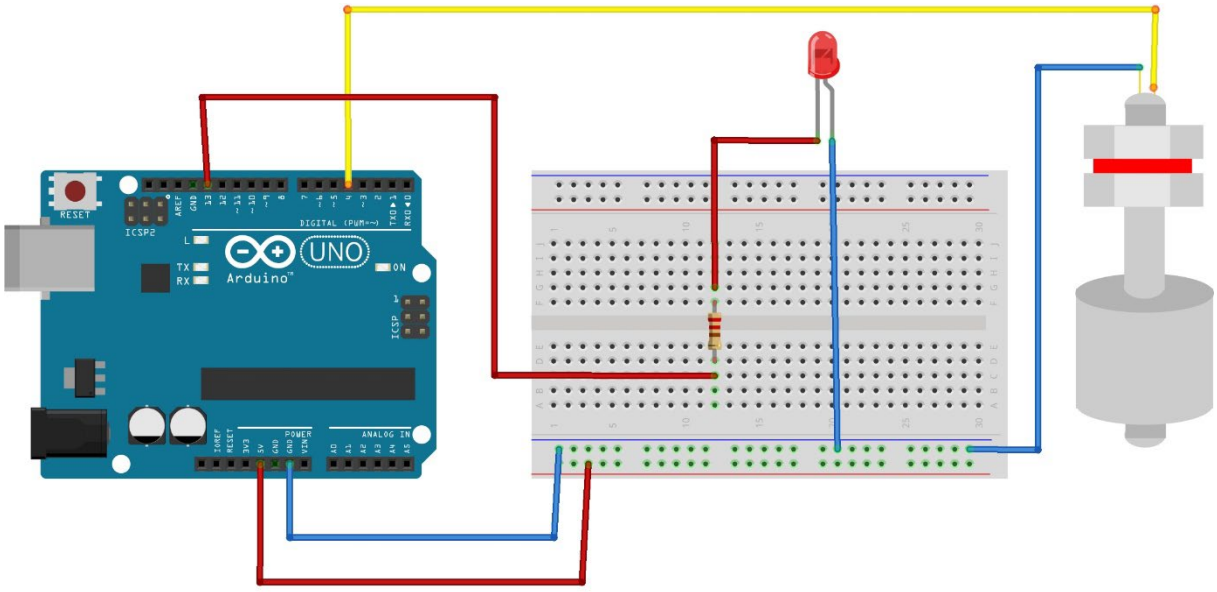
Plovak	Tehnički podaci:
Maksimalna snaga kontakta: 10 W	**10 W**
Maksimalni sklopni napon: 220V DC/AC	**220V DC/AC**
Maksimalna sklopna struja	**1,5 A**
Maksimalni probojni napon	**300 V DC/AC**
Maksimalna nosiva struja	**3A**
Maksimalni kontaktni otpor	**100 m ohma**
Temperatura	**-10 / +85 Celsuisa**
Promjer navoja	**9,5 mm / 0,374"**
Veličina kućišta prekidača	**23,3 x 57,7 mm / 0,9" x 2,27" (maks. D*V)**

Duljina kabela	**36 cm / 14,2"**
Neto težina	**70g**

Zadatak 1.

Spojiti Arduino pločicu s računalom te učitati “Plovak” program u mikrokontroler. Zatim spojiti napajanje Arduina, LED diodu te plovak kako je prikazano na slici.

Nakon provjere ispravnosti programa i hardvera okrenuti senzor za 180 stupnjeva te prilagoditi program novonastaloj situaciji.



Slika 44 Shema spajanja za plastični plutajući prekidač senzor nivoa tekućine

Senzor je spojen na Pin 4 i LED dioda na pin 13 koji ima ugrađenu LED diodu na samoj Arduino ploči. Nakon pokretanja serijske komunikacije se digitalne vrijednosti čitaju sa senzora i pohranjuju u varijablu.

Kada na varijabli "PLOVAK" ima napona odnosno nalazi se u visokom stanju, stanju (1), to označava da je razina vode niska i da tek treba dosegnuti predviđenu granicu. Sada Arduino ispisuje poruku "IZVAN VODE" i isključuje LED diodu.

Kada na varijabli "PLOVAK" nema napona odnosno nalazi se u niskom stanju, stanju (0), to označava da je razina vode visoka i da treba ograničiti dodatni dotok vode. Sada Arduino ispisuje poruku "U VODI" i isključuje LED diodu.

Kod:

```
/*
  Plastični plutajući prekidač, senzor nivoa tekućine
  Joško Smolčić, 09.10.2023.
*/

#define PLOVAK  4     // broj pina za plovak
#define LED          13      // broj pina za LED diodu

void setup()
{
  // inicijalizacija LED pina kao izlaza
  pinMode(LED, OUTPUT);
  // Inicijalizacija pina za plovak kao ulaza

  pinMode(PLOVAK, INPUT_PULLUP); // Unutarnji Arduino otpornik 10k

}

void loop()
{
  if(digitalRead(PLOVAK) == LOW)
  {
    // upali (LED on)
    digitalWrite(LED, HIGH);
    Serial.println("U vodi");
  }
  else
  {
    // ugasi  (LED off)
    digitalWrite(LED, LOW);
    Serial.println("Izvan vode");
  }
}
```

5.4 VJEŽBA br. 17 Beskontaktni senzor razine tekućine.

Beskontaktni senzor razine tekućine XKC-Y25-V digitalni je senzor koji otkriva prisutnost tekućine. Ovaj beskontaktni senzor razine tekućine može otkriti visinu tekućine u zatvorenom spremniku, prevladavajući problem debelih stjenki spremnika. Na modulu se nalazi LED dioda koja će se uključiti ako se otkrije tekućina, inače se gasi. Na tijelu modula nalazi se i vijak pomoću kojeg se podešava osjetljivost senzora. Ovaj senzor je prikladan za primjene kao što je otkrivanje otrovnih tekućina poput jakih kiselina, jakih lužina i drugih tekućina koje potencijalno mogu biti kontaktno opasne.

Slika 45 Beskontaktni senzor razine tekućine XKC-Y25-V

Senzor radi tako što za detekciju vode i otkrivanje prisutnosti tekućine koristi kondenzator. U nedostatku tekućine u blizini senzora, bit će prisutan raspodijeljeni kapacitet, pa stoga postoji određeni statički kapacitet prema masi na senzoru. Ako je tekućina blizu senzora, konačni kapacitet senzora raste. Tada će se promjenjivi kapacitet pretvoriti u neku varijaciju signala. Zatim će određeni algoritam otkriti i odrediti kolika je ta varijacija. Kada se prekorači određena vrijednost praga, smatra se da je razina tekućine dosegla točku indukcije.

Postoji NPN i PNP izvedba ovog senzora. Ovisno o odabiru senzora mijenjate o podešavanje na MODE kabelu.

Ovaj modul ima 4 žice:

VCC: Napajanje modula – 5V do 24V – smeđe

OUT: Izlaz senzora – žuto

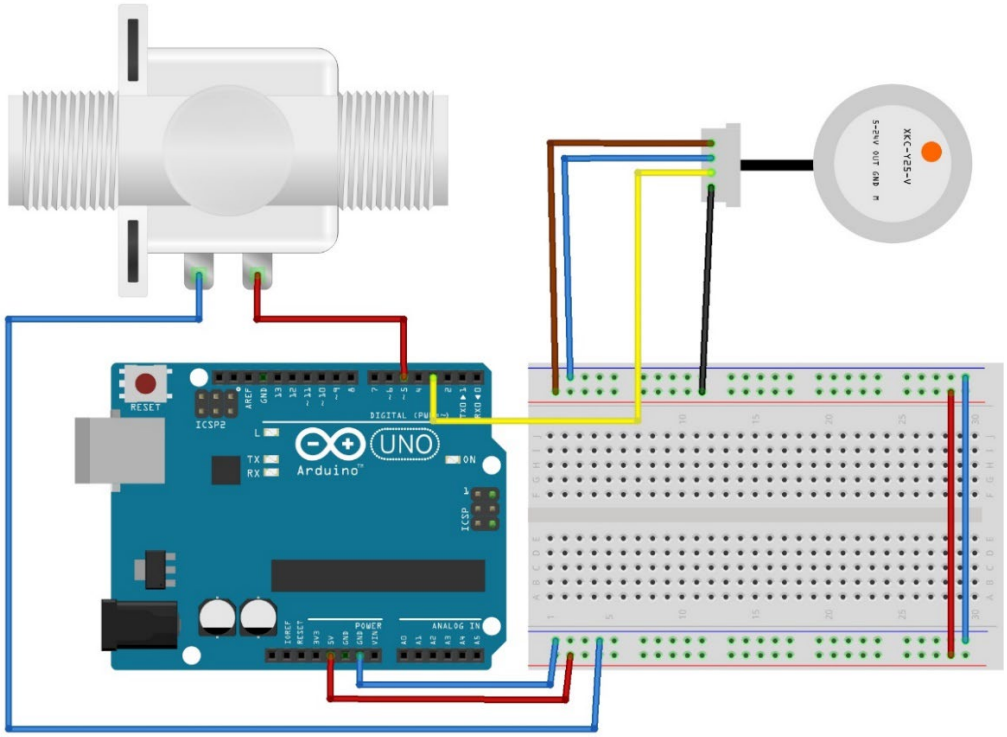
GND: Uzemljenje – plavo

MODE: Pin za odabir načina rada (ako je Low, izlaz će biti Active Low, a ako je High, izlaz će biti Active High. Ako nije spojen, izlaz ostaje Active High) – crn.

Naziv	Tehnički podaci
Model:	**XKC-Y25-NPN**
Izlaz:	**NPN**
Ulazni napon (InVCC):	**DC 5~12V**
Struja:	**5mA**
Izlazni napon (visoka razina):	**InVCC**
Izlazni napon (niska razina): 0V	**0V**
Izlazna struja:	**1~100mA**
Vrijeme odziva:	**500 mS**
Radna temperatura:	**0~105°**
Debljina indukcije (osjetljivost):	**0~13 mm**
Komunikacija:	**RS485**
Vlažnost:	**5%~100%**
Materijal:	**ABS**
Zaštita od prodora:	**IP67**
Model:	**XKC-Y25-NPN**

Zadatak 1.

Spojiti Arduino pločicu s računalom te učitati "Senzor nivoa vode" program u mikrokontroler. Zatim spojiti napajanje Arduina, beskontaktni senzor i elektromagnetski ventil kako je prikazano na slici.



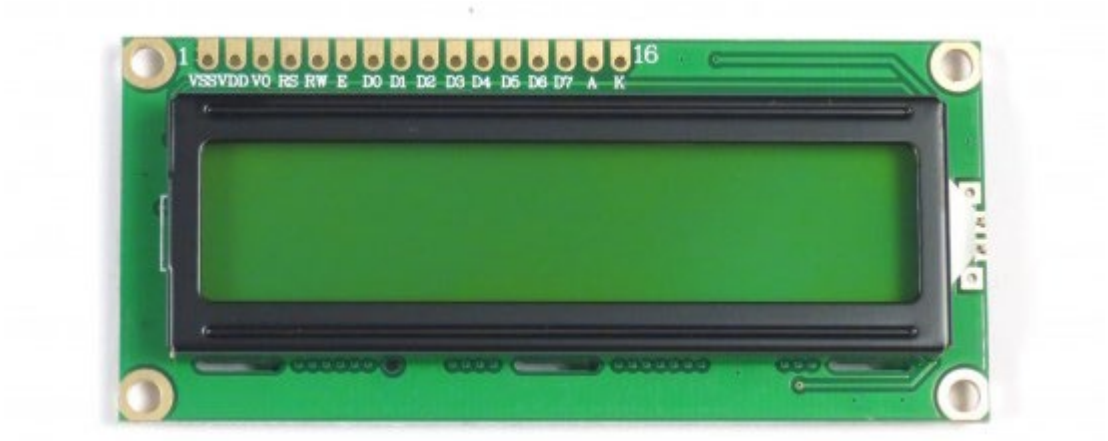
Slika 46 Shema spajanja za beskontaktni senzor razine tekućine XKC-Y25-V

Kod:

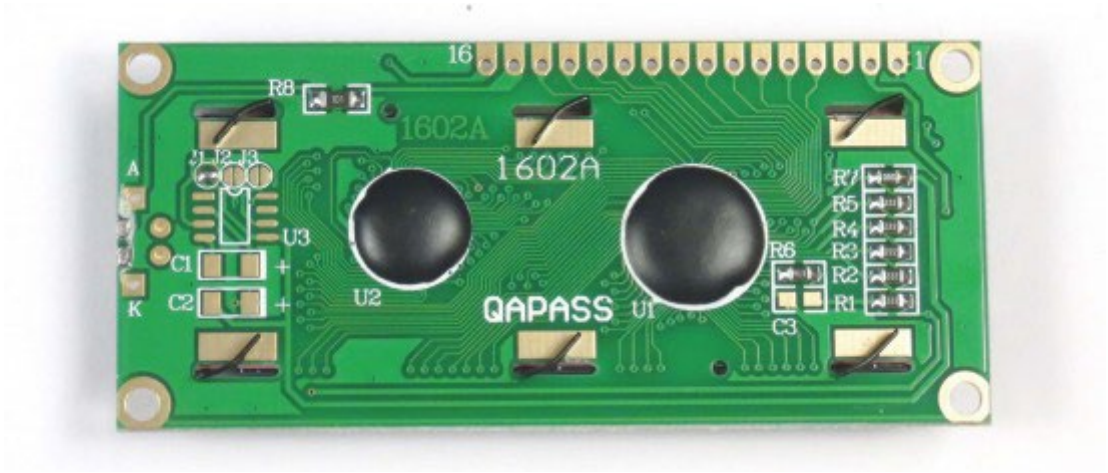
```
//Senzor nivoa vode: XKC-Y25-V
//Autor: Joško Smolčić 06.04.2020

#define s1_IZLAZ 5
#define s1_ULAZ 3

int senzor_1;


void setup()
{
 Serial.begin(9600);
  pinMode(5, OUTPUT);
  pinMode(3, INPUT);
}

void loop()
{
  //------------------------------SENZOR_1 ------------------------------
----------------------
  senzor_1 = digitalRead(s1_ULAZ);

   // ispitivanje stanja senzora. Senzor ima dvije vrijednosti 0 i 1.
   // kad nema vode onda je 0, kad ima vode onda je 1.
  if (senzor_1 <= 0)
  {
    digitalWrite(s1_IZLAZ, HIGH);
  }
 else
  {
    digitalWrite(s1_IZLAZ, LOW);
 }

 //Ispis stanja bezkontaktnog senzora na Serial Monitoru
  Serial.print("Sensor_1 =  ");
  Serial.println(senzor_1);

  //Kašnjenje 300 ms u svrhu lakšeg ispisa na Serial Monitoru.
  delay(300);


}
```

5.5 VJEŽBA br. 18 LCD zaslon 16 x 2

LCD zaslon veličine 16 x 2, je prikladan način za pregled podataka s Arduina. On nam omogućuje vidjeti vrijednosti koje koristimo na Arduinu bez pristupa računalu? Zasloni s tekućim kristalima (LCD) izvrsni su za prikaz niza riječi ili podataka senzora.



Slika 47 LCD zaslon veličine 16 x 2



Slika 48 LCD zaslon veličine 16 x 2, straga

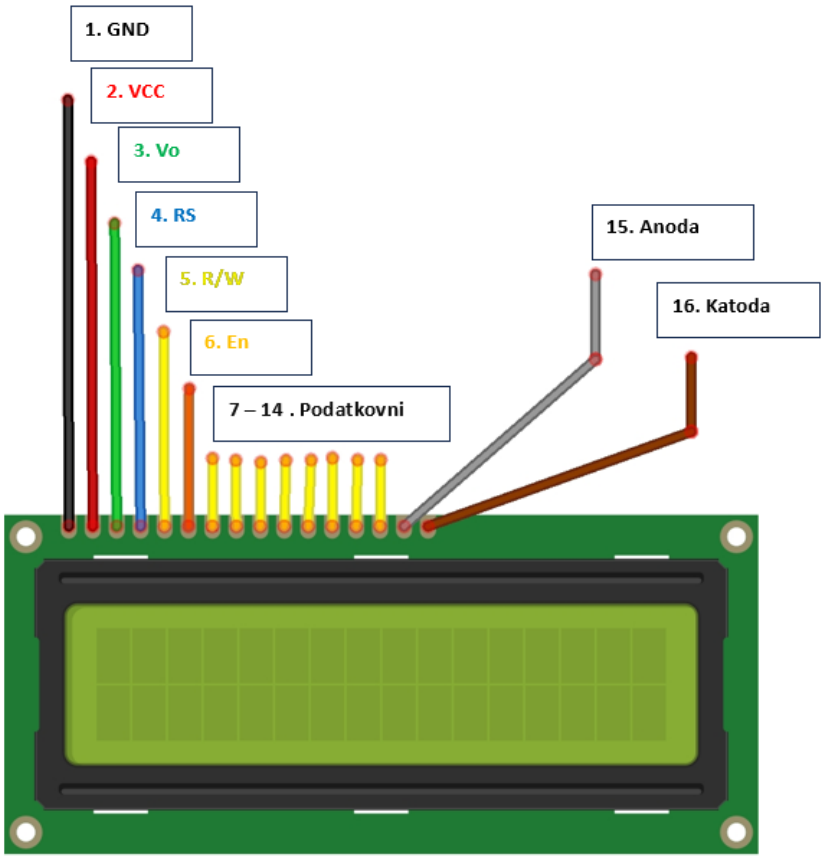
LCD (Liquid Crystal Display) ili zaslon s tekućim kristalima vrsta je zaslona koji koristi tekuće kristale za prikaz znakova. Kada se aktiviraju električnom strujom, ovi tekući kristali postaju neprozirni, blokirajući pozadinsko osvjetljenje koje se nalazi iza zaslona. Kao rezultat toga, to će područje biti tamnije od ostatka. Aktiviranjem sloja tekućih kristala u određenim pikselima mogu se generirati znakovi.

Ako pažljivo pogledate, možete vidjeti sićušne pravokutnike za svaki znak na ekranu, kao i piksele koji čine znak. Svaki od ovih pravokutnika je mreža od 5×8 piksela.

LCD zasloni sa znakovima dostupni su u raznim veličinama i bojama, 16×1, 16×2, 16×4, 20×4. Tu su moguće i razne kombinacije boja i kontrasta pa tako imamo: bijeli tekst na plavoj pozadini (to je najčešća kombinacija), crni tekst na zelenoj pozadini i mnoge druge. Jedna od prednosti korištenja bilo kojeg od ovih zaslona u vašem projektu je ta što su "zamjenjivi", što znači da ih možete jednostavno zamijeniti drugim LCD-om druge veličine ili boje. Vaš će kod biti potrebno malo dotjerati, ali ožičenje će ostati isto!

Raspored pinova na LCD 16x2 zaslonu

Standardni LCD 16x2 zaslon ima 16 pinova.



Slika 49 LCD zaslon veličine 16 x 2 raspored pinova

GND – pin za uzemljenje, masa.

VCC - pin za napajanje LCD-a i obično je spojen na napon od 5 volti.

Vo (LCD Contrast) - pin za kontrolu kontrasta LCD-a. Koristeći naponski dijelitelj i potenciometar, možemo izvršiti precizno podešavanja kontrasta.

RS (Register Select) - pin se koristi za odvajanje naredbi od podataka (kao što je postavljanje pokazivača na određenu lokaciju, brisanje zaslona itd.). RS pin je postavljen na LOW kada šalje naredbe na LCD i HIGH kada šalje podatke.

R/W (Read/Write) - pin omogućuje čitanje podataka s ili pisanje podataka na LCD. Budući da se LCD koristi samo kao izlazni uređaj, ovaj pin je obično postavljen LOW. Tada je LCD u načinu rada WRITE.

E (Enable) - pin se koristi za omogućavanje prikaza. Kada je ovaj pin postavljen na LOW, LCD ignorira aktivnost na R/W, RS i linijama sabirnice podataka; kada je postavljen na HIGH, tada LCD obrađuje dolazne podatke.

D0-D7 (Data Bus) - pinovi od D0 do D7 služe za prijenos 8-bitnih podataka koje šaljemo na zaslon. Na primjer, da bismo vidjeli veliko slovo 'A' na zaslonu, ove pinove postavljamo na 0100 0001 (prema ASCII tablici).

A-K (Anode & Cathode) – ovi pinovi se koriste za kontrolu pozadinskog osvjetljenja LCD-a.

Priključivanje napajanja LCD zaslona.

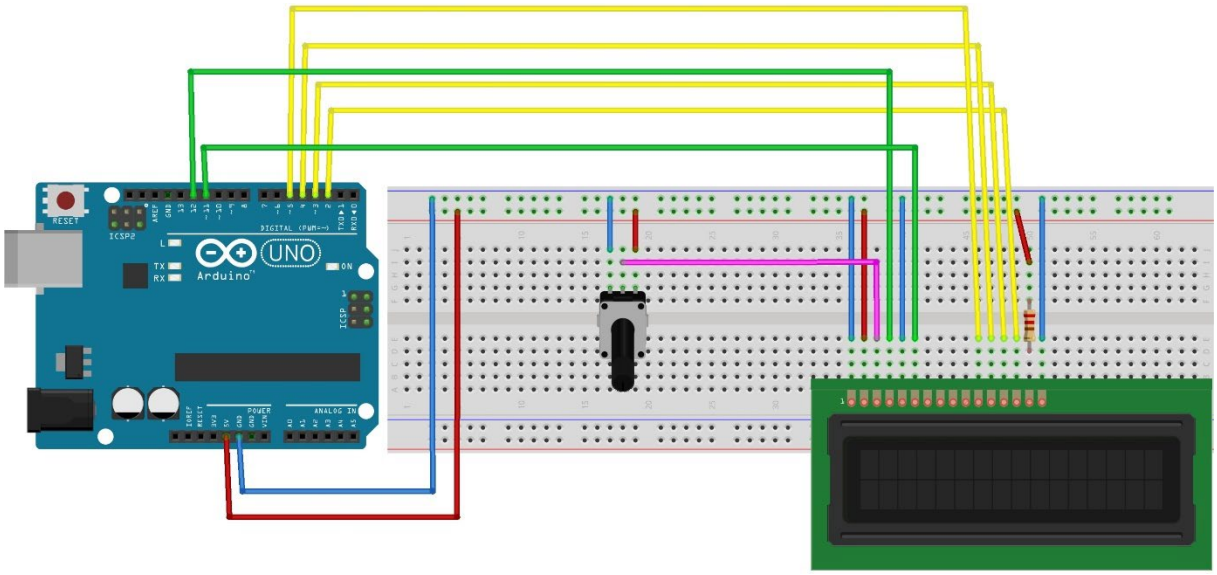
LCD ima dva odvojena priključka za napajanje: jedan za LCD (pinovi 1 i 2) i jedan za LCD pozadinsko osvjetljenje (pinovi 15 i 16). Spojite LCD pinove 1 i 16 na GND i 2 i 15 na 5V.

Ovisno o proizvođaču, neki LCD-i uključuju otpornik za ograničavanje struje za pozadinsko osvjetljenje. Nalazi se na stražnjoj strani LCD-a, blizu pina 15. Ako vaš LCD ne sadrži ovaj otpornik ili ako niste sigurni sadrži li ga, morate dodati jedan između 5V i pina 15. Trebalo bi biti odgovarajuće koristiti otpornik od 220 ohma, iako ovako visoka vrijednost može malo prigušiti pozadinsko osvjetljenje. Za bolje rezultate provjerite datasheet vašeg LCD-a te iznose za maksimalnu struju pozadinskog osvjetljenja i odaberite odgovarajuću vrijednost otpornika.

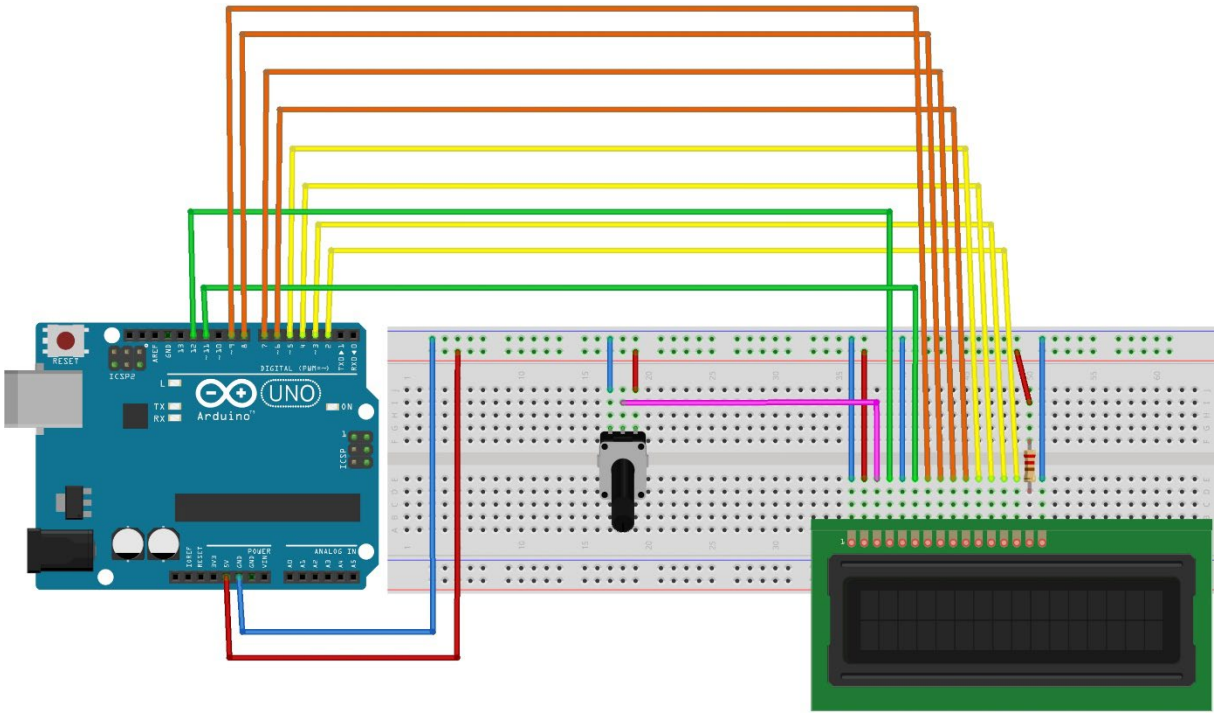
Zatim spojimo potenciometar na zaslon. Ovo je neophodno za fino podešavanje kontrasta zaslona za najbolju vidljivost. Spojite jednu stranu 10K potenciometra na 5V, a drugu na uzemljenje potom srednji izvod sa potenciometra spojite na LCD pin 3.

Povezivanje LCD zaslona i Arduina.

Prilikom povezivanja Arduina i LCD zaslona imamo dvije mogućnosti za povezivanje. Prva mogućnost je korištenje **8-bitnog** načina povezivanja a druga mogućnost je izbor **4-bitnog** načina povezivanja LCD zaslona.



Slika 50 4-pinski način povezivanja LCD zaslona



Slika 51 8-pinski način povezivanja LCD zaslona

Postoji velika razlika između 8-bitnog i 4-bitnog načina rada osim broja podatkovnih pinova potrebnih za povezivanje LCD-a.

8-bitni način je znatno brži od 4-bitnog načina. To je zato što se u 8-bitnom načinu rada podaci zapisuju u jednoj operaciji, dok se u 4-bitnom načinu rada bajt dijeli na dva dijela i izvode se dvije operacije pisanja.

Stoga se 4-bitni način rada obično koristi za spremanje I/O pinova, dok je 8-bitni način najprikladniji kada je brzina prioritet u aplikaciji i kada je dostupno najmanje 10 I/O pinova.

Ako koristimo, LCD zaslon u 4-bitnom načinu rada potrebno je samo šest pinova: RS, EN, D7, D6, D5 i D4. U tom slučaju spojite četiri podatkovna pina LCD-a (D4-D7) na digitalne pinove 5 do 2 na Arduinu, EN pin na digitalni pin 11, a RS pin na digitalni pin 12.

Korisne funkcije u biblioteci LiquidCrystal.h

lcd.home() - postavlja kursor u gornji lijevi kut LCD-a bez brisanja zaslona.

lcd.blink() - prikazuje trepćući blok od 5×8 piksela na mjestu na kojem će biti upisan sljedeći znak.

lcd.noBlink() -isključuje treptanje LCD kursora.

lcd.cursor() - prikazuje podvlaku (crtu) na mjestu na kojem će biti upisan sljedeći znak.

lcd.noCursor() - funkcija skriva LCD pokazivač.

lcd.scrollDisplayRight() - pomiče sadržaj zaslona jedno mjesto udesno. Ako želite da se tekst neprekidno pomiče, morate koristiti ovu funkciju unutar for petlje.

lcd.cursor() - prikazuje podvlaku (crtu) na mjestu na kojem će biti upisan sljedeći znak.

lcd.noCursor() - funkcija skriva LCD pokazivač.

lcd.scrollDisplayRight() - pomiče sadržaj zaslona jedno mjesto udesno. Ako želite da se tekst neprekidno pomiče, morate koristiti ovu funkciju unutar for petlje.

CGROM i CGRAM

Svi LCD-ovi s upravljačkim programom Hitachi HD44780 imaju dvije vrste memorije: CGROM i CGRAM (ROM i RAM za generator znakova).

CGROM je trajna memorija koja zadržava podatke čak i kada se isključi napajanje, dok je CGRAM nepostojana memorija koja gubi podatke kada se isključi napajanje.

CGROM pohranjuje font koji se pojavljuje na LCD zaslonu znakova. Kada date naredbu LCD-u sa znakovima da prikaže slovo 'A', on mora znati koje točke treba uključiti kako bismo vidjeli 'A'. Ovi se podaci pohranjuju u CGROM.

CGRAM je dodatna memorija za pohranu korisnički definiranih znakova. Ovaj RAM je ograničen na 64 bajta. Stoga, za LCD od 5 × 8 piksela, samo 8 korisnički definiranih znakova može se pohraniti u CGRAM, dok se za LCD od 5 × 10 piksela mogu pohraniti samo 4.

Ispisivanje posebnih znakova na 16 x 2 LCD zaslonu.

Ukoliko vam klasični zadani font nije dovoljan, možete izraditi vlastite prilagođene znakove (glifove) i simbole. Oni su korisni kada trebate prikazati znak koji nije u standardnom ASCII skupu znakova.

Kako se likovi na zaslonu sastoje od matrice 5×8 piksela; vaš novi znak koji želite izraditi morate definirati unutar ove 5×8 matrice. Za to koristimo funkciju **createChar()**.

Da biste koristili **createChar(),** prvo morate stvoriti niz od 8 bajtova. Svaki bajt u nizu odgovara retku u matrici 5×8. U bajtu znamenke 0 i 1 označavaju koji pikseli u nizu trebaju biti UKLJUČENI, a koji ISKLJUČENI.

Svi ovi korisnički definirani znakovi pohranjeni su u CGRAM LCD-a.

Jedino ograničenje prilikom stvaranja posebnih znakova je to što biblioteka LiquidCrystal podržava samo osam prilagođenih znakova. To je ipak dovoljno za stvaranje određenog broja korisnih znakova.

U sljedećem primjeru koda možete vidjeti kako kreirati posebne znakove.

Kod:

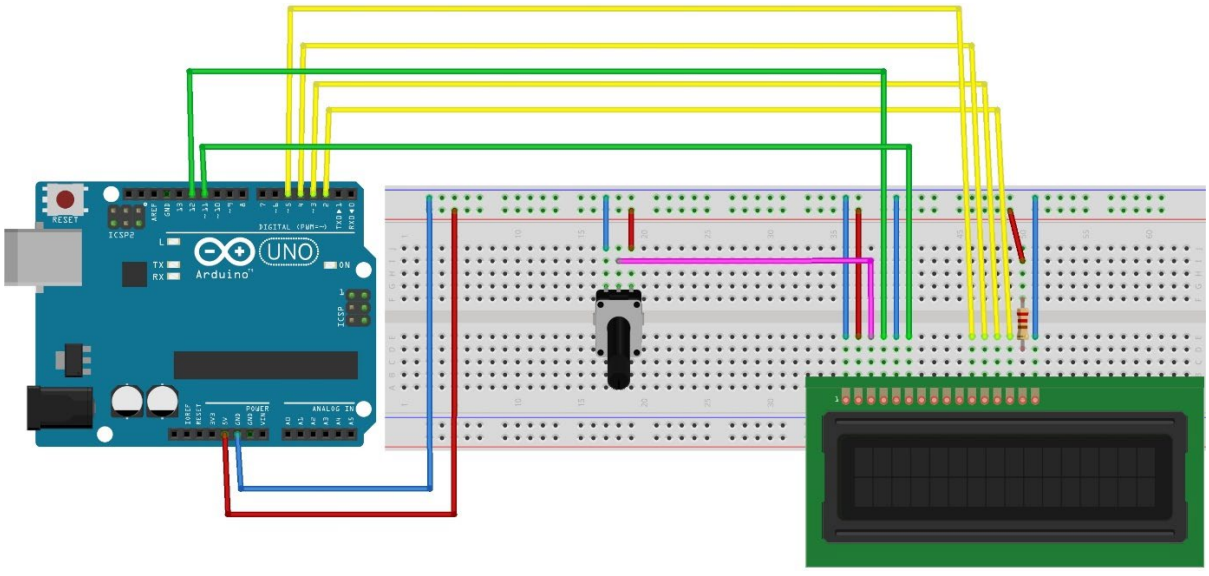
```
/*
LCD POSEBNI ZNAKOVI

  Program stvara i ispisuje četiri posebna znaka na 16 x 2 LCD zaslon.
  To su znakovi: srce, zvono, zvučnik, zaključano.

 Kreirao, Joško Smolčić, 02.11.2023.
 */


// uključivanje LiquidCrystal.h biblioteke:
#include <LiquidCrystal.h>

// dodjeljivanje pinova za komunikaciju sa LCD zaslonom.
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// kodovi za posebno stvorene znakove:
byte Srce[8] = {
0b00000,
0b01010,
0b11111,
0b11111,
0b01110,
0b00100,
0b00000,
0b00000
};
```

```
byte Zvono[8] = {
0b00100,
0b01110,
0b01110,
0b01110,
0b11111,
0b00000,
0b00100,
0b00000
};

byte Zvucnik[8] = {
0b00001,
0b00011,
0b01111,
0b01111,
0b01111,
0b00011,
0b00001,
0b00000
};

byte Zakljucano[8] = {
0b01110,
0b10001,
0b10001,
0b11111,
0b11011,
0b11011,
0b11111,
0b00000
};

void setup()
{
  // pokretanje LCD zaslona i namještanje broja stupaca i redaka:
  lcd.begin(16, 2);

  // stvaranje novog znaka
  lcd.createChar(0, Srce);


  lcd.createChar(1, Zvono);


  lcd.createChar(2, Zvucnik);
```

```
  lcd.createChar(3, Zakljucano);

  // isprazni LCD sučelje
  lcd.clear();

  // Ispiši poruku na LCD
  lcd.print("Poseban znak");
}
// ispisivanje svih znakova
void loop()
{
  lcd.setCursor(0, 1);
  lcd.write(byte(0));

  lcd.setCursor(2, 1);
  lcd.write(byte(1));

  lcd.setCursor(4, 1);
  lcd.write(byte(2));

  lcd.setCursor(6, 1);
  lcd.write(byte(3));

}
```

Zadatak 1

Koristeći zadanu shemu i kod izvrši spajanje LCD zaslona, učitavanje koda, te u kod unesi potrebne izmjene da bi se izvršilo ispisivanje: „Moj prvi ispis“ u prvoj liniji i tvoje ime i prezime u drugoj liniji. Nakon uspješnog ispisa zamjeni redoslijed ispisivanja linija te pomakni ispis prema sredini zaslona.



Slika 52 Shema spajanja LCD zaslona

Kod:

```
/*
LCD - Moj prvi ispis

  Program ispisuje "Moj prvi ispis"  na 16 x 2 LCD zaslon u gornjoj liniji,
   dok u donjoj liniji ispisuje ime i prezime.


 Kreirao, Joško Smolčić, 01.11.2023.
 */

// uključivanje LiquidCrystal.h biblioteke:
#include <LiquidCrystal.h>

// Stvaranje LCD objekta. Parametri: (rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  // pokretanje LCD zaslona i namještanje broja stupaca i redaka:
  lcd.begin(16, 2);

  // isprazni LCD sučelje
  lcd.clear();
}

void loop()
{
  // Ispiši poruku na LCD
  lcd.print(" Moj prvi ispis!");


  // namještanje pokazivača na stupac 0 linija 1
  // Napomena: linija 1 je drugi redak jer brojenje počinje sa 0.

  lcd.setCursor(0, 1);
  // Ispiši poruku na LCD
  lcd.print(" Ime i prezime");
}
```

Zadatak 2

Koristeći program „LCD POSEBNI ZNAKOVI“, napraviti njegovo proširivanje te programirati dodatno ispisivanje još tri znaka. Za dva znaka „Kvačica“ i „Nota“ koristiti kodove u tablici ispod, a treći znak kreirati samostalno po vlastitoj želji po već viđenom uzorku.

Rješenje:

```
byte Kvacica[8] = {
0b00000,
0b00001,
0b00011,
0b10110,
0b11100,
0b01000,
0b00000,
0b00000
};
```

```
byte Nota[8] = {
0b00001,
0b00011,
0b00101,
0b01001,
0b01001,
0b01011,
0b11011,
0b11000
};
```

5.5 VJEŽBA br. 19 LCD termostat sa 100k termistorom

Termistor 100k

U svojoj osnovi termistori su **promjenjivi otpornici** koji mijenjaju svoj otpor s promjenom temperature. To su prije svega jednostavne i jeftine komponente s kojima možemo na lak način očitati podatke o temperaturi za naše projekte kao što su udaljene meteorološke stanice, sustavi zaštite od pregrijavanja ili sustavi kućne automatizacije. Oni su analogni senzori, tako da je njihov kod relativno jednostavan u usporedbi s digitalnim temperaturnim senzorima koji zahtijevaju posebne biblioteke i puno koda.

Slika 53 NTC 100k termistori

Termistori se dijele prema načinu na koji njihova otpornost reagira na promjene temperature. U termistorima s negativnim temperaturnim koeficijentom **(NTC)**, otpor se smanjuje s porastom temperature. Temperaturno područje rada **NTC** termistora je od **-50** do **150** stupnjeva celzijusa. U termistorima s pozitivnim temperaturnim koeficijentom **(PTC)**, otpor raste s porastom temperature. Temperaturno područje rada **PTC** termistora je od **-50** do **220** stupnjeva celzijusa.

NTC termistori su najčešće korištena vrsta termistora i takav termistor ćemo koristiti i u ovoj vježbi. NTC termistori izrađeni su od poluvodičkog materijala (kao što je metalni oksid ili keramika) koji se zagrijava i komprimira kako bi se formirao temperaturno osjetljiv vodljivi materijal.

Vodljivi materijal sadrži nosače naboja koji omogućuju protok struje kroz njega (elektroni ili šupljine). Visoke temperature uzrokuju da poluvodički materijal oslobađa više nositelja naboja.

U NTC termistorima izrađenim od željeznog oksida, elektroni su nositelji naboja, dok su kod nikal-oksidnih NTC termistora, nositelji naboja šupljine.

Izračunavanje i pretvorba otpora u temperaturu kod 100k termistora

S obzirom da je termistor promjenjivi otpornik, prvo moramo izmjeriti vrijednost otpora kako bi smo iz njega mogli izračunati temperaturu. Međutim, Arduino ne može izravno mjeriti otpor, može mjeriti samo napon. Arduino će mjeriti napon u točki između termistora i poznatog otpornika. Ovo je poznato kao razdjelnik napona.

Jednadžba za razdjelnik napona je:

$$\boldsymbol{U_{aut} = U_{in} * (\frac{R2}{R1 + R2})}$$

gdje je:

$\mathbf{U_{aut}}$ = napon između termistora i poznatog otpora

$\mathbf{U_{in}}$ = napon razina Arduina, uglavnom Vcc = 5V

R1 = otpor poznatog otpornika

R2 = otpor termistora

Ono što mi trebamo je saznati trenutni otpor termistora. Stoga je potrebno urediti jednadžbu na sljedeći način:

$$\boldsymbol{R2 = R1 * (\frac{U_{in}}{U_{aut}} - 1)}$$

Nakon što saznamo trenutnu vrijednost termistora za pretvorbu otpora u temperaturu koristimo **Steinhart-Hartovu** jednadžbu kako bi smo izračunali trenutnu temperaturu.

Steinhart-Hartova jednadžba je model otpora poluvodiča pri različitim temperaturama. Jednadžba je:

$$\boldsymbol{\frac{1}{T} = A + B_{ln} + C(\ln R)^3}$$

gdje je:

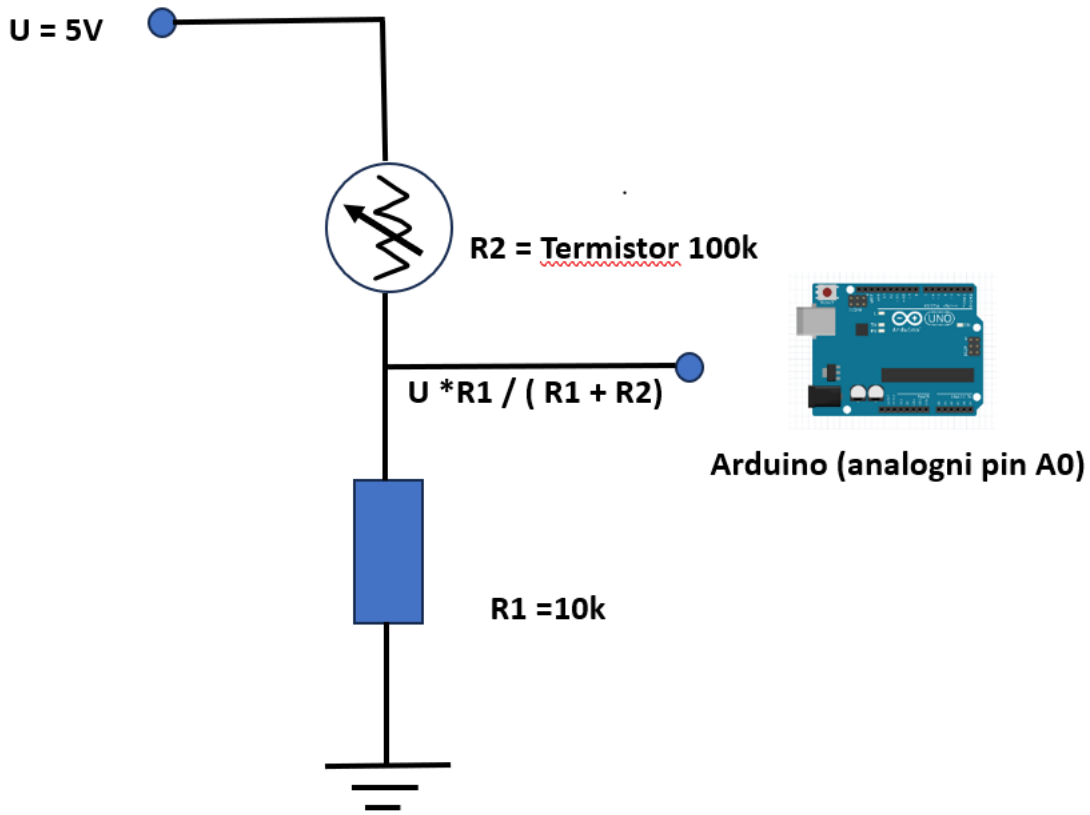
T je temperatura (u kelvinima),

R je otpor na T (u ohmima),

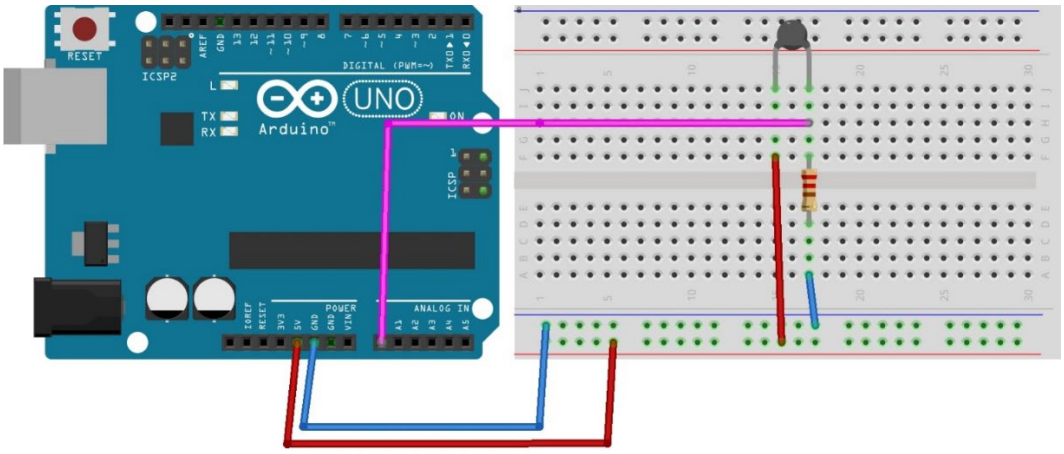
A, B i **C** su Steinhart-Hartovi koeficijenti, koji variraju ovisno o vrsti i modelu termistora i temperaturnom rasponu.

Spajanje 100k termistora u strujni krug

Arduino mjeri napon u točki između termistora i poznatog otpornika i spaja se na način kako je prikazano na shemi ispod.



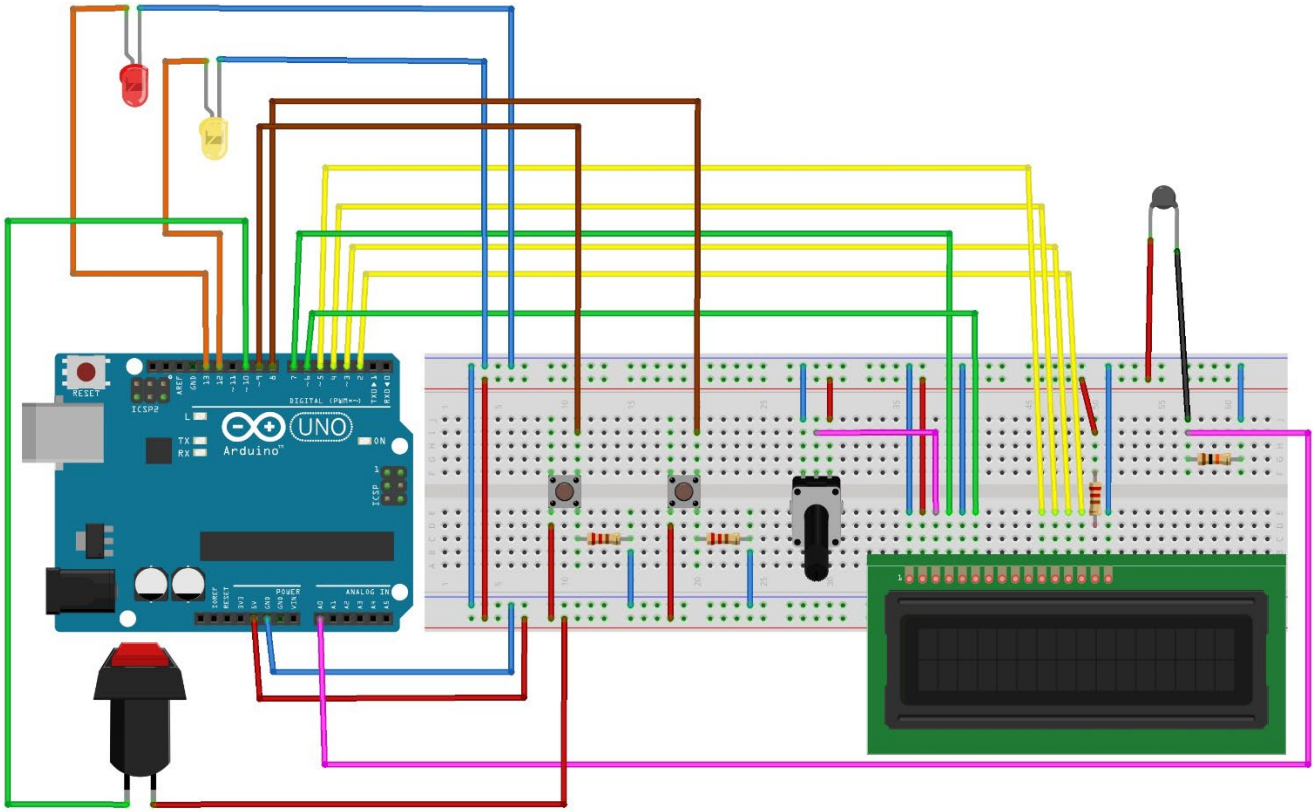
Slika 54 Shema spajanja termistora



Slika 55 Način spajanja termistora sa Arduinom

Zadatak 1.

Prema zadanoj shemi spojiti model LCD termostata te učitati program. Nakon pokretanja modela termostata testirati ispravnost rada te izvršiti eventualnu kalibraciju termostata.



Slika 56 Shema spajanja

Kod:

```
/*
LCD_termostat

  Ovaj program je osmišljen u svrhu rada termostata koji upravlja sa
uređajem za zagrijavanje ili hlađenje.
  Termostat je osmišljen na način da ima dvije tipke s kojima korisnik može
zadati zadanu temperaturu.
  Jedna tipka je za odabir više a jedna tipka je za odabir niže temperature.
  Trenutna temperatura  se uspoređuje sa zadanom te se ovisno o rezultatu
  ako je zadana temperaturea veća na pinovima 12 i 13 dobiva signal i pali
LED dioda.
  Ti se signali koriste za aktiviranje releja koji aktiviraju grijalicu.
  Također je predviđena i treća tipka koja služi za paljenje i gašenje
osvjetljenja na LCD ekranu.
  Za očitanje temperature se koristi termistor od 100k ohma i serijski
otpornik od 10k ohma.

  Program napisao: Joško Smolčić, 11/2015.
 */


#include <math.h>
#include <EEPROM.h> // učitavanje EEPROM biblioteke
#include <LiquidCrystal.h>// učitavanje LCD biblioteke
LiquidCrystal lcd(7,6,5,4,3,2); // // pinovi za spajanje LCD zaslona
7,6,5,4,3,2

float settemp; // varijabla za namještaje temperature
int paljenje =13; // LED na pinu 13
int paljenje_fan =12; // LED na pinu 12
int temp_vise = 9;  // temperatura više na pinu 9
int temp_nize = 8;   // temperatura niže na pinu 8
int signal_svjetla = 10;  //tipka za paljenje svjetla
int stanje_svjetla = 11;  // stanje svjetla
int onoff = 0; // 0 = off, 1 = on


///////_____________ IZRAČUN I PRETVORBA
TEMPERATURE_________________////////////

double Thermistor(int ADpretvorba) {
 double Temp;
 Temp = log(10000.0*((1024.0/ADpretvorba-1)));
 Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))*
Temp );
 Temp = Temp - 273.15;
```

```
 //Temp = (Temp * 9.0)/ 5.0 + 32.0;
 Temp = Temp - 2.5; //kalibracija
 return Temp;
}

///////_____________ PALJENJE ILI GAŠENJE OSVJETLJENJA LCD
ZASLONA_____________////////////

void doSomething()
{
  if (onoff==0)
  {
    onoff=1;
    digitalWrite(11, HIGH);

    Serial.println("LCD svijetli");
  }
  else
    if (onoff==1)
    {
      onoff=0;
      digitalWrite(11, LOW);

      Serial.println("LCD ugašen");
    }

 }


void setup() {
 Serial.begin(9600);

  lcd.begin(16, 2); // pokretanje LCD zaslona i namještanje broja stupaca i
redaka:
  lcd.setCursor(0,0); // namještanje pokazivača na stupac 0, linija 0
  lcd.print("Pozdrav!"); // prilikom pokretanja ispisuje Pozdrav u trajanju
od 1 sekunde.
  lcd.clear(); // isprazni LCD sučelje
  EEPROM.read (1); // adresa: lokacija s koje se čita

  pinMode(paljenje, OUTPUT);
  pinMode(paljenje_fan, OUTPUT);
  pinMode(signal_svjetla, INPUT);
  pinMode(stanje_svjetla, OUTPUT);

}

void loop() {
  int val;
```

```
  double temp;
  val=analogRead(0);
  temp=Thermistor(val);
  Serial.print("Temperature = ");
  Serial.print(temp);
  Serial.println(" C");
  delay(100);
//---------------------------------------

  lcd.setCursor (0,0); // namještanje pokazivača na stupac 0, linija 0
  lcd.print ("Temp.   ");
  lcd.print (temp);  // Ispis trenutne temperature u C
  lcd.print ('C');

//-------------------------------------------------------

settemp = EEPROM.read(1); // čitanje namještene temperature iz eEPROM-a

  delay (250); // kašnjenje zbog stabilnosti programa


  ///////////_______ If petlja za očitavanje vrijednosti sa pina za
podizanje zadane temperature._______ //////////

  if
    (digitalRead(temp_vise)== HIGH )
  {
    settemp ++ ;  // za svako očitanje sa zadanog pina (pritisak tipke)
dodaje za jedan broj
  }

else{ // u suprotnom (ako nema očitanja sa pina) ne radi ništa
}

////////////_____ If petlja za očitavanje vrijednosti sa pina za smanjenje
zadane temperature.___ /////

if
(digitalRead (temp_nize) == HIGH)// if we detect a 1 on the other switch pin
{
  (settemp --);// za svako očitanje sa zadanog pina (pritisak tipke)
smanjuje za jedan broj zadanu temperaturu

}
else {
// u suprotnom (ako nema očitanja sa pina) ne radi ništa
}
```

```
if (temp < settemp) // ako je stvarna temperatura manja od zadane
temperature
{
  digitalWrite (paljenje, HIGH); // upali LED diodu
}
else // u suprotnom ugasi LED diodu
{
  digitalWrite (paljenje,LOW);  //  ugasi LED diodu
}


//---------------------------------------------------------------

lcd.setCursor (0,1); // namještanje pokazivača na stupac 0, linija 1
lcd.print ("Zadano  "); // Ispis Zadano prije ispisa brojke zadane
temperature
lcd.print (settemp);  // ispis temperature
lcd.print ('C');  // ispis znaka C (Celzijus)
Serial.println(settemp);  // Ispis namještne temperature na serijskom
monitoru

EEPROM.write (1,settemp); /* zapišite najnoviju zadanu temperaturu u eeprom
pohranu podataka
iz razloga ako se isključi napajanje, vaša zadana temperatura bude
spremljena!*/


if(digitalRead(signal_svjetla) == HIGH)
{
  doSomething();
}
else
{// u suprotnom ne radi ništa
}
delay (250); // kašnjenje 250 milisekundi
}
```

5.6 VJEŽBA br. 20 Geeetech Arduino GPRS shield.

Slika 57 Geeetech Arduino GPRS shield.

Arduino GPRS shield se zasniva na SIM900 modulu od SIMCOM proizvođača i kompatibilan je sa Arduino pločicom i njezinim klonovima. GPRS shield omogućava komunikaciju koristeći GSM telefonsku mrežu, te omogućava: SMS, MMS, GPRS i Audio koristeći UART te AT komande (GSM 07.07, 07.05) te također SIMCOM poboljšane AT komande. Shield je opremljen sa 12 GPIO pinova, 2PWM pina, te ADC od SIM900 modula.

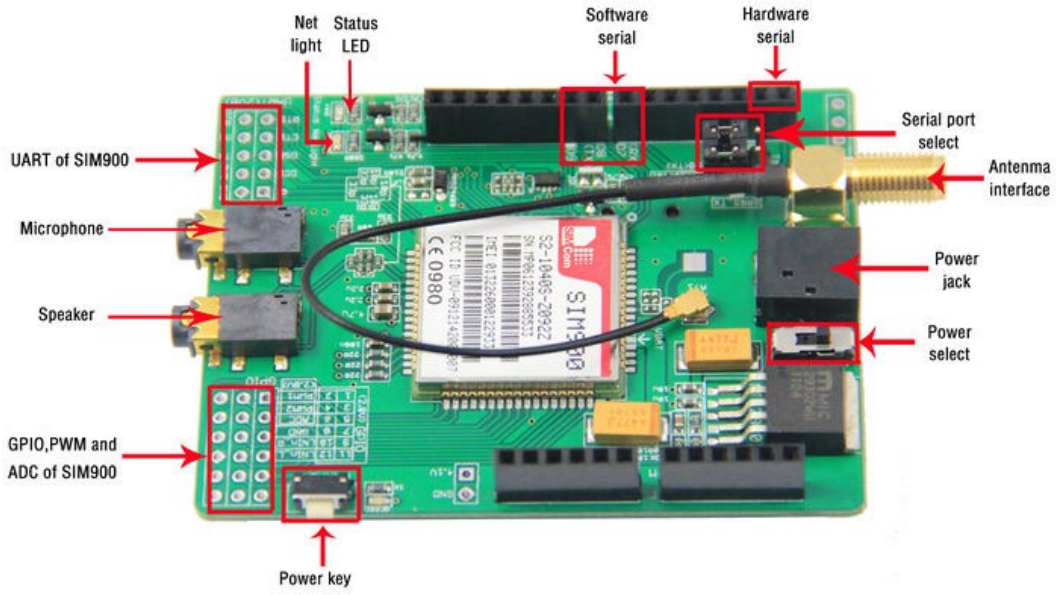
The GPRS Shield is based on SIM900 module from SIMCOM and compatible with Arduino and its clones. The GPRS Shield provides you a way to communicate using the GSM cell phone network. The shield allows you to achieve SMS, MMS, GPRS and Audio via UART by sending AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands). The shield also has the 12 GPIOs, 2 PWMs and an ADC of the SIM900 module(They are all 2V8 logic) present onboard.

Osnovne značajke ovog shielda su sljedeće:

- Frekventni pojas 850/900/1800/1900 MHz- pokriva u radu GSM mreže u svim zemljama na svijetu.
- GPRS multi-slot klasa 10/8
- GPRS mobilna stanica klase B
- Sukladno sa GSM faza 2/2+
- Klasa 4 (2W@850/900MHz)
- Klasa 1(1 W @ 1800 / 1900MHz)
- Upravljanje pomoću AT naredbi- Standardne naredbe: GSM 07.07 & 07.05 i poboljšane naredbe: SIMCOM AAT naredbe.

- Short Message Service – mogućnost slanja male količine podataka preko mreže (ASCII ili hexadecimalno).
- Ugrađen TCP/UDP - mogućnost učitavanja podataka na web server
- RTC podržan.
- Mogućnost izbora serijskog porta.
- Utičnica za mikrofon I slušalice
- Mala potrošnja energije - 1.5mA(sleep mode)
- Industrijski temperaturni opseg - -40°C to +85 °C

Pregled hardvera GPRS pločice i zauzetost pinova Arduino UNO pločice



Slika 58 Raspored pinova za Geeetech Arduino GPRS shield.

Hardver	Objašnjenje
Power select	izbor napajanja GPRS shielda (eksterno ili 5v preko Arduina)
Power jack	priključak za eksterno 4.8~5VDC napajanje
Antenna interface	priključak za antenu
Serial port select	izbor za software serial port ili hardware serial port način na koji će biti spojen na GPRS Shield
Hardware Serial	D0/D1 na Arduinu
Software serial	D7/D8 na Arduinu
Status LED	pokazuje dali je SIM900 uključen
Net light	pokazuje dali je SIM900 povezan sa mrežom
UART of SIM900	UART pinovi zaizlaz iz SIM900
Microphone	za odgovor na poziv

Speaker	za odgovor na poziv
GPIO,PWM and ADC of SIM900	GPIO,PWM i ADC izlazni pinovi od SIM900
Power key	uključivanje I isključivanje napona za SIM900
Power select	izbor napajanja GPRS shielda (eksterno ili 5v preko Arduina)

Zauzetost pinova na Arduino UNO pločici nakon spajanja GPRS shielda:

- **D0** - Slobodan ako ste izabrali software serial port za komunikaciju sa GPRS Shieldom
- **D1** - Slobodan ako ste izabrali software serial port za komunikaciju sa GPRS Shieldom
- **D2** - Slobodan
- **D3** - Slobodan
- **D4** - Slobodan
- **D5** - Slobodan
- **D6** - Slobodan
- **D7** - Zauzet ako ste izabrali software serial port za komunikaciju sa GPRS Shieldom
- **D8** - Zauzet ako ste izabrali software serial port za komunikaciju sa GPRS Shieldom
- **D9** - Zauzet za softversko paljenje ili gašenje SIM900
- **D10** - Slobodan
- **D11** - Slobodan
- **D12** - Slobodan
- **D13** - Slobodan
- **D14**(A0) - Slobodan
- **D15**(A1) - Slobodan
- **D16**(A2) - Slobodan
- **D17**(A3) - Slobodan
- **D18**(A4) - Slobodan
- **D19**(A5) - Slobodan
- **Napomena:** A4 i A5 su spojeni na I2C pinove na SIM900. Zbog toga SIM900 ne može biti dostupan preko I2C .

Hardverske i softverske postavke prije upotrebe Arduino GSM shilda

Prije montaže Arduino GSM shielda na, Arduino UNO pločicu potrebno je napraviti određena namještanja:

- Potrebno je izabrati komunikacijski port. To se radi hardverski premještajem jumpera na samoj pločici čime izabiremo **"Softver serial"** komunikaciju ili **"Hardver serial"** komunikaciju. To je vidljivo na slikama ispod. U ovom slučaju biramo Softver serial komunikaciju što znači da će naši pinovi za komunikaciju sa Arduino UNO pločicom biti pinovi D7 I D8.

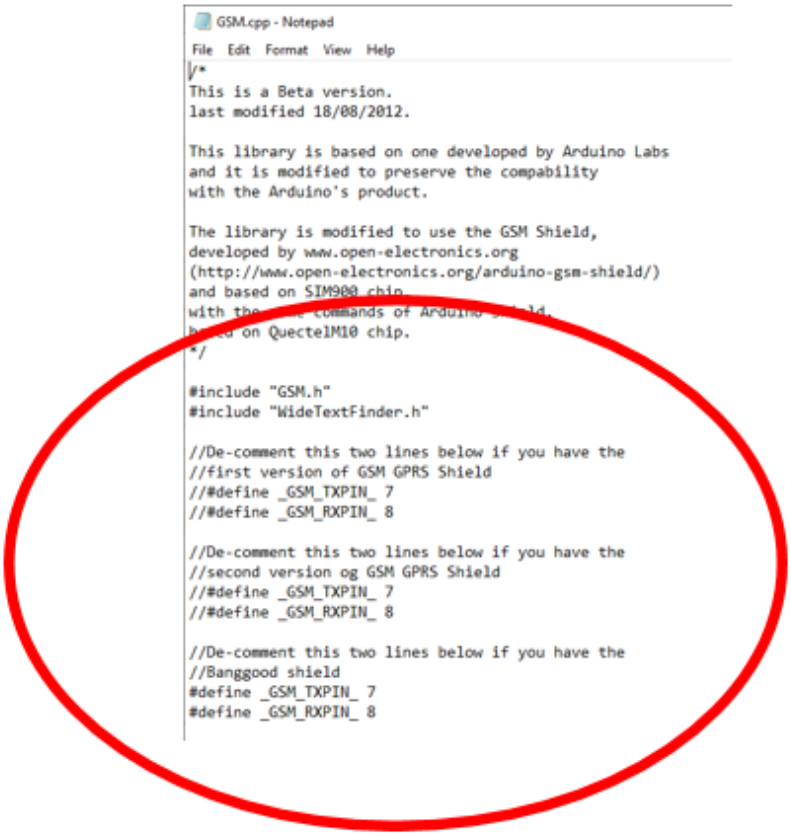
Slika 59 Izbor “Softver serial” ili “Hardver serial” komunikacije

Slika 60 Spajanje jumpera za izbor “Softver serial” ili “Hardver serial” komunikacije

- Slijedeći korak se odnosi na mogućnost Arduino GSM shilda da se softverski uključuje odnosno isključuje. Za to će biti potrebno izvršiti prespajanje **„JP“** kontakata na samoj pločici, na način kao na slici ispod.

Slika 61 „JP“ kontakt

U sljedećem koraku nam još preostaje izvršiti određene promjene u samoj Biblioteci (Library)
Potrebno je u klasi GSM folderu GSM.cpp, napraviti upis pinova za komunikaciju na slijedeći način:

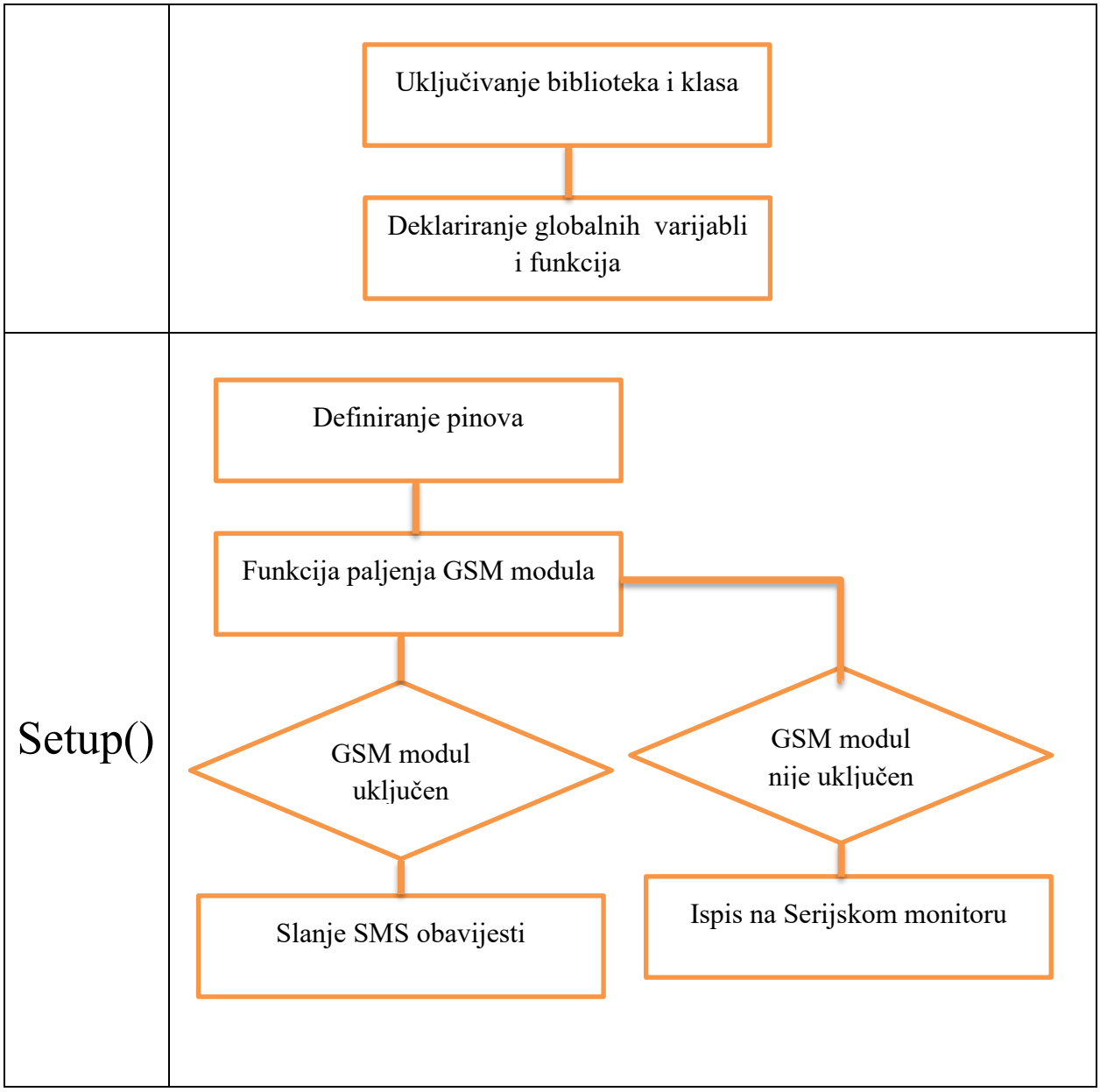


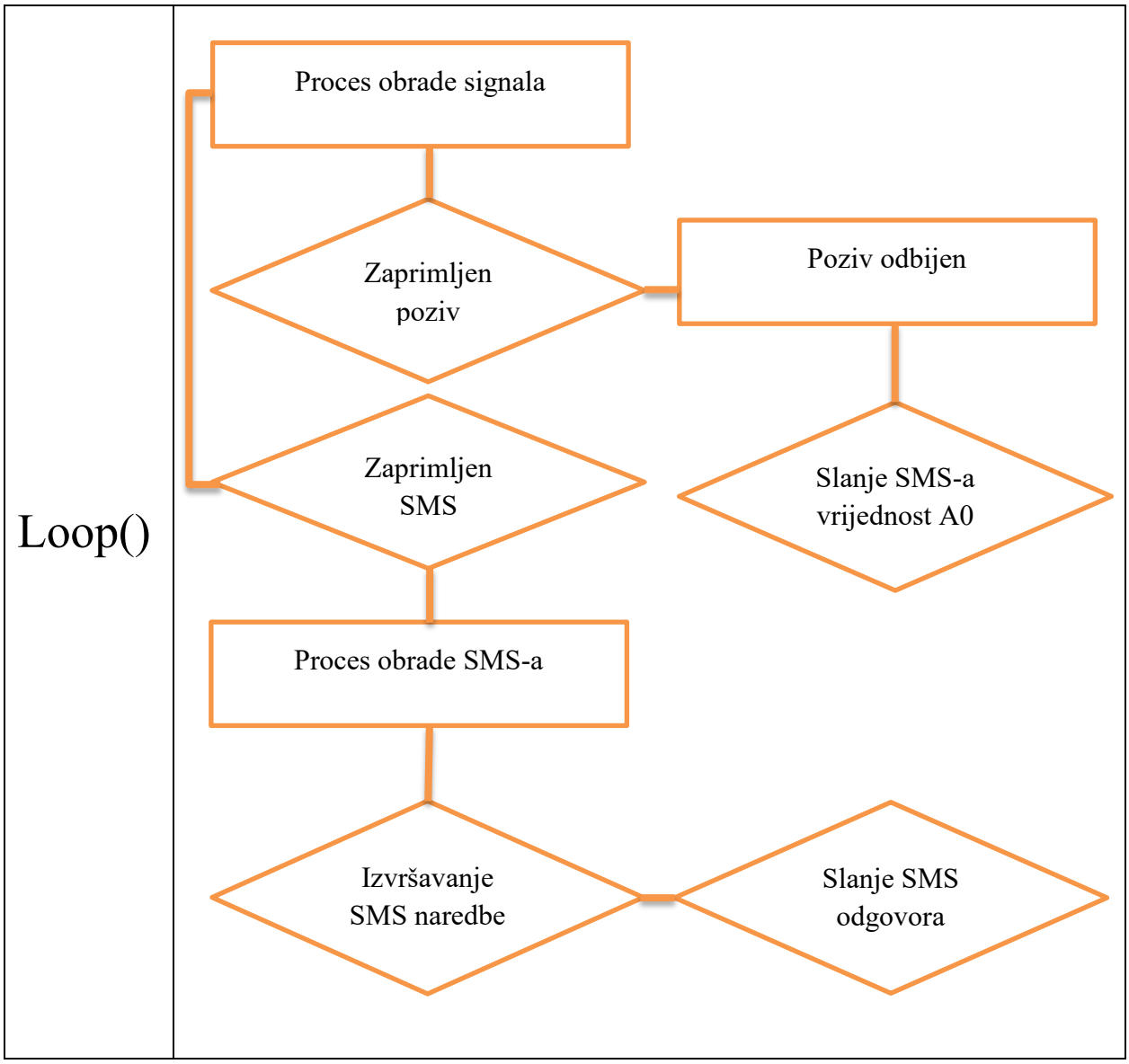
Slika 62 Upis pinova za komunikaciju

Softver

Za stvaranje ovog koda je korištena i već gotova biblioteka (Libraries). Tvorac biblioteke je Marco Martines. https://github.com/MarcoMartines/GSM-GPRS-GPS-Shield

Programski dijagram




Loop()
Proces obrade signala
Zaprimljen poziv
Poziv odbijen
Zaprimljen SMS
Slanje SMS-a vrijednost A0
Proces obrade SMS-a
Izvršavanje SMS naredbe
Slanje SMS odgovora

Priručnik za korištenje programa

Upravljanje GSM modulom vrši se isključivo preko internet mreže. U tu svrhu se koriste određene SMS naredbe kojima je omogućeno paljenje i gašenje dva izlaza korištenjem SMS poruka.
Također postoji mogućnost administriranja i autorizacije korištenjem SMS poruka.
Administrator je osoba koja je upisana na prvom mjestu u SIM kartici.
Nakon što se jedan Administrator upiše, ne može se upisati ni jedan drugi dok se ovaj sam ne izbriše, brisanjem pojedinačnog broja ili kompletnog SIM imenika.
Ukoliko se izvrši pojedinačno brisanje, mogućnost dodavanja novog administratora imaju svi oni brojevi koji su već zapisani u SIM imenik. Sljedeći broj koji se pošalje za upis će biti upisan na prvo slobodno mjesto te će on postati administrator.
U slučaju brisanja kompletnog SIM imenika kao administrator se može postaviti bilo koji broj koji pošalje SMS sa riječi „ADMIN“ (obavezno veliki slovima), taj broj postaje administrator te on dodaje ostale brojeve.
Program omogućuje upis do 10 brojeva u imenik na SIM karticu i svi imaju korisnička prava.
Broj upisa je moguće proširiti na broj upisa koji podržava odabrana SIM kartica u svom imeniku, ali samo promjenom vrijednosti u kodu.

Korisničke naredbe.

"ON 1" : pali prvi izlaz na pinu broj 12, OUT 1 = ON (PIN 12)
"OFF 1" : gasi prvi izlaz na pinu 12, OUT 1 = OFF (PIN 12)
"ON 2" : pali drugi izlaz na pinu broj 13, OUT 2 = ON (PIN 13)
"OFF 2" : gasi drugi izlaz na pinu broj 13, OUT 2 = OFF (PIN 13)
"STATUS" : vraća SMS sa stanjem oba izlaza.
"INFO" Zahtjeva od SIM900 modula sljedeće informacije (operator, jakost signala , IMEI)
"ACK ON" : Omogućuje povratni SMS kao potvrdu paljenja ili gašenja jednog od izlaza
"ACK OFF" : Onemogućava povratni SMS kao potvrdu paljenja ili gašenja jednog od izlaza (default)
"PRI ON" Omogućuje privatnost (default) (ne pokazuje cijeli broj pozivatelja u Serial Monitoru)
"PRI OFF" Onemogućuje privatnost

Administratorske naredbe

"ADMIN" : Nakon slanja ove poruke broj koji je pošalje je automatski upisan na mjesto broj 1 u SIM imenik na SIM kartici, pod uvjetom da je mjesto prazno, odnosno da već ranije nije netko upisan.
"SIM POSITION-X" : Slanjem ove poruke te izborom broja umjesto X između 1-10 dobivate povratni SMS sa brojem koji je upisan na toj poziciji.

"SIM DELETE-X" : Slanjem ove poruke te izborom broja umjesto X između 1-10 brišete pojedinačni broj sa SIM kartice.
"SIM WRITE-+38591xxxxxx:Korisnik" : Slanjem ove poruke upisati će te napisani broj u prvo slobodno mjesto na SIM kartici. Broj mora biti u sljedećem formatu:+385- broj države, 91-broj davaoca usluge, xxxxxxx- osobni broj sa 6 ili 7 znamenk.
Sve skupa: +38591xxxxxxx. Nakon broja stavljamo znak dvotočke te nakon njega upisujemo ime vlasnika broja.
"SIM BOOK DELETE" : Slanjem ove SMS poruke će biti izbrisan kompletan imenik na SIM kartici.
"SEND SMS-+38591xxxxxx:Ovo je poruka" : Slanjem ovakve poruke GSM modul će proslijediti takvu SMS poruku na broj u poruci. Ova naredba se koristi za provjeru stanja na računu od strane administratora
"CALL-+38591xxxxxx" : Slanjem ovakve SMS naredbe GSM modul će izvršiti pozivanje broja u SMS poruci. Ova naredba također može služiti za provjeru stanja računa.
"POWER OFF" : Ova naredba gasi GSM modul. Ovo se može koristiti u sigurnosne svrhe u slučaju da administrator želi naglo prekinuti rad GSM modula.
Nakon gašenja modula, ne postoji način da se modul upali preko mreže na daljinu. Potrebno je fizičko resetiranje modula, (isključivanje te ponovno uključivanje napajanja) da bi se modul ponovno pokrenuo.
Napomena: izbjegavati uzastopno slanje ovo poruke. Nakon prvog slanja GSM modul se gasi te eventualno drugo slanje neće biti zaprimljeno. Nakon fizičkog resetiranja modula, druga poslana poruka će biti zaprimljena te će se modul opet ugasiti.

Napomene:

- Postoji i mogućnost poziva GSM modula nakon čega je poziv odbijen a kao potvrda SMS porukom stiže vrijednost analognog pina A0.
- Sve SMS poruke koje nisu autorizirane, odnosno ne dolaze od jednog od upisanih brojeva na SIM kartici, će biti proslijeđene administratoru.
- Administratorom se smatra broj koji je na SIM kartici upisan na poziciju broj 1.
- Sve naredbe se pišu bez navodnika.

Kod

```
// TAB - 1
/*////////////////////////////////////////////////////////////////////////
////////////////////////////////////
  * File:             Arduino_GSM_shield_modul
  * Autor:            Joško Smolčić
  * Verzija:          1.1
  * Datum izrade:     10.01.2020
  *
  *
  * ____________________________________________________
  * DETALJI O PROGRAMU:
  * ____________________________________________________
  *
```

```
  *  Arduino_GSM_shield_modul, je program koji služi u svrhu upravljanja
Arduino Uno mikrokontrolerskom
  *  pločicom uz korištenje "GSM Shield" modula, koji je opremljen sa
SIM900 procesorom.
  *  U radu je korištena ova GSM shield pločica:
  *  http://wiki.seeedstudio.com/GPRS_Shield_v1.0/#a-simple-source-code-
examples
  *
  * Program koristi bibliotekom čiji je tvorac Marco Martines:
  * https://github.com/MarcoMartines/GSM-GPRS-GPS-Shield
  * GSM Shield for Arduino: www.open-electronics.org
  *
  * Arduino_GSM_shield_modul program se bazira na programu od: Guido
Ottaviani-->guido@guiott.com<--, GSM_control,
  * koji također koristi neznatno promijenjenu biblioteku od Marka
Martineza.
  *
  * Arduino_GSM_shield_modul  je slobodan softver i može kopirati ili
nadograđivati i koristiti
  * od strane bilo koga. Svrha programa je da koristi svima koji trebaju
ovakvu vrstu programa i ne daje
  * nikakvo jamstvo za upotrebu. SVAKO KORIŠTENJE JE NA VLASTITU
ODGOVORNOST.
  *
  *
  * ____________________________________________________
  * FUNKCIJE PROGRAMA:
  * ____________________________________________________
  *
  * Ovim programom je omogućeno paljenje i gašenje dva izlaza korištenjem
SMS poruka.
  * Također postoji mogućnost administriranja i autorizacije  korištenjem
SMS poruka.
  * Administrator je osoba koja je upisana na prvom mjestu u SIM kartici.
  * Nakon što se jedan Administrator upiše ne može se upisati ni jedan
drugi dok se ovaj sam ne izbriše.
  * Program omogućuje upis do 10 brojeva u imenik na SIM karticu i svi
imaju korisnička prava.
  * Broj upisa je moguće  proširiti na broj upisa koji podržava odabrana
SIM kartica u svom imeniku.
  *
  *
  * KORISNIČKE KOMANDE U PROGRAMU:
  * ------------------------------
  *
  * "ON 1" : pali prvi izlaz na pinu broj 12,   OUT 1 = ON  (PIN 12)
    "OFF 1" : gasi prvi izlaz na pinu 12,       OUT 1 = OFF (PIN 12)
    "ON 2" : pali drugi izlaz na pinu broj 13,   OUT 2 = ON  (PIN 13)
    "OFF 2" : gasi drugi izlaz na pinu broj 13,  OUT 2 = OFF (PIN 13)
```

```
    "STATUS" : vraća SMS sa stanjem oba izlaza.
  *
  * "INFO" Zahtjeva od  SIM900 modula sljedeće informacije (operator,
jakost signala , IMEI)
    "ACK ON" : Omogućuje povratni SMS kao potvrdu paljenja ili gašenja
jednog od izlaza
    "ACK OFF" : Onemogućava povratni SMS kao potvrdu paljenja ili gašenja
jednog od izlaza (default)
    "PRI ON" Omogućuje privatnost (default) (ne pokazuje cijeli broj
pozivatelja u Serial Monitoru)
    "PRI OFF" Onemogućuje privatnost
    "POWER OFF" : Ova naredba gasi GSM modul. Ovo se može koristiti u
sigurnosne svrhe u slučaju da administrator želi naglo prekinuti rad GSM
modula.
  *               Nakon gašenja modula, ne postoji način da se modul
upali preko mreže na daljinu.
  *               Potrebno je fizičko resetiranje modula, (isključivanje
te ponovno uključivanje napajanja) da bi se modul ponovno pokrenuo.
  * ADMINISTRATORSKE KOMANDE U PROGRAMU:
  * ------------------------------
  *
  * "ADMIN" : Nakon slanja ove poruke broj koji je pošalje je automatski
upisan na mjesto broj 1 u SIM imenik na SIM kartici.
  *           Pod uvjetom da je mjesto prazno, odnosno da već ranije nije
netko upisan.
  * "SIM POSITION-X" : Slanjem ove poruke te izborom broja umjesto X
između 1-10 dobivate povratni SMS sa brojem koji je upisan na toj
poziciji.
  * "SIM DELETE-X" : Slanjem ove poruke te izborom broja umjesto X između
1-10 brišete pojedinačni broj sa SIM kartice.
  * "SIM WRITE-+38591xxxxxx:Korisnik" : Slanjem ove poruke upisati će te
napisani broj u prvo slobodno mjesto na SIM kartici.
  *                                     Broj mora biti u sljedećem
formatu:+385- broj države, 91-broj davaoca usluge, xxxxxxx- osobni broj
sa 6 ili 7 znamenki.
  *                                     Sve skupa: +38591xxxxxxx. Nakon
broja stavljamo znak dvotočke te nakon njega upisujemo ime vlasnika
broja.
  *"SIM BOOK DELETE" : Slanjem ove SMS poruke će biti izbrisan kompletan
imenik na SIM kartici.
  *"SEND SMS-+38591xxxxxx:Ovo je poruka" : Slanjem ovakve poruke GSM
modul će proslijediti takvu SMS poruku na broj u poruci.
  *                       Ova naredba se koristi za provjeru stanja na
računu od strane administratora
  *"CALL-+38591xxxxxx" : Slanjem ovakve SMS naredbe GSM modul će izvršiti
pozivanje broja u SMS poruci. Ova naredba također može služiti za
provjeru stanja računa.
```

```
  *Postoji i mogućnost poziva GSM modula nakon čega je poziv odbijen a
kao potvrda SMS porukom stiže vrijednost analognog
pina  A0.
  *
  *Sve SMS poruke koje nisu autorizirane, odnosno ne dolaze od jednog od
upisanih brojeva na SIM kartici, će biti proslijeđene administratoru.
  *Administratorom se smatra broj koji je na SIM kartici upisan na
poziciju broj 1.
  *NAPOMENA: Sve naredbe se pišu bez navodnika.

 *//////////////////////////////////////////////////////////////////////
//////


#include "SIM900.h"
#include <SoftwareSerial.h>
//We don't need the http functions. So we can disable the next line.
//#include "inetGSM.h"
#include "sms.h"
#include "call.h"
//#include <EEPROM.h>

//To change pins for Software Serial, use the two lines in GSM.cpp.

//We have to create the classes for SMSs and calls.
CallGSM call;
SMSGSM sms;

byte stat=0;
int value=0;

//pin 7 = SW serial TX: _GSM_TXPIN_
//pin 8 = SW serial RX: _GSM_RXPIN_
//pin 9 = shield PwrKey: GSM_ON
//pin11 = Temporary output
//pin12 = Output 1
//pin13 = Output 2
//pinA0 = Analog In

const int TmpOut = 11;
const int Out1   = 12;
const int Out2   = 13;
const int AnalogIn = A0;

boolean started=false;

char phone_number[16]; // array for the phone number string
char TmpStr[159];
```

```
void powerUp();

void powerDown();

void ReadSms(void);

void PrivacyPrint(void);

void ToUpperTmpStr(void);

boolean SuperUserCmd(void);

boolean UserCmd(void);

char * getAtResult(char * DescString, char * AtString, char StartChar,
char StopChar);


void setup()
{
     pinMode(TmpOut,OUTPUT);
     pinMode(Out1,OUTPUT);
     pinMode(Out2,OUTPUT);
     powerUp();
     //Serial connection used for debug and status messages
     Serial.begin(19200);

     Serial.println(F("GSM Shield testing."));
     //Start configuration of shield with baudrate.
     //For http uses is raccomanded to use 4800 or slower.
     if(gsm.begin(19200))
     {
         Serial.println(F("\nstatus=READY"));
         started=true;
     }
     else
     {
         Serial.println(F("\nstatus=IDLE"));
     }

     if(started)
     {
       //Enable this two lines if you want to send an SMS.
       //E.g.: to know if the board restarts after a blackout
       if (sms.SendSMS("+385914672230", "Arduino GSM  modul je spreman za
rad."));
       //Serial.println("\nSMS sent OK");
       //sms.SendSMS(1, "Arduino GSM  modul je spreman za rad.");
//Ľˇalje obavijest na poziciju broj jedan na u SIM imeniku.
```

```
        Serial.println("\nSMS sent OK");
     }


}

void loop()
{

    if(started)
    {//Chekcs status of call
      stat=call.CallStatusWithAuth(phone_number,1,10);
      //If the incoming call is from an authorized number
      //saved on SIM in the positions range from 1 to 3.
      //writing 0, 0 means that every call is authorized.
      //it could be useful but check carefully your environment
      if((stat==CALL_NONE))
      {
        ReadSms();
      }
      else if((stat==CALL_INCOM_VOICE_AUTH))
      {//if the caller is authorized (both superuser or normal user)
       //the TmpOut is temporary set to HIGH and a SMS is sent back to
the caller
       //returning the analog value of AnalogIn port
          PrivacyPrint();
          Serial.println(F(""));
          digitalWrite(TmpOut, HIGH);
          //Hang up the call.
          call.HangUp();
          delay(2000);
          digitalWrite(TmpOut, LOW);
          //Check the value of the input.
          value=analogRead(AnalogIn);
          //Convert the int to a string.
          itoa(value,TmpStr,10);
          //Send an SMS to the caller with
          //the value read previously.
          Serial.print(F("Sending pin "));
          Serial.print(AnalogIn);
          Serial.print(F(" analog value: "));
          Serial.print(TmpStr);
          Serial.println(F(" to calling phone"));
          sms.SendSMS(phone_number,TmpStr);
       }
        else
       {
        PrivacyPrint();
        Serial.println(F(""));
```

```
        }
    }
}


 //TAB 2
boolean AckOn=false; // enable the call back phone call as an ACK  (ACK
OFF)
boolean Privacy=true;// printout of phone number masked at default (PRI
ON)
char position;

void ReadSms(void)
{//Read if there are messages on SIM card and decode the commnand
   position = sms.IsSMSPresent(SMS_ALL);
   if (position)
      {
        // read new SMS checking if the caller is authorized
        // if the number is into the SIM phone book this is a superuser
and can perform
        // also administrative functions
        if (GETSMS_AUTH_SMS == sms.GetAuthorizedSMS(position,
phone_number, TmpStr, 159, 1, 10))
        {
          Serial.println(F(""));
          Serial.print(F("Superuser call: "));
          PrivacyPrint();
          Serial.print(F("  Msg: "));
          Serial.println(TmpStr);

          ToUpperTmpStr();


          if(!SuperUserCmd()) // administrative actions
          {
            if(!UserCmd()) // and standard actions
            {// wrong command string
              Serial.println(F("Command not recognized"));
              sms.SendSMS(phone_number, "Naredba nije prepoznata");
              //PROBA SLIJEDEÄŚE NAREDBE

            }
          }
        }

 else
  {// the caller is not in SIM or in EEPROM phone books
        Serial.print(F("NOT granted call: "));
        Serial.print(phone_number);
```

```
        Serial.print(F("  Msg: "));
        Serial.println(TmpStr);
 sms.SendSMS(phone_number, "Ovaj broj nema administratorska prava");
 sms.SendSMS(1, TmpStr); // Šalje poruku koja dolazi od nepznatog
poĹˇiljaoca na broj koji je zapisan na poziciji 1 u SIM imeniku.
                              // u sluÄŤaju da davaoc usluge poĹˇalje
SMS sa obavijestima, SMS se proslјeÄ‘uje administratoru.


////////////////////////////////////////////////////////////  ADMIN
  ////////////////////////////

   if(!strcmp(TmpStr,"ADMIN"))
       {// Upiši broj na prvo mjesto u SIM karticu
           char prvi_dio_stringa [20] = "AT+CPBW=1,\"";
           char zadnji_dio_stringa [30] = "\",,\"Administrator\"";
                 strcat(prvi_dio_stringa, phone_number);
                 strcat(prvi_dio_stringa, zadnji_dio_stringa);
                 Serial.print(F("String AT nardbe glasi: "));
                 Serial.println(prvi_dio_stringa);
                 gsm.SimpleWriteln(prvi_dio_stringa);
                 delay(100);
                 Serial.print(TmpStr);
                 TmpStr[0]='\0';//resetiranje stringa
                 Serial.print(TmpStr);
                 strcat(TmpStr, (getAtResult("Ovaj broj je dobio
administratorska prava: ", "AT+CPBR=1", '"', '"')));
                 sms.SendSMS(phone_number, TmpStr);
             }

////////////////////////////////////////////////////////////////////////
////////////////////////////

      }
        // port status printout
        Serial.print(F("Out 1 = "));
        Serial.print(digitalRead(Out1));
        Serial.print(F(" - Out 2 = "));
        Serial.println(digitalRead(Out2));
        sms.DeleteSMS(position);
    }
     delay(1000);
}

boolean SuperUserCmd(void)
 {// administrative actions


  if(!strcmp(TmpStr,"ACK ON"))
```

```
  {// Enable Acknowledge call back
    AckOn = true;
  }
  else if(!strcmp(TmpStr,"ACK OFF"))
  {// Disable Acknowledge call back
    AckOn = false;
  }

  else if(!strcmp(TmpStr,"PRI ON"))
  {// Enable privacy
    Privacy = true;
  }
  else if(!strcmp(TmpStr,"PRI OFF"))
  {// Disable privacy
    Privacy = false;
  }
  else if(!strcmp(TmpStr,"INFO"))
  {// Request some info
       strcpy(TmpStr, "\0");

       //TmpStr[0]='\0';//resetiranje stringa

  strcat(TmpStr, (getAtResult("Operator: ", "AT+COPS?", '"', '"')));
         //delay(10);
  strcat(TmpStr, (getAtResult("\nSignal: ", "AT+CSQ", ':', ',')));
        // delay(10);
  strcat(TmpStr, (getAtResult("\nIMEI: ", "AT+GSN", '\n', 'O')));
        //delay(10);

       Serial.println(F("INFORMACIJE: "));
       Serial.println(TmpStr);
       sms.SendSMS(phone_number,TmpStr);
  }

//////////////////////////////////////////////////////  SIM POSITION
//////////////////////////
 else if(NULL != strstr(TmpStr,"SIM POSITION-"))
  {// Request a page from SIM phone book one by one position.

    char * PageC = strrchr(TmpStr, '-');
    char Page = atoi(PageC+1);

    int P = (int)Page;
    char c [10];
    sprintf(c, "%d",P); //pretvara int u string

    Serial.print(F("Page je= "));
    Serial.println(P);
    Serial.print(F( "A c je = "));
```

```
    Serial.println(c);

    char naredbaAT1 [40] = "AT+CPBR=";
    strcat(naredbaAT1, c);

    Serial.print(F("AT naredba je = "));
    Serial.println(naredbaAT1);

    strcat(TmpStr, (getAtResult("Broj: ", naredbaAT1, '"', '"')));

    sms.SendSMS(phone_number,TmpStr);

    sms.DeleteSMS(position);

  }


////////////////////////////////////////////////////////////////////////////
//////////////////////////

//////////////////SIM_DELETE  ///////////////////////////////////

  else if(NULL != strstr(TmpStr,"SIM DELETE-"))
  {// Request for delete number from SIM Phone Book, one by one.

    char * PageC = strrchr(TmpStr, '-');
    char Page = atoi(PageC+1);

    int P = (int)Page;
    char c [10];
    sprintf(c, "%d",P); //pretvara int u string
   /*
    Serial.print("Page je= ");
    Serial.println(c);
    Serial.print( "A c je = ");
    Serial.println(c);

    */
     char naredbaAT1 [40] = "AT+CPBW=";
     char naredbaAT2 [40] = "AT+CPBR=";
     strcat(naredbaAT1, c);
     strcat(naredbaAT2, c);

     //Serial.print("Naredba za brisanje glasi:  ");
     //Serial.println(naredbaAT1);
strcat(TmpStr, (getAtResult("Izbrisani broj je bio:  ", naredbaAT2, '"',
'"')));

     sms.SendSMS(phone_number,TmpStr);
```

```
     delay(100);

     strcat(TmpStr, (getAtResult(" ", naredbaAT1, '"', '"')));

     sms.DeleteSMS(position);
      //sms.SendSMS(phone_number,TmpStr);
  }

 ////////////////////////////////////////////////////////////////////////
////////////////////////////////

 ////////////////////////////////////////////////////  SIM WRITE
////////////////////////////////////

 //-------------------ÄŠITANJE BROJA POZIVATELJA I IMENA IZ SMS PORUKE---
------------------

  else if(NULL != strstr(TmpStr,"SIM WRITE-"))
  { // Write a single address to SMS phone book

       //Serial.print(F("1Puni TmpStr je = "));
       //Serial.println(TmpStr);

       char * broj_ime = strrchr(TmpStr, '-');
       broj_ime = broj_ime +1;

       //Serial.print(F("2Nakon _ TmpStr je = "));
       //Serial.println(broj_ime);

       char * Ime = strrchr(TmpStr, ':');
       Ime = Ime +1;

       //Serial.print(F("3Ime je nakon : = "));
       //Serial.println(broj_ime);

      char *i;
      char * broj = strtok_r(broj_ime,":",&i);

      //Serial.print(F("4Broj za upis je = "));
     //Serial.println(broj);


//----------------------STVARANJE AT NAREDBE--------------------------
// Primjer kako naredba treba izgledati u konačnom obliku.
// SIM900.println("AT+CPBW=,\"+38591467xxxx\",,\"Korisnik\"");

    char prvi_dio_stringa [20] = "AT+CPBW=,\"";
    char srednji_dio_stringa [12] = "\",,\"";
    char zadnji_dio_stringa [10] = "\"";
```

```
    //char zadnji_dio_stringa [30] = "\",,\"Korisnik\"";
    strcat(prvi_dio_stringa, broj);
    delay(10);
    strcat(prvi_dio_stringa, srednji_dio_stringa);
    delay(10);
    strcat(prvi_dio_stringa, Ime);
    delay(10);
    strcat(prvi_dio_stringa, zadnji_dio_stringa);
    delay(10);

    Serial.print(F("5Kompletna AT naredba je: "));
    Serial.println(prvi_dio_stringa);


    strcat(TmpStr, (getAtResult("Upis ", prvi_dio_stringa, '"', '"')));

    sms.SendSMS(phone_number,TmpStr);
    sms.DeleteSMS(position);


 //------------------------------------------------------------------------
-----------------------

  }

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

//////////////////////////////////////////////////////  SIM BOOK
DELETE  ////////////////////////////////
  else if(NULL != strstr(TmpStr,"SIM BOOK DELETE"))
  {// SIM book format

    for(int i=1; i<10;i++)
    {
      char prvi_dio_stringa [20] = "AT+CPBW=";
      char c [10];
      sprintf(c, "%d",i); //pretvara int u string
      //Serial.println(c);
      strcat(prvi_dio_stringa, c);
      //Serial.println(prvi_dio_stringa);
      delay(100);
      gsm.SimpleWriteln(prvi_dio_stringa);
      delay(100);
      prvi_dio_stringa[0]='\0';//resetiranje stringa
      prvi_dio_stringa [20] = "AT+CPBW=";
      c[0]='\0';//resetiranje stringa
    }
```

```
      sms.SendSMS(phone_number, "SIM imenik je izbrisan "); // feedback
anyway
  }

  ////////////////////////////////////////////////////////////////////////
//////////////////////
  ////////////////////////////////////////////////  SEND
SMS  ////////////////////////////////
  else if(NULL != strstr(TmpStr,"SEND SMS-"))
  {// Send SMS from SMS message on number from same SMS message.

     //Serial.print(F("TMPSTR je = "));
     //Serial.println(TmpStr);

     char * poruka_broj = strrchr(TmpStr, '-');
     poruka_broj = poruka_broj +1;

     //Serial.print(F("2Nakon _ TmpStr je = "));
     //Serial.println(poruka_broj);

     char * poruka = strrchr(poruka_broj, ':');
     poruka = poruka +1;

     //Serial.print(F("3Ime je nakon : = "));
     //Serial.println(poruka);

     char * i;

     char * broj = strtok_r(poruka_broj,":",&i); //reĹže string nakon
znaka ':'

     //Serial.print(F("4Broj za upis je = "));
     //Serial.println(broj);

     sms.SendSMS(broj, poruka); // feedback anyway
  }

  ////////////////////////////////////////////////////////////////////////
//////////////////////
  ////////////////////////////////////////////////  CALL  ////////
//////////////////////
  else if(NULL != strstr(TmpStr,"CALL-"))
  {// Call number from SMS message

     //Serial.print(F("TMPSTR je = "));
     //Serial.println(TmpStr);

     char * broj = strrchr(TmpStr, '-');
     broj = broj +1;
```

```
    Serial.print(F("Broj je = "));
    Serial.println(broj);

    call.Call(broj);

  }

  ////////////////////////////////////////////////////////////////////////
//////////////////////////

  else
  {// If none of the above
    return false;
  }
  return true;
  }
/////////////////////////////////////////  UserCmd Funkcija
////////////////////////////////
 boolean UserCmd(void)
 {// user actions
  if(!strcmp(TmpStr,"ON 1"))
  {// OUT 1 = ON
      digitalWrite(Out1,HIGH);
       if(AckOn)
       {// confirm the action with a call back to the calling number if
enabled
        Serial.println(F("Call Back Ack to: "));
        PrivacyPrint();
        //call.Call(phone_number);
        sms.SendSMS(phone_number,TmpStr);

      }
  }
  else if(!strcmp(TmpStr,"OFF 1"))
  {// OUT 1 = OFF
      digitalWrite(Out1,LOW);
      if(AckOn)
      {
        Serial.println(F("Call Back Ack to: "));
        PrivacyPrint();
       // call.Call(phone_number);
       sms.SendSMS(phone_number,TmpStr);
      }
  }
  else if(!strcmp(TmpStr,"ON 2"))
  {// OUT 2 = ON
      digitalWrite(Out2,HIGH);
      if(AckOn)
```

```
      {
        Serial.println(F("Call Back Ack to: "));
        PrivacyPrint();
        call.Call(phone_number);
      }
  }
  else if(!strcmp(TmpStr,"OFF 2"))
  {// OUT 2 = OFF
      digitalWrite(Out2, LOW);
      if(AckOn)
      {
        Serial.println(F("Call Back Ack to: "));
        PrivacyPrint();
        call.Call(phone_number);
      }
  }
  else if(!strcmp(TmpStr,"STATUS"))
  {// Request current status of the outputs without any change
    char A[2];
    strcpy(TmpStr,"Out 1=");
    itoa(digitalRead(Out1),A,2);
    strncat(TmpStr,A,2);
    strncat(TmpStr,"  Out 2=",8);
    itoa(digitalRead(Out2),A,2);
    strncat(TmpStr,A,2);
    sms.SendSMS(phone_number,TmpStr);
  }
 /* else if(!strcmp(TmpStr,"POWER ON"))
  {// Power on GSM shield by SMS message.
      powerUp();
  }
  */
  else if(!strcmp(TmpStr,"POWER OFF"))
  {// Power of GSM shield by SMS message.
      sms.DeleteSMS(position);
      delay(100);
      powerDown();
  }
  else
    {// If none of the above
      return false;
    }
  return true;
}

void PrivacyPrint(void)
{// if Privacy == OFF print the whole calling phone number
  char PriPhone[20];
  strcpy(PriPhone,phone_number);
```

```
  if(Privacy)
  {// mask out the phone number on printout to allow privacy
    memset(PriPhone,'*', 9);
  }
  Serial.print(PriPhone);
}

char * getAtResult(char * DescString, char * AtString, char StartChar,
char StopChar)
{//get the value of the AT result cleaning out of character terminators
       char * pchS;
       char * pchE;
       char gsmRead[140];
       strcat(TmpStr,DescString);
       gsm.SimpleWriteln(AtString);
       delay(1000);
       gsm.read(gsmRead, 140);
       pchS = strchr (gsmRead,StartChar);
       pchE = strrchr (gsmRead,StopChar);
       *pchE = '\0';
       return(pchS +1);
}

void ToUpperTmpStr(void)
{// command string can be both upper or lower case
  int i=0;
  while (TmpStr[i])
  {
    TmpStr[i]=toupper(TmpStr[i]);
    i++;
  }
}

void powerUp()
{
 pinMode(9, OUTPUT);
 digitalWrite(9,LOW);
 delay(1000);
 digitalWrite(9,HIGH);
 delay(2000);
 digitalWrite(9,LOW);
 delay(3000);
}

void powerDown()
{
 pinMode(9, OUTPUT);
 digitalWrite(9,LOW);
 delay(1000);
```

```
 digitalWrite(9,HIGH);
 delay(2000);
 digitalWrite(9,LOW);
 delay(3000);
}

/*
char messageSendToNumber (char primljena_poruka)
{
  char messageToSand;
  char numberToSend;
  sms.SendSMS(phone_number,TmpStr);
}
*/
```

5.7 VJEŽBA br. 21 Arduino releji

Klasični Arduino relej

Releji su električni prekidači koji mogu biti kontrolirani od strane Arduina ili bilo kojeg drugog mikrokontrolera. Njihova osnovna primjena je da kontroliraju paljenje i gašenje uređaja koji koriste visoke napone ili visoke struje u svom radu. Releje možemo definirati kao most između mikrokontrolera i uređaja kojim se upravlja.

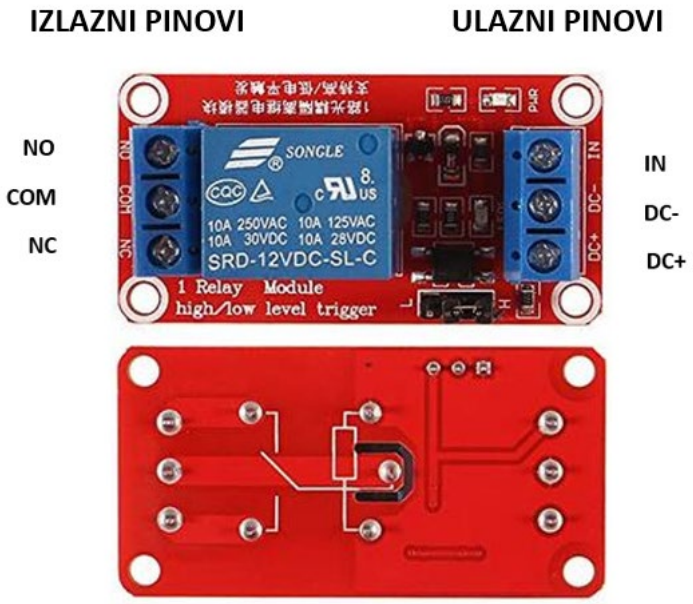
Osnovni razlog zašto ne koristimo izravno mikrokontroler nego koristimo releje u svrhu upravljanja uređajima je taj što mikrokontrolerom možemo upravljati samo uređajima malog napona i struje dok preko releja s mikrokontrolerom možemo upravljati i uređajima visoke snage te ujedno štitimo mikrokontroler od mogućeg oštećenja od strane takvih uređaja.

Slika 63 Klasični Arduino relej

Relej za prekidanje ili uspostavljanje strujnog kruga koristi elektromagnet koji otvara i zatvara strujne kontakte. Elektromagnet se obično sastoji od višestrukih namotaja bakrene žice na željeznoj jezgri. Kada struja teče kroz tu žicu koja je dio primarnog strujnog kruga oko elektromagneta se stvara magnetno polje koje privlači željeznu kotvu. Kotva nosi na sebi električne kontakte, koji onda otvaraju ili zatvaraju sekundarni strujni krug. Kada se prekine struja kroz elektromagnet, elektromagnet više ne privlači željeznu kotvu, i ona se vraća u početni položaj, obično uz pomoć opruge. Time električni kontakti prekidaju ili uspostavljaju strujni krug.

Spajanje releja

Releji imaju ulazne i izlazne pinove, stezaljke. Ulazni pinovi su upravljački niskonaponski a izlazni pinovi su upravljani visokonaponski.



Slika 64 Ulazni i izlazni pinovi za klasični Arduino relej

Ulazni pinovi:

IN signal sa Arduina kojim se upravlja okidanjem releja

DC- napajanje releja, masa.

DC+ napajanje releja, uobičajeno 5V sa Arduina

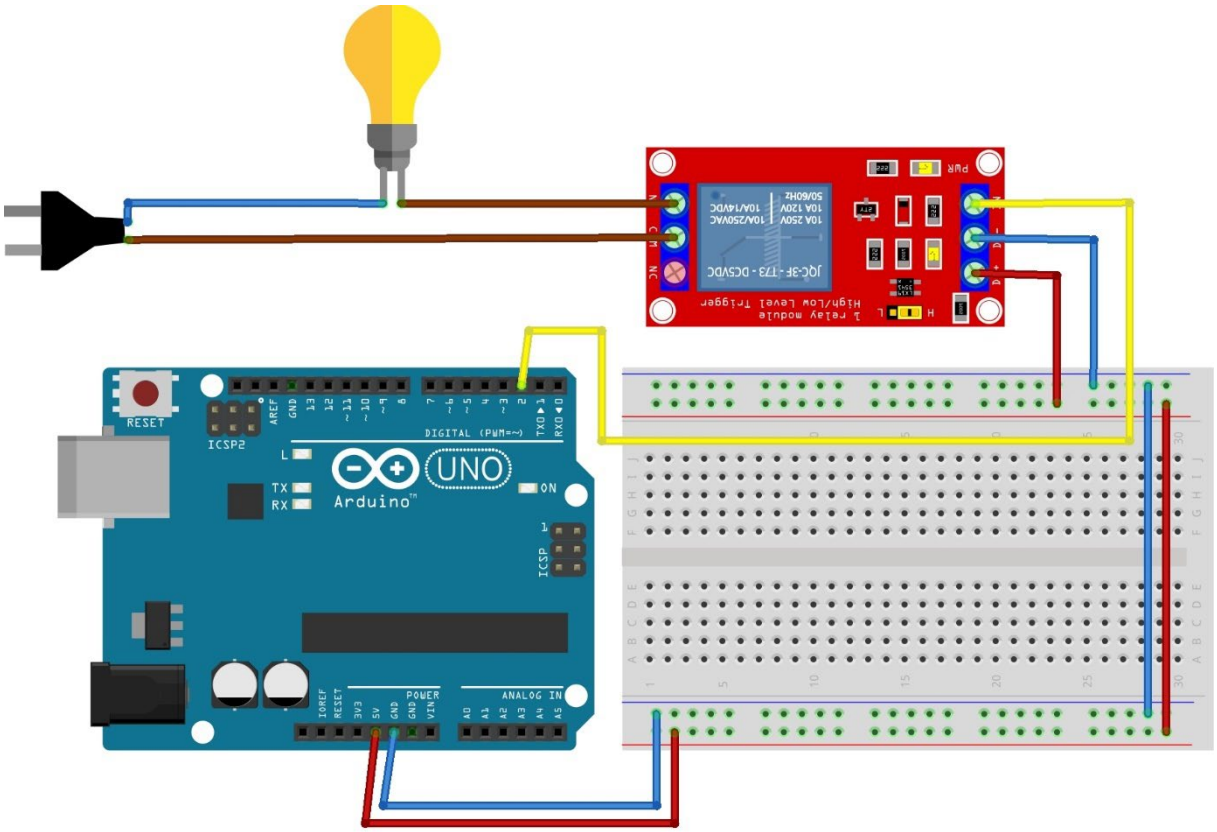
Izlazni pinovi:

NO **Normal open** – **NO** pin, nije spojen sa **COM** pinom dok ne dobije okidački signal **IN** sa primarnog strujnog kruga, nakon čega je i **NO** pin spojen sa **COM** pinom.

COM Pin na koji priključujemo korisni signal sa sekundarnog strujnog kruga koji želimo prekinuti ili proslijediti. To može biti ili napon ili uzemljenje.

NC **Normal close** – **NC** pin je spojen sa **COM** pinom dok ne dobije okidački signal **IN** sa primarnog strujnog kruga, nakon čega je i **NC** pin odvojen od **COM** pina.

L/H spojnik U zadanom stanju kada je spojnik umetnut na strani koja je označena sa **H** relej aktiviramo sa signalom visoke razine (HIGH 5V), međutim moguće je relej aktivirati i sa signalom niske razine (LOW 0V), za to je potrebno spojnik premjestiti na **L** stranu.



Slika 65 Spajanje Arduino releja, u NO modu.

Solid state releji

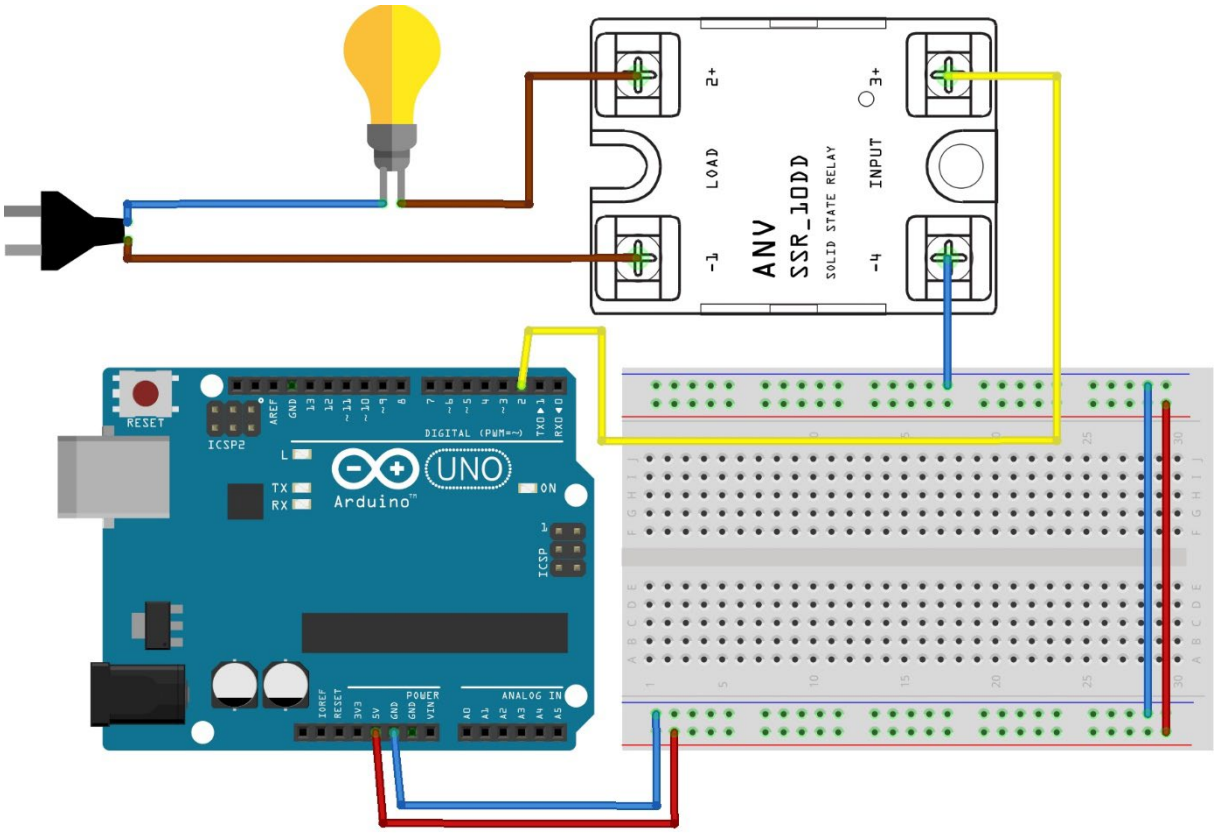
Solid State releji su poluvodički ekvivalenti elektromehaničkih releja i koriste se kao i klasični kontaktni releji za kontrolu električnih opterećenja ali beskontaktno, stoga što umjesto zavojnice i kontakata oni za prekidanje koriste poluvodičku tehnologiju.

Slika 66 Solid state relay s DC ulazom i AC izlazom

Baš kao i obični elektromehanički releji, SSR-ovi pružaju potpunu električnu izolaciju između svojih ulaznih i izlaznih kontakata sa svojim izlazom koji djeluje poput konvencionalnog električnog prekidača jer ima vrlo visok, gotovo beskonačan otpor kada nije vodljiv (otvoren) i vrlo nizak otpor. pri provođenju (zatvoreno). Poluvodički releji mogu biti dizajnirani za prebacivanje izmjenične ili istosmjerne struje korištenjem SCR, TRIAC ili izlaza sklopnog tranzistora umjesto uobičajenih mehaničkih normalno otvorenih (NO) kontakata.

Dok klasični elektromehanički releji imaju ograničen životni ciklus svojih kontakta, i imaju sporije brzine ukopčavanja, posebno veliki energetski releji i kontaktori. Solid state releji nemaju takva ograničenja. Stoga je glavna prednost poluvodičkih releja u odnosu na konvencionalne elektro-mehaničke releje to što nemaju pomičnih dijelova koji bi se istrošili, a samim time i problema s odskakivanjem kontakta, mogu se uključiti i isključiti puno brže od mehaničkih releja pri tome se mogu i pomicati, te smanjiti šumove prilikom uključivanja i isključivanja.

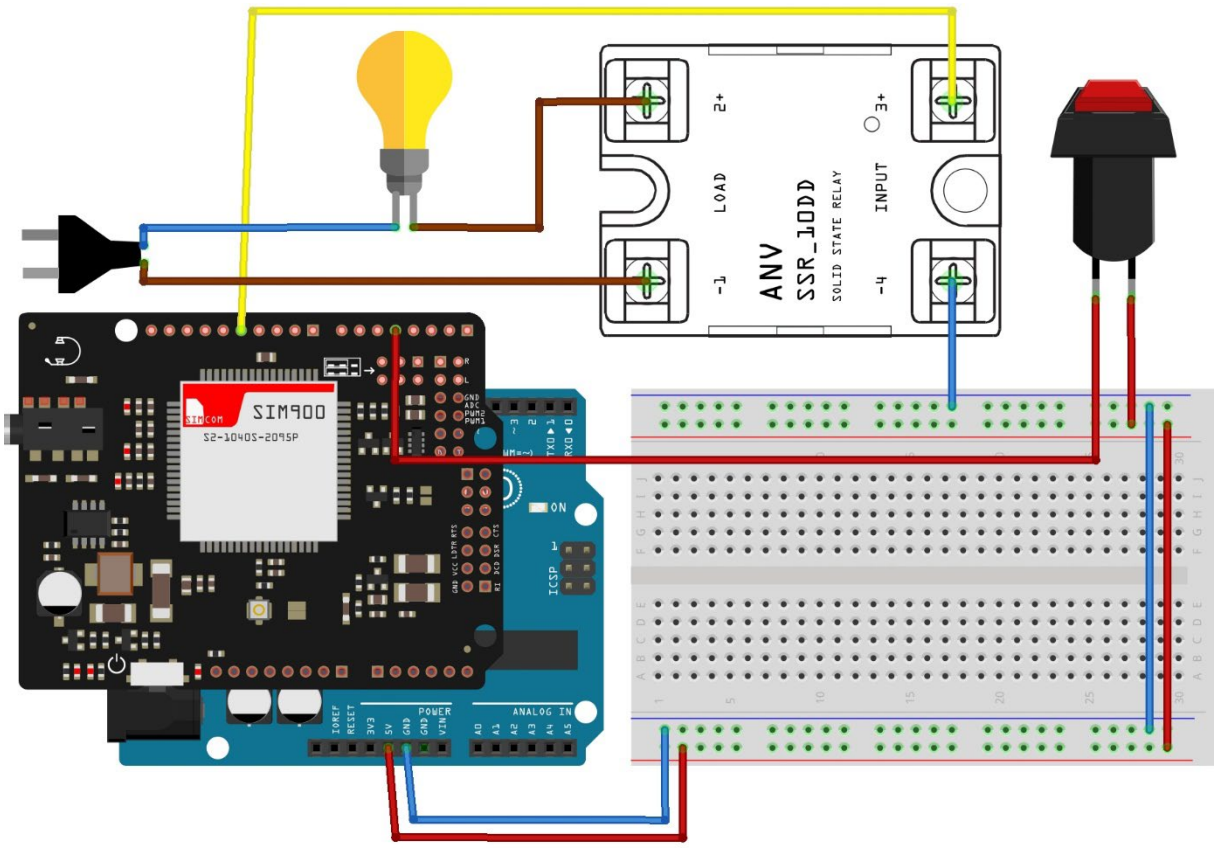
Solid state relej, spajanje



Slika 67 Solid state relej, shema spajanja

Zadatak 1.

Spojiti model uređaja za resetiranje informatičke opreme na daljinu prema zadanoj shemi te potom učitati kod „Distanc_reseter“ te zatim testirati sustav.



Slika 68 Shema za Distanc reseter

Kod:

```
/*
 DISTANC_RESETER

Ovo je upravljački program za Arduino sa ugrađenim
GSM shieldom. Pomoću tog sklopa se vrši upravljanje relejom
preko kojeg je ostvareno napajanje 220V za neki određeni
korisni uređaj.
Program osluškuje telefonski poziv preko GSM mreže
te nakon zaprimljenog poziva vrši prepoznavanje i
autorizaciju broja pozivatelja. Ukoliko je broj
pozivatelja autoriziran program odbije poziv bez
uspostave poziva.
Nakon toga mikrokontroler prekida napajanje uređaja
u vremenu od 15 sekundi te se potom samo  aktivira te ponovno
```

```
pali.
Ukoliko broj nije autoriziran nema djelovanja.
Osnovna primjena je u resetiranju telekomunikacijske opreme
na daljinu.

Sklop s ovim programom je opremljen i tasterom nakon čijeg
pritiska se aktivira poziv telefonskom broju koji je
naveden u programu. Ovo služi u svrhu provjere rada GSM mreže.

Program napisao: Joško Smolčić 4/2015


 */

#include <SoftwareSerial.h> // uključivanje SoftwareSerial.h biblioteke
SoftwareSerial SIM900(7, 8); // izbor pinova za komunikaciju

//AUTORIZACIJA_____________________________________________________
char inchar; // zadržava nadolazeći znak sa GSM shielda.
// Will hold the incoming character from the GSM shield

int onoff=0; // 0 = off, 1 = on (onoff varijabla postavljena na off)

//ZVONO_____________________________________________________________________
const int buttonPin = 5;
const int ledPin =  13;
int buttonState = 0;

//POWER
int state_9_pin = 0;
int dugme = 9;

//###########################################################################

void setup()
{
  Serial.begin(19200);

//BELL-----------------------------------
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
pinMode(13, OUTPUT); // LEDs - off = red, on = green

//DO SOMETHING----------------------------

pinMode(12, OUTPUT);

digitalWrite(12, HIGH);
```

```
//AUTORIZACIJA---------------------------------------

  SIM900.begin(19200);
  delay(100);  // vrijeme za logiranje na mrežu
  SIM900.print("AT+CLIP=1\r"); // uključuje ID pozivatelja
  delay(100);
}
//>>>>>>>>>>>>>>__ FUNKCIJA POZIVANJA
TIPKOM___>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

void zvono_()
{
   buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH); // pali LED diodu sa pina 13
    SIM900.println("ATD + +38591XXXXXXX;"); //broj koji će biti pozvan
   delay(500);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }}
//>>>>>>>>>>>>>>>>>>>>__FUNKCIJA GAŠENJA I
PALJENJA__>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

void doSomething()
{
 digitalWrite(12, LOW);
 delay (15000);
 digitalWrite(12, HIGH);
 }

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
/*
void powerUp()
{
 pinMode(9, OUTPUT);
 digitalWrite(9,LOW);
 delay(1000);
 digitalWrite(9,HIGH);
 delay(2000);
 digitalWrite(9,LOW);
 delay(3000);
}
*/

//>>>>>>>>>>>>>>>>>__FUNKCIJA ZA SAMOPOKRETANJE__>>>>>>>>>>>>>>>>>>
void power()
```

```
{
  digitalWrite(9,HIGH);
  int state_9_pin = digitalRead(dugme);
  if (state_9_pin = LOW)
  {
   // powerUp();
    digitalWrite(9,HIGH);
    }
  }
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     void authorization()
{
  //Prvi korisnik
  if(SIM900.available() >0)
  {
    inchar=SIM900.read();
    if (inchar=='+')
    {
      delay(10);
      inchar=SIM900.read();
      if (inchar=='3')
      {
        delay(10);
        inchar=SIM900.read();
        if (inchar=='8')
        {
          delay(10);
          inchar=SIM900.read();
          if (inchar=='5')
          {
            delay(10);
            inchar=SIM900.read();
            if (inchar=='9')
            {
              delay(10);
              inchar=SIM900.read();
              if (inchar=='1')
              {
                delay(10);
                inchar=SIM900.read();
                if (inchar=='x')
                {
                  delay(10);
                  inchar=SIM900.read();
                  if (inchar=='x')
                  {
                    delay(10);
                    inchar=SIM900.read();
                    if (inchar=='x')
```

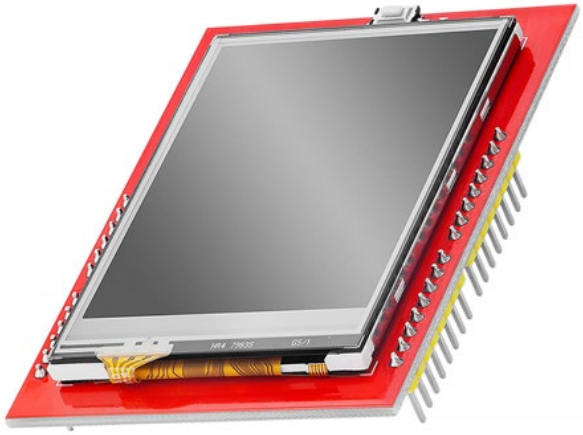
```
                    {
                      delay(10);
                      inchar=SIM900.read();
                      if (inchar=='x')
                      {
                        delay(10);
                      inchar=SIM900.read();
                      if (inchar=='x')
                      {delay(10);
                      inchar=SIM900.read();
                      if (inchar=='x')
                      {delay(10);
                      inchar=SIM900.read();
                      if (inchar=='x')
                      {
                        Serial.println("do something");
                        delay(10);
                        //digitalWrite(12, HIGH);
                        // ako je broj prepoznat izvršava se funkcija
doSomething
                        doSomething(); // poziv za izvršavanje funkcije
                         SIM900.print("ATH\r"); // odbijanje poziva
pozivatelja

                        delay(100);
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
//Drugi korisnik
if(SIM900.available() >0)
  {
    inchar=SIM900.read();
    if (inchar=='+')
    {
      delay(10);
      inchar=SIM900.read();
      if (inchar=='3')
      {
```

```
        delay(10);
        inchar=SIM900.read();
        if (inchar=='8')
        {
          delay(10);
          inchar=SIM900.read();
          if (inchar=='5')
          {
            delay(10);
            inchar=SIM900.read();
            if (inchar=='9')
            {
              delay(10);
              inchar=SIM900.read();
              if (inchar=='9')
              {
                delay(10);
                inchar=SIM900.read();
                if (inchar=='X')
                {
                  delay(10);
                  inchar=SIM900.read();
                  if (inchar=='X')
                  {
                    delay(10);
                    inchar=SIM900.read();
                    if (inchar=='X')
                    {
                      delay(10);
                      inchar=SIM900.read();
                      if (inchar=='X')
                      {
                        delay(10);
                      inchar=SIM900.read();
                      if (inchar=='X')
                      {delay(10);
                      inchar=SIM900.read();
                      if (inchar=='X')
                      {delay(10);
                      inchar=SIM900.read();
                      if (inchar=='X')
                      {
                        Serial.println("do sometehing");
                        delay(10);
                        //digitalWrite(12, HIGH);

                        doSomething();
                        SIM900.print("ATH\r");
```

```
                        delay(100);
                        SIM900.println("ATD + +385914672230;");
                        delay(100);
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
}
//#########################################################################


void loop()
{
  power(); // izvršavanje funkcije za samo uključivanje

    zvono_();  // izvršavanje funkcije za pozivanje

     authorization(); // izvršavanje funkcije za autorizaciju
}
```

5.8 VJEŽBA br. 22 TFT LCD 2,4“ zaslon

TFT LCD (Thin Film Transistor Liquid Crystal Display) modul veličine 2,4“ je jedan od najprikladniji i najkorištenijih LCD modula na dodir koji se koristi u raznim Arduino aplikacijama. Modul ima zaslon u boji te se njegova veličina podudara sa veličinom Arduino UNO pločice čime čine kompaktnu cijelinu. Na ekranu veličine 240 x 320 piksela se osim što je osjetljiv na dodir također mogu prikazivati i slike. Osim prikladnosti u smislu veličine i upravljanja, ono što ovaj LCD zaslon čini dodatno prihvatljivim za razne aplikacije i primjene je i njegova pristupačna cijena jer spada u grupu jeftinijih LCD zaslona u boji osjetljivih na dodir.



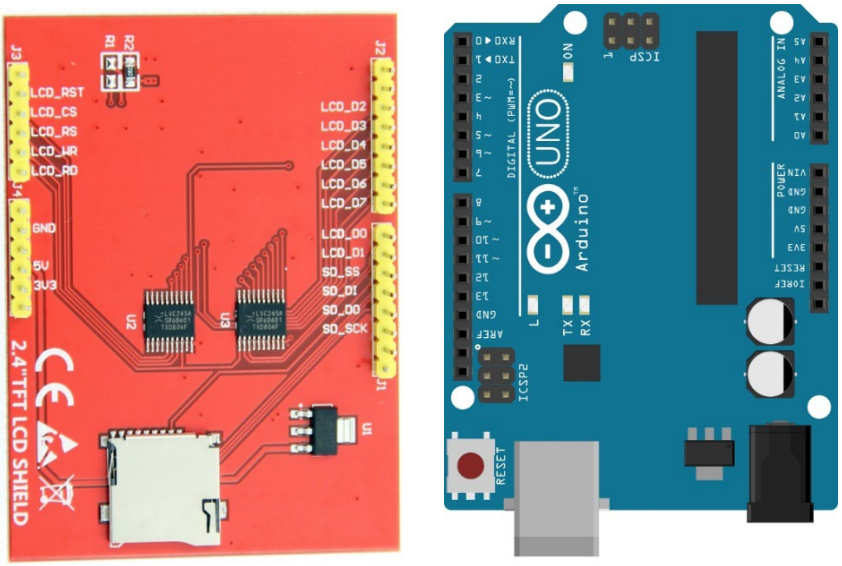
Slika 69 TFT LCD 2,4" zaslon na dodir

LCD ima odličan kontrast živih boja i dolazi s ugrađenim priključkom za microSD karticu. Ovaj TFT zaslon sastoji se od jakog pozadinskog osvjetljenja (4 bijela LED pozadinska osvjetljenja) i šarenog zaslona od 240X320 piksela. Također ima individualnu kontrolu RGB piksela koja daje mnogo bolju rezoluciju od crno-bijelih 128×64 zaslona.

Karakteristike TFT LCD 204“ zaslona

Naziv	Karakteristike
Boja zaslona	RGB 65K boja
Veličina zaslona	2,4 (inča)
Tip zaslona	TFT (Thin Film Transistor)
Driver	IC ILI9341
Rezolucija	320*240 (piksel)
Sučelje modula	8-bitno paralelno sučelje
Aktivno područje	48,96*36,72 (mm)
Veličina PCB modula	72,20*52,7 (mm)
Radna temperatura	-20°C~60°C
Temperatura skladištenja	-30°C~70°C
Radni napon	5V/3.3V
Potrošnja energije	TBD (nije određeno)
Težina proizvoda	(paket sadrži) 39 (g)

Raspored pinova na TFT LCD 2,4“ zaslonu



Slika 70 Raspored pinova TFT LCD 2,4" i Arduino UNO R3

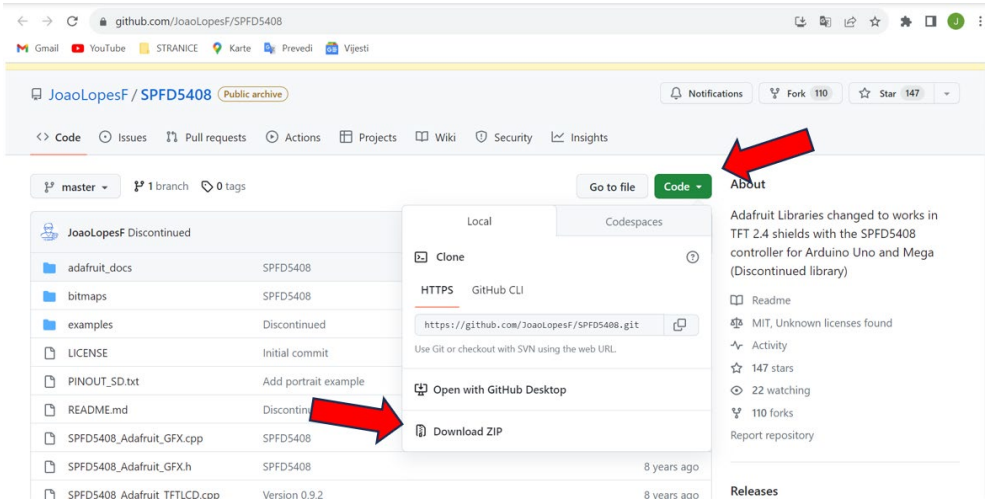
Naziv pina na LCD	Arduino UNO pin	Opis pina	Namjena pina
LCD_RST	A4	Signal resetiranja LCD sabirnice, niska razina resetiranja	LCD upravljački pinovi
LCD_CS	A3	Signal odabira čipa LCD sabirnice, omogućena niska razina	
LCD_RS	A2	Naredba LCD sabirnice / signal odabira podataka, niska razina: naredba, visoka razina: podaci	
LCD_WR	A1	Signal pisanja LCD sabirnice	
LCD_RD	A0	Signal čitanja LCD sabirnice	
LCD_D0	D8	LCD 8-bitni podaci Bit 0	LCD podatkovni pinovi
LCD_D1	D9	LCD 8-bitni podaci Bit 1	
LCD_D2	D2	LCD 8-bitni podaci Bit 2	
LCD_D3	D3	LCD 8-bitni podaci Bit 3	
LCD_D4	D4	LCD 8-bitni podaci Bit 4	
LCD_D5	D5	LCD 8-bitni podaci Bit 5	
LCD_D6	D6	LCD 8-bitni podaci Bit 6	
LCD_D7	D7	LCD 8-bitni podaci Bit 7	
SD_SS	D10	Signal odabira čipa SPI sabirnice SD kartice, omogućena niska razina	SD kartica, podatkovni pinovi
SD_DI	D11	SD_DI SD kartica SPI sabirnica MOSI signal	
SD_DO	D12	SD kartica SPI sabirnica MISO signal	
SD_SCK	D13	Taktni signal SPI sabirnice SD kartice	
GND	GND	Masa, uzemljenje	Napajanje
5V	5V	Ulazno napajanje 5V	
3,3V	3,3V	Ulazno napajanje od 3,3 V, ovaj pin se može isključiti	

Priprema i pokretanje TFT LCD 2,4“ zaslona.

Ako imamo ispunjene hardverske zahtjeve, Arduino ploču i 2,4 inčni TFT Shield, povezivanje TFT LCD-a s Arduinom vrlo je jednostavno. Tada moramo ispuniti i softverske zahtjeve a to znači imati odgovorajuću Arduino IDE & TFT biblioteku. Na internetu su dostupne mnoge biblioteke za rad TFT LCD Shielda, ali različiti TFT LCD-i imaju različite ugrađene upravljačke programe. Dakle, prvo moramo identificirati upravljački program za TFT LCD Shield i zatim instalirati odgovarajuću biblioteku za taj program. Ovdje koristimo 2,4 inčni TFT Shield koji ima ILI9341 upravljački program (Driver). Ovo su pojedinačni koraci za pripremu i pokretanje TFT LCD zaslona:

Korak 1: preuzmite TFT LCD biblioteku za Arduino s donje poveznice u ZIP formatu.

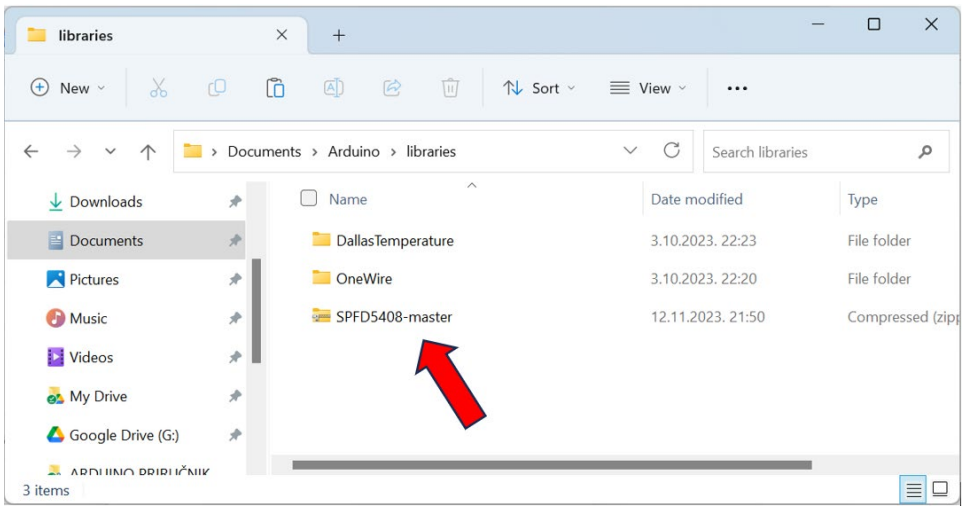
https://github.com/JoaoLopesF/SPFD5408



Slika 71 TFT LCD biblioteku za Arduino

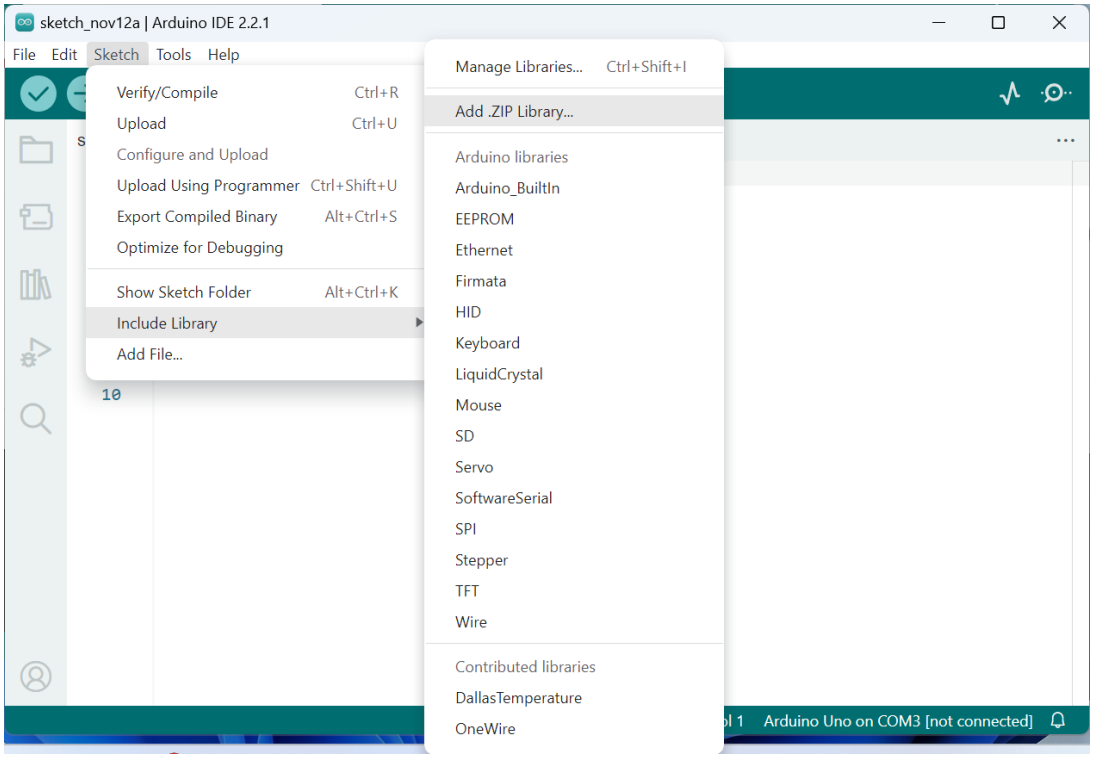
Korak 2: Nakon ovoga, kopirajte biblioteku SPFD5408 u folder na sljedećoj adresi:

C:\Users\Ime korisnika ili računala\Documents\Arduino\libraries



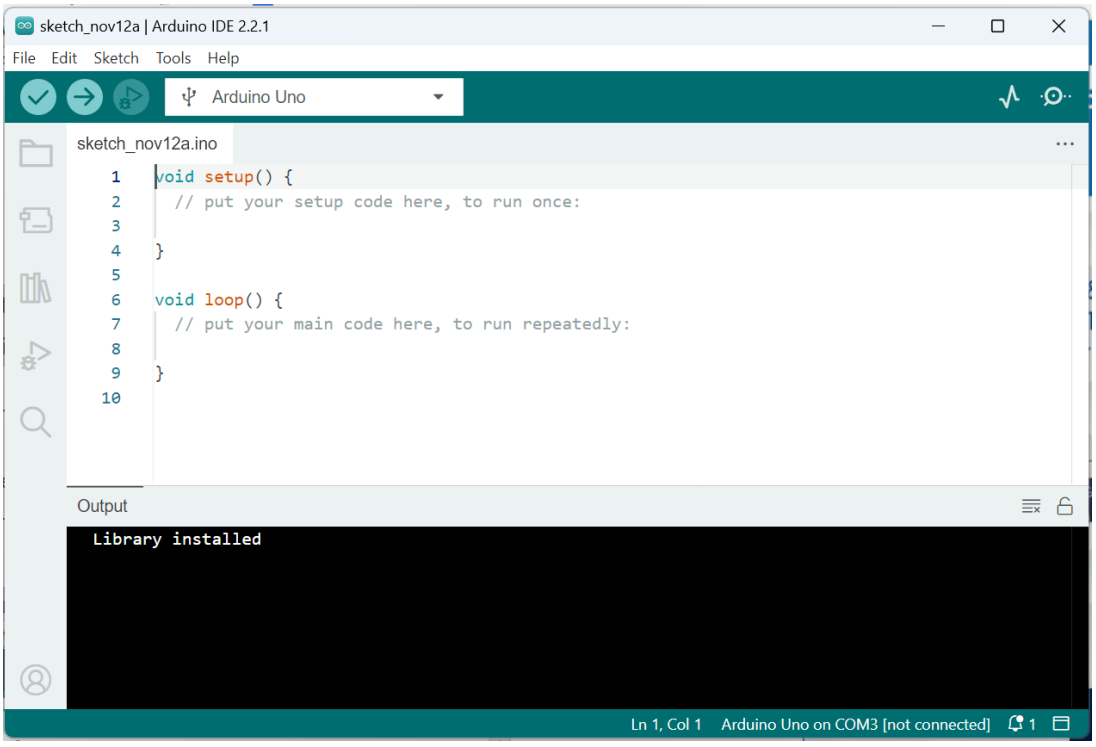
Slika 72 Kopiranje biblioteke SPFD5408 u folder

Korak 3: Sad otvorite **Arduino IDE** te izaberite **Sketch -> Include Library -> Add .ZIP Library**



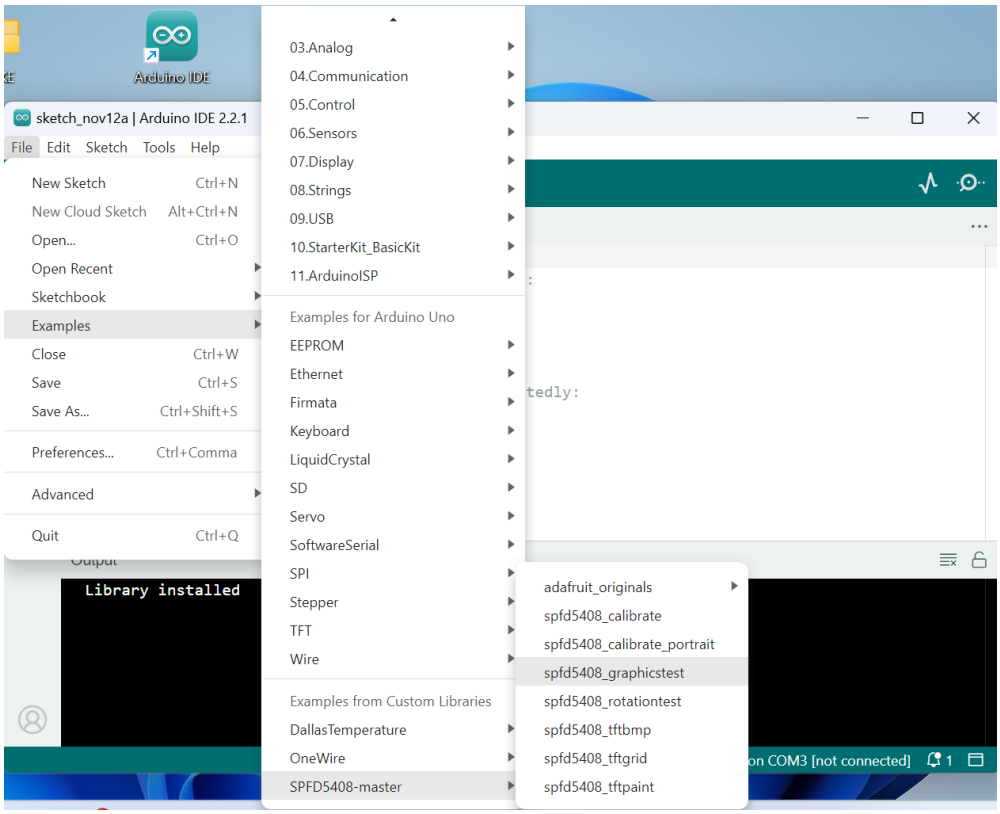
Slika 73 Sketch -> Include Library -> Add .ZIP Library

Sada nakon otvaranja SPFD5408 Master biblioteke, možete vidjeti da je datoteka vaše biblioteke instalirana u Arduino IDE.



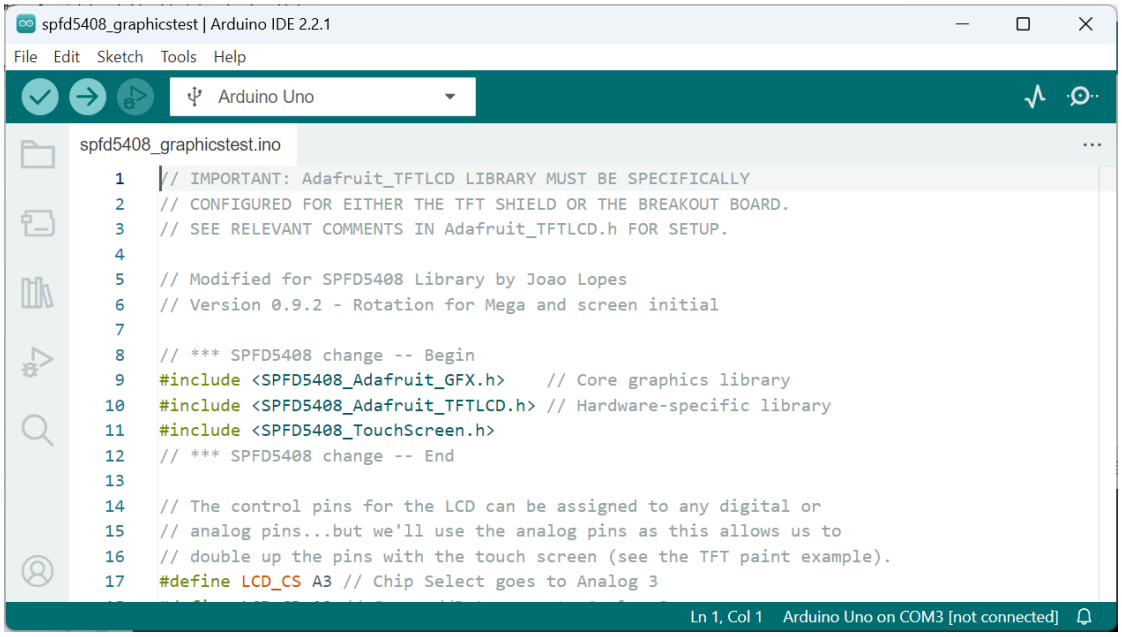
Slika 74 Biblioteka instalirana u Arduino IDE

Korak 4: Sada unutar Arduino IDE programa idite na , **File -> Example -> SPFD5408-master -> spfd5408_graphictest.** Otvorite program, kompajlirajte ga te potom učitajte u Arduino.



Slika 75 File -> Example -> SPFD5408-master -> spfd5408_graphictest

Nakon učitavanja na TFT LCD zaslonu će te dobiti testni ispis. Korisnici mogu modificirati ovaj testni kod prema svojim zahtjevima i potrebama za testiranje zaslona.



Slika 76 Testni ispis

Priručnik za korištenje

Na slijedećm linku je dostupan korisnički priručnik za TFT LCD 2,4“ zaslon:

http://www.lcdwiki.com/res/MAR2406/2.4inch_Arduino_8BIT_Module_MAR2406_User_Manual_EN.pdf

Zadatak 1.

Spojiti model uređaja za autorizaciju e-bicikla prema zadanoj shemi te potom učitati kod „E-bicikla LCD_key“ te zatim testirati sustav.

Kod:

```
/*////////////////////////////////////////////////////////////////////////
////////////////////////////////////
  * File:             E-bicikla LCD_key,  OBRTNA TEHNIČKA ŠKOLA SPLIT
  * Autor:            Joško Smolčić
  * Verzija:          1.0
  * Datum izrade:     09.11.2022
  *
  *
  * ___________________________________________________
  * DETALJI O PROGRAMU:
  * ___________________________________________________
  *
  *  E-bicikla LCD_key, je program koji služi u svrhu upravljanja Arduino
Uno mikrokontrolerskom pločicom uz korištenje "2,4 TFT (thin-film
transistor) LCD " modula, koji je opremljen sa Driver IC: ILI9341
procesorom.
  *  U radu je korišten ovaj 2,4 TFT  LCD " modul:
  *  http://www.lcdwiki.com/2.4inch_Arduino_Display
  *
  * Program se koristi bibliotekom čiji je tvorac Joao Lopes:
  * https://github.com/JoaoLopesF/SPFD5408
  *
  *
  * E-bicikla LCD_key, program se bazira na programu od: IOT DESIGN PRO -->
  * https://iotdesignpro.com/projects/touch-screen-solenoid-door-lock-
system-using-arduino  koji također koristi neznatno promijenjenu biblioteku
od Joao Lopes.
  *
  *  E-bicikla LCD_key  je slobodan softver i može kopirati ili
nadograđivati i koristiti od strane bilo koga. Svrha programa je da koristi
svima koji trebaju ovakvu vrstu programa i ne daje nikakvo jamstvo za
upotrebu. SVAKO KORIŠTENJE JE NA VLASTITU ODGOVORNOST.
```

```
  *
  *
  * _____________________________________________________
  * FUNKCIJE PROGRAMA:
  * _____________________________________________________
  *
  * Ovim programom je omogućena autorizacija lozinkom prilikom pokretanja e-
bicikla.
  * Također poznavaocu lozinke je omogućena promjena lozinke.
  * Nakon uspješne autorizacije sklop daje izlazni napon (5V) na pinu 11,
Arduino UNO.
  * Nakon jednom pokrenute autorizacije sklop konstantno daje napon te dok
se ne isključi napajanje moguće je koristiti e-bicikl. Nakon isključenog
napajanja potrebno je opet izvršiti autorizaciju za novo korištenje.
  *
  *
  *
  * KORISNIČKE KOMANDE U PROGRAMU:
  * ------------------------------
  *
  * Nakon unošenja lozinke, lozinku potvrđujemo na tipku OK-:
  * Ukoliko želimo promjeniti lozinku potrebno je dva puta pritisnuti tipku
X.
  * Nakon toga se otvara dijalog gdje teba potvrdno odgovoriti na pitanje
Promjeniti PIN?
  * Nakon potvrdnog odgovora upisujemo važeći pin nakon čega ako je pin
ispravan imamo mogućnost upisa
  * novog željenog pina nakon OK potvrde.
  *
  * DODATNE IZMJENE U PROGRAMU U BIBLIOTECI:
  * ------------------------------
  *
  * S obzirom da se radi o jeftinom kineskom klonu 2,4 TFT LCD modulu sa
ILI9341 procesorom
  *
https://www.aliexpress.com/item/32618152565.html?spm=a2g0o.order_list.0.0.41
d31802ELrPMx
  * bile su neophodne promjene u programu i biblioteci.
  *
  * S obzirom da je biblioteka pisana izvorno za SPFD5408 Driver, unutar
programa je bilo neophodno podesiti na sljedeći način:
  *
  *#define YP A2  // must be an analog pin, use "An" notation!
   #define XM A3  // must be an analog pin, use "An" notation!
   #define YM 8   // can be a digital pin
   #define XP 9   // can be a digital pin


   #define WHITE   0xFFFF //Black->White (0x0000 prethodna vrijednost)
```

```
   #define YELLOW    0xFFE0 //Blue->Yellow (0x001F prethodna vrijednost)
   #define CYAN     0x07FF //Red->Cyan  (0xF800 prethodna vrijednost)
   #define PINK   Green //Green-> Pink  (0x07E0 prethodna vrijednost)
   #define RED    0xF800 //Cyan -> Red  (0x07FF prethodna vrijednost)
   #define GREEN 0x07E0 //Pink -> Green   (0xF81F prethodna vrijednost)
   #define BLUE  0x001F //Yellow->Blue (0xFFE0 prethodna vrijednost)
   #define BLACK   0x0000 //White-> Black (0xFFFF prethodna vrijednost)
  *
  *
  * ____Calibrate TFT LCD_____
  #define TS_MINX 125
  #define TS_MINY 905
  #define TS_MAXX 965
  #define TS_MAXY 85

  * te podešavanje prikaza.
  *
  * Na biblioteci SPFD5408_TouchScreen.cpp je bilo neophodno promjeniti
liniju koda 159
  *
  * return TSPoint(1103 - x, 1023 - y, z);

   // *** SPFD5408 change -- Begin
   // SPFD5408 change, because Y coordinate is inverted in this controller
   return TSPoint( x,  1023 - y, z);




Napomene:
  *  Ovaj modul daje obrnuti prikaz te je potrebno izvršiti kalibraciju

  *  Za kalibraciju je korišten program unutar example ponude u biblioteci:
spfd5408_tftpaint.


 *////////////////////////////////////////////////////////////////////////
//////////////////////////


#include <SPFD5408_Adafruit_GFX.h>    // Core graphics library
#include <SPFD5408_Adafruit_TFTLCD.h> // Hardware-specific library
#include <SPFD5408_TouchScreen.h>
#include <EEPROM.h>
#define YP A2  // must be an analog pin, use "An" notation!
#define XM A3  // must be an analog pin, use "An" notation!
```

```
#define YM 8   // can be a digital pin
#define XP 9   // can be a digital pin
#define LCD_CS A3
#define LCD_CD A2
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4
#define WHITE   0xFFFF //Black->White (0x0000 prethodna vrijednost)
#define YELLOW    0xFFE0 //Blue->Yellow (0x001F prethodna vrijednost)
#define CYAN     0x07FF //Red->Cyan  (0xF800 prethodna vrijednost)
#define PINK   Green //Green-> Pink  (0x07E0 prethodna vrijednost)
#define RED    0xF800 //Cyan -> Red  (0x07FF prethodna vrijednost)
#define GREEN 0x07E0 //Pink -> Green   (0xF81F prethodna vrijednost)
#define BLUE  0x001F //Yellow->Blue (0xFFE0 prethodna vrijednost)
#define BLACK   0x0000 //White-> Black (0xFFFF prethodna vrijednost)
#define MINPRESSURE 10
#define MAXPRESSURE 1000
/*_______Assigned______*/
/*____Calibrate TFT LCD_____*/
#define TS_MINX 125
#define TS_MINY 905
#define TS_MAXX 965
#define TS_MAXY 85
/*______End of Calibration______*/
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300); //300 is the sensitivity
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET); //Start
communication with LCD
String symbol[4][3] = {
  { "1", "2", "3"},
  { "4", "5", "6"},
  { "7", "8", "9"},
  { "X", "0", "OK"}
};
int X,Y;
int Number;
int relayPin = 11;// set pin 11 for relay output
int buzzer = 12;
int pass, currentpage;
int password;
int addr = 0;
int addr1 = 1;
unsigned long interval = 2000;
unsigned long previousMillis = 0;
unsigned long currentMillis;

void setup()
{
  Serial.begin(9600); //Use serial monitor for debugging
  tft.reset(); //Always reset at start
```

```
  tft.begin(0x9341); // My LCD uses LIL9341 Interface driver IC
  tft.setRotation(8); //
  tft.fillScreen(WHITE);
  draw_BoxNButtons();
  pinMode(relayPin, OUTPUT);
  pinMode(buzzer, OUTPUT);
  //digitalWrite(relayPin, LOW);// set relay pin to LOW
  currentpage = 0;
  pass = 0;
}
void loop() {


currentMillis = millis(); // grab current time
TSPoint p = waitTouch();
X = p.y; Y = p.x;
//  Serial.print(X); Serial.print(','); Serial.println(Y);// + " " + Y);
DetectButtons();
delay(300);



}


TSPoint waitTouch() {
  TSPoint p;
  do {
    p = ts.getPoint();
    pinMode(XM, OUTPUT);
    pinMode(YP, OUTPUT);
  } while((p.z < MINPRESSURE )|| (p.z > MAXPRESSURE));
  p.x = map(p.x, TS_MINX, TS_MAXX, 0, 320);
  p.y = map(p.y, TS_MINY, TS_MAXY, 0, 240);;
  return p;
}



void DetectButtons()
{
   if (currentpage == 0)
   {
    tft.fillRect(0, 0, 240, 80, CYAN);  //clear result box
    tft.setCursor(10, 20);
    tft.setTextSize(4);
    tft.setTextColor(BLACK);
  if (X<80 && X>0) //Detecting Buttons on Column 1
  {
    if (Y>0 && Y<65) //If cancel Button is pressed
```

```
    {Serial.println ("Cancel");
    Number=0;
    if ((unsigned long)(currentMillis - previousMillis) >= interval) {
      Serial.print("Detected");
      currentpage = 1;
      previousMillis = millis();
    }
    }
     if (Y>65 && Y<120) //If Button 1 is pressed
    {Serial.println ("Button 7");
    if (Number==0)
    Number=7;
    else
    Number = (Number*10) + 7; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>120 && Y<180) //If Button 4 is pressed
    {Serial.println ("Button 4");
    if (Number==0)
    Number=4;
    else
    Number = (Number*10) + 4; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>180 && Y<240) //If Button 1 is pressed
    {Serial.println ("Button 1");
    if (Number==0)
    Number=1;
    else
    Number = (Number*10) + 1; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
  }
    if (X<160 && X>80) //Detecting Buttons on Column 2
  {
    if (Y>0 && Y<65)
    {Serial.println ("Button 0"); //Button 0 is Pressed
    if (Number==0)
    Number=0;
    else
    Number = (Number*10) + 0; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>65 && Y<120)
    {Serial.println ("Button 8");
```

```
    if (Number==0)
    Number=8;
    else
    Number = (Number*10) + 8; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>120 && Y<180)
    {Serial.println ("Button 5");
    if (Number==0)
    Number=5;
    else
    Number = (Number*10) + 5; //Pressed twic
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>180 && Y<240)
    {Serial.println ("Button 2");
    if (Number==0)
    Number=2;
    else
    Number = (Number*10) + 2; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
  }
    if (X<240 && X>160) //Detecting Buttons on Column 3
  {
    if (Y>0 && Y<65)
    {Serial.println ("OK");
    checkPassword();
    Number = 0;

    delay(150);
    }
     if (Y>65 && Y<120)
    {Serial.println ("Button 9");
    if (Number==0)
    Number=9;
    else
    Number = (Number*10) + 9; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>120 && Y<180)
    {Serial.println ("Button 6");
    if (Number==0)
    Number=6;
    else
```

```
    Number = (Number*10) + 6; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>180 && Y<240)
    {Serial.println ("Button 3");
    if (Number==0)
    Number=3;
    else
    Number = (Number*10) + 3; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
  }
}
if (currentpage == 1) {
    tft.fillScreen(BLACK);
    tft.setCursor(10, 40);
    tft.setTextSize(4);
    tft.setTextColor(YELLOW);
    tft.println("Promijeni");
    tft.println("   PIN?");
    tft.fillRect  (35,180,60,50,GREEN);
    tft.fillRect  (140,180,60,50,RED);
    //tft.println(" ");
    tft.setTextSize(3);
    tft.setTextColor(WHITE);
    tft.setCursor(45, 190);// lijevi veći broj pomiče X os s lijeva na
desno, desni veći broj Y os  odozgora  prema dolje.
    tft.println("Da");
    tft.setCursor(157, 190); // lijevi veći broj pomiče X os s lijeva na
desno, desni veći broj Y os  odozgora  prema dolje.
    tft.println("Ne");
    if (Y>100 && Y<140)
    {
     if (X>35 && X<95){
      Serial.print("Da");
      setup();
      pass = 1;
      checkPassword();
     }
     if (X>140 && X<200){
      Serial.print("No");
      setup();
     }
   }
}
if (currentpage == 2)
{
```

```
    tft.fillRect(0, 0, 240, 80, CYAN);  //clear result box
    tft.setCursor(10, 20);
    tft.setTextSize(4);
    tft.setTextColor(BLACK);
  if (X<80 && X>0) //Detecting Buttons on Column 1
  {
    if (Y>0 && Y<80) //If cancel Button is pressed
    {Serial.println ("Set");
    Serial.println(Number);
    int password1 = (Number / 100); //12 First two digit of entered number
    int password2 = (Number % 100); //34 Last two digit of entered number
    Serial.println(password1);
    Serial.println(password2);
    EEPROM.write(addr, password1);
    EEPROM.write(addr1, password2);
    setup();
    Number = 0;
    }
     if (Y>80 && Y<140) //If Button 1 is pressed
    {Serial.println ("Button 7");
    if (Number==0)
    Number=7;
    else
    Number = (Number*10) + 7; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>140 && Y<200) //If Button 4 is pressed
    {Serial.println ("Button 4");
    if (Number==0)
    Number=4;
    else
    Number = (Number*10) + 4; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>200 && Y<260) //If Button 1 is pressed
    {Serial.println ("Button 1");
    if (Number==0)
    Number=1;
    else
    Number = (Number*10) + 1; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
  }
    if (X<160 && X>80) //Detecting Buttons on Column 2
  {
    if (Y>0 && Y<85)
```

```
    {Serial.println ("Button 0"); //Button 0 is Pressed
    if (Number==0)
    Number=0;
    else
    Number = (Number*10) + 0; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>85 && Y<140)
    {Serial.println ("Button 8");
    if (Number==0)
    Number=8;
    else
    Number = (Number*10) + 8; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>140 && Y<192)
    {Serial.println ("Button 5");
    if (Number==0)
    Number=5;
    else
    Number = (Number*10) + 5; //Pressed twic
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>192 && Y<245)
    {Serial.println ("Button 2");
    if (Number==0)
    Number=2;
    else
    Number = (Number*10) + 2; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
  }
    if (X<240 && X>160) //Detecting Buttons on Column 3
  {
   // if (Y>0 && Y<85)
    //{Serial.println ("OK");
    //delay(150);
    //}
     if (Y>85 && Y<140)
    {Serial.println ("Button 9");
    if (Number==0)
    Number=9;
    else
    Number = (Number*10) + 9; //Pressed twice
    tft.println(Number); //update new value
```

```
    delay(150);
    }
     if (Y>140 && Y<192)
    {Serial.println ("Button 6");
    if (Number==0)
    Number=6;
    else
    Number = (Number*10) + 6; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
    }
     if (Y>192 && Y<245)
    {Serial.println ("Button 3");
    if (Number==0)
    Number=3;
    else
    Number = (Number*10) + 3; //Pressed twice
    tft.println(Number); //update new value
    delay(150);
      }
    }
  }
}


void draw_BoxNButtons()
{
  //Draw the Result Box
  tft.fillRect(0, 0, 240, 80, CYAN);
  tft.setCursor(2, 5);
  tft.setTextSize(2);
  tft.setTextColor(BLUE);
  tft.println(" OBRTNA TEH. SKOLA ");
   tft.setTextColor(BLACK);
   tft.setCursor(37, 27);
   tft.setTextSize(3);
  tft.println("Unesi PIN");

  tft.setCursor(70, 60);
  tft.setTextSize(2);
  tft.setTextColor(BLUE);
  tft.println(" SPLIT ");
 //Draw First Column
  tft.fillRect  (0,260,80,60,RED);
  tft.fillRect  (0,200,80,60,BLACK);
  tft.fillRect  (0,140,80,60,BLACK);
  tft.fillRect  (0,80,80,60,BLACK);
 //Draw SEcond Column
  tft.fillRect  (80,260,80,60,BLACK);
```

```
  tft.fillRect  (80,200,80,60,BLACK);
  tft.fillRect  (80,140,80,60,BLACK);
  tft.fillRect  (80,80,80,60,BLACK);
 //Draw Third Column
  tft.fillRect  (160,260,80,60,BLUE);
  tft.fillRect  (160,200,80,60,BLACK);
  tft.fillRect  (160,140,80,60,BLACK);
  tft.fillRect  (160,80,80,60,BLACK);
  //Draw Horizontal Lines
  for (int h=80; h<=320; h+=60)
  tft.drawFastHLine(0, h, 240, WHITE);
  //Draw Vertical Lines
  for (int v=0; v<=240; v+=80)
  tft.drawFastVLine(v, 80, 240, WHITE);
  //Display keypad lables
  for (int j=0;j<4;j++) {
    for (int i=0;i<3;i++) {
      tft.setCursor(22 + (85*i), 100 + (60*j));
      tft.setTextSize(3);
      tft.setTextColor(WHITE);
      tft.println(symbol[j][i]);
    }
  }
}
void checkPassword() {
     int value1 = EEPROM.read(addr) ;
     int value2 = EEPROM.read(addr1) ;
     password = (value1*100)+(value2);
  if (pass == 0){
     Serial.print(password);
     Serial.println();
     Serial.print(Number);
     if (Number == password){
      Serial.println("Access Granted");
      Serial.println("Relay ON");
      tft.fillScreen(BLACK);
      tft.setCursor(10, 20);
      tft.setTextSize(4);
      tft.setTextColor(BLUE);
      tft.println("OBRTNA");
      tft.println(" TEHNICKA");
      tft.println(" SKOLA");
      tft.println(" ");
      //tft.println(" E bicikla");
      //tft.println(" ");
      tft.setTextColor(GREEN);
      tft.println(" Pristup");
      tft.println(" dozvoljen");
       tft.setTextColor(RED);
```

```
      tft.println(" Vozii!!");
      digitalWrite(relayPin, HIGH);// Turn the relay switch ON to drive e-
bike.

      digitalWrite(buzzer, HIGH);
      delay(1000);
      digitalWrite(buzzer, LOW);
      Serial.println("Relay OFF");
      delay(6000);
      //digitalWrite(relayPin, LOW);// set relay pin to LOW

}
     else {
      Serial.println("Pristup odbijen");
      tft.fillScreen(BLACK);
      tft.setCursor(15, 80);
      tft.setTextSize(5);
      tft.setTextColor(RED);
      tft.println("Pristup");
      tft.println("  ");
      tft.setCursor(15, 160);
      tft.println("ODBIJEN");
      delay(3000);
      //digitalWrite(relayPin, LOW);// set relay pin to LOW
   }
     Number = 0;
     setup();
   }
if (pass==1){
  tft.fillRect(0, 0, 240, 80, CYAN);
  tft.setCursor(5, 10);
  tft.setTextSize(2);
  tft.setTextColor(BLACK);
  tft.println("Unesi trenutni PIN:");
  if (X<240 && X>160 && Y>0 && Y<85)
    {
      Serial.println ("OK");
      Serial.print(password);
      Serial.println(Number);
    if (Number == password){
      Number = 0;
      tft.fillRect(0, 0, 240, 80, CYAN);
      tft.setCursor(5, 10);
      tft.setTextColor(BLACK);
      currentpage = 2;
      tft.println("Unesi novi PIN:");
      delay(3000);
      if (X<80 && X>0 && Y>0 && Y<80){
        Serial.print("Set");
```

```
      }
      }
    else{
      Serial.println("Krivi PIN");
      tft.fillScreen(BLACK);
      tft.setCursor(60, 100);
      tft.setTextSize(4);
      tft.setTextColor(RED);
      tft.println("Krivi");
      tft.setCursor(86, 140);
      tft.println("PIN");
      delay(2500);
      setup();
    }
    }
    }

}
```

LITERATURA

[1] dr.sc. Tonko Kovačević, PROGRAMIRANJE, Repetitorij s laboratorijskim vježbama Sveučilište u Splitu, Sveučilišni odjel za stručne studije, travanj 2017.

[2] dr.sc. Tonko Kovačević, Joško Smolčić struč.spec.ing.el,: Programiranje u C++ VJEŽBE, Programiranje u C++ na arduino platformi, Sveučilište u Splitu, Sveučilišni odjel za stručne studije, studeni 2017.

[3] Guido Ottaviani: GSM remote control Software-GSM_Control: https://guiott.com/GSMcontrol/GSMcontrol2.htm

[4] Geeetech Wiki: https://www.geeetech.com/wiki/index.php/Arduino_GPRS_Shield

[5] Smolčić Joško.: Razvoj platforme za upravljanje sustavom autonomnih tuševa, Diplomski rad, Sveučilište u Splitu, Sveučilišni odjel za stručne studije, srpanj 2013.

[6] SIMCom, A Company of SIM Tech: SIM900AT Commands Manual_V1.11, Release 12.01.2015.

[7] https://www.arduinocircuit.com/how-to-use-xkc-y25-v-liquid-sensor-with-ardui

[8] MarcoMartines / GSM/GPRS & GPS Shield Library for modules using SIM900/SIM908: https://github.com/MarcoMartines/GSM-GPRS-GPS-Shield

[9] Rui Santos, RANDOM NERD TUTORIALS, https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/

[10] https://www.arduinoecia.com.br/como-usar-sensor-de-liquido-xkc-y25-arduino/

[11] https://electropeak.com/learn/interfacing-plastic-liquid-level-float-switch-with-arduino/
[12] https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/
[13] https://arduinogetstarted.com/tutorials/arduino-relay
[14] https://www.circuitgeeks.com/arduino-temperature-sensor-thermistor/
[15] https://www.electronics-tutorials.ws/power/solid-state-relay.html
[16] Joško Smolčić, Tonko Kovačević, Siniša Zorica, Sandra Antunović Terzić, Development of a comparable GSM control device using open source technology, CIET 2020,
[17] http://www.lcdwiki.com/res/MAR2406/2.4inch_Arduino_8BIT_Module_MAR2406_User_Manual_EN.pdf