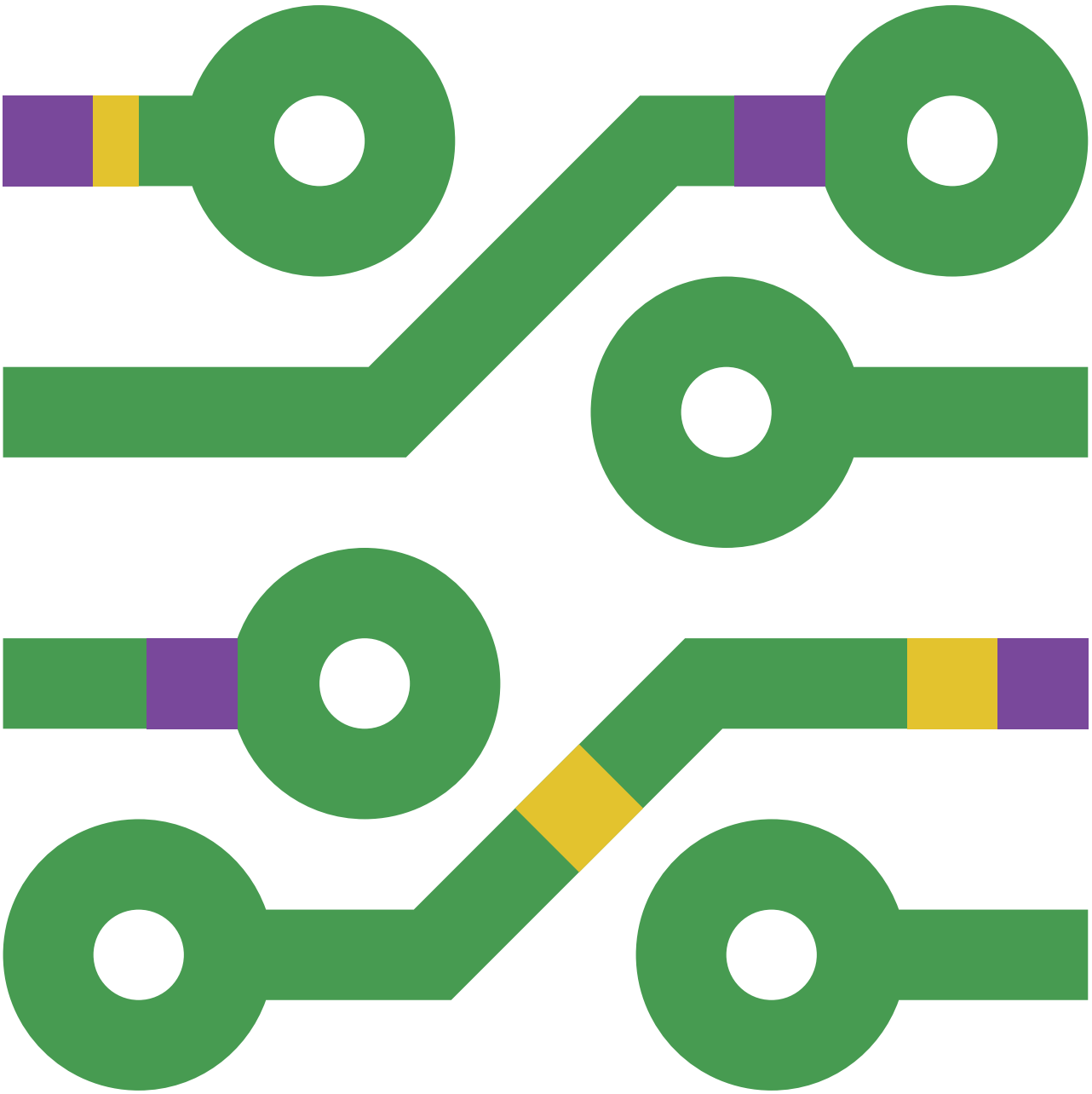


ARDUINO ZA SVE

PRIRUČNIK ZA NASTAVNIKE
Osnovna škola

Mr. sc. Muamer Halilović,
dipl. ing. el.

Podržano od:

< IT Girls >

UJEDINJENE NACIJE
BOSNA I HERCEGOVINA

ARDUINO ZA SVE

PRIRUČNIK ZA NASTAVNIKE
Osnovna škola

Mr. sc. Muamer Halilović, dipl. ing. el.

2019.

Sadržaj

Uvod

Ok, vjerujte, ni meni nije lako početi. Ali kao i svako putovanje i ovo mora nekako krenuti. Prvo što moram napisati jeste da je važno shvatiti da je ovaj priručnik pisan jednostavnim rječnikom, te da su neke stvari pojednostavljene da bi ih sudionici lakše razumjeli. Naravno, ohrabrujem sve one koji žele više istraživati i otići puno dalje od ovog priručnika. Vjerujem da mnogi pojmovi koji će se ovdje spominjati i nisu tako nepoznati. Cilj je ovo učiniti razumljivim, stvari pojednostaviti na taj način da znanje koje se kroz ovaj priručnik stekne bude dovoljno da čitatelj dobije samopouzdanje da samostalno nastavi istraživati svijet mikroračunala.

Naši se životi vrte oko megabajta, megaherca, čipova, procesora, AI, IoT, robota, ma jednostavno čitava zbrka pojmova i, priznajete, nekada se pravimo da ih razumijemo, a nekada pomislimo kako bi bilo baš dobro koristiti i stvarati, jednostavno biti u tom svijetu, neki bi rekli shvatiti matricu (op.a. MATRIX).

Dok pišem ovo, namjerno koristim pojam stvarati, jer umjetnici stvaraju i kreiraju, zar ne? A programeri su, htjeli mi to priznati ili ne, umjetnici novoga doba.

Odmah moramo napraviti razliku između klasičnih programera i *embedded* programera. *Embedded* ili programeri mikrokontrolerskih ili SoC sustava su nešto kao Ferrari u automobilskoj industriji, priča za sebe, pa ako mogu birati, biram Ferrari.

Projekt Arduino počeo je 2003. godine kao program za studente na Interaction Design Institute Ivrea (Institutu za dizajn interakcije Ivrea) u Ivrei, Italija. Cilj je bio kreirati jeftinu i jednostavnu platformu za izučavanje i kreiranje mikrokontrolerskih sustava kako za početnike tako i za profesionalce. Platforma je osmišljena tako da na jednostavan način omogući kreiranje i testiranje prototipova uređaja koji imaju interakciju s okolinom koristeći senzore i aktuatore.

Broj prodanih originalnih Arduino platformi je milijunski, a broj onih kloniranih vjerojatno nikada nećemo ni saznati. Dakle, konačno smo u prilici da uđemo u jedan novi svijet u kojemu je bukvalno samo nebo granica.

Mr. sc. Muamer Halilović, dipl. ing. el.

Arduino – Main chip

Prije no što uđemo u svijet mikrokontrolera, moramo napraviti usporedbu između „mozga" platforme Arduino i klasičnoga CPU-a iz stolnog ili prijenosnog računala.

Platforma Arduino izrađena je oko kontrolera ATMEL ATMEGA 328p, čija su svojstva sljedeća:

- 28 nožica
- Napajanje od 3 do 5 volti
- 0,1 wat
- Radi na 16 MHz
- 32 KB HDD – flash memorije
- 2 KB RAM memorije
- Cijena: oko 2 $

Slika 1. ATMEGA 328p na platformi Arduino i izvan nje

Moderni PC vrlo vjerojatno ima CPU 6, 7. ili 8. generacije, a za usporedbu uzet ćemo i5-6400.

Slika 2. CPU i5 6. generacije

- 1151 nožica
- Napajanje 1,35 volti
- 35 wati
- Radi na 2,8 GHz
- Nema flash memoriju, ali većina računala danas ima HDD s minimalno 250 GB
- Nema interni RAM, ali većina računala danas ima minimalno 4 RAM-a
- Cijena: oko 150 $

Naravno da je CPU daleko „moćniji" uređaj, ali zamislite samo uređaj za mjerenje tjelesne temperature veličine slim-fit desktop računala, koji košta nekoliko stotina konvertibilnih maraka. Jednostavno nema smisla.

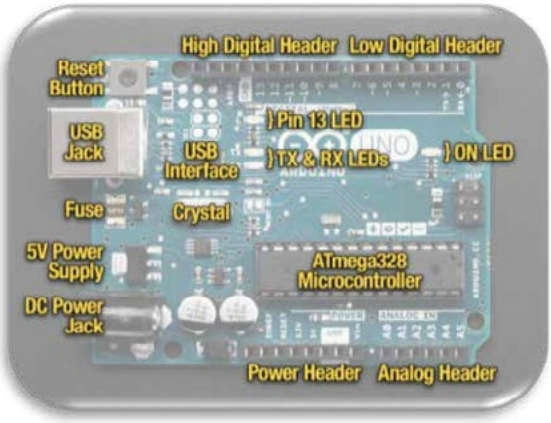
Mora se naglasiti još jedna osnovna razlika između PC računara i mikroračunala, a to je da na PC-u nema ograničenja za instalaciju aplikacija. Kad nestane HDD prostora, napravi se *upgrade* i doda novi HDD, dok kod mikroračunala imamo samo jednu aplikaciju, a to je ona za koju je mikroračunarski sustav namijenjen.

Platforma Arduino

Slika 3. Golf MKII

Prosječan Bosanac i Hercegovac vjerojatno bi bez problema mogao navesti sve dijelove omiljenoga VW automobila GOLF MKII, poznatijeg na našim prostorima kao „Golf 2".

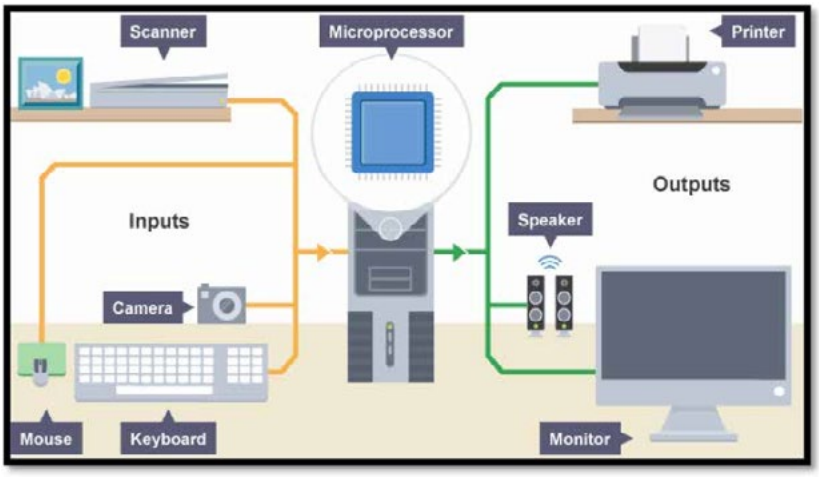
Iako je „Golf 2" proizveden u višestruko manjoj seriji od platforme Arduino, vjerojatno na prste ruke možemo pobrojati naše prosječne sugrađane koji bi isto to mogli napraviti i za platformu Arduino, pa idemo to malo promijeniti.



Slika 4. Platforma Arduino

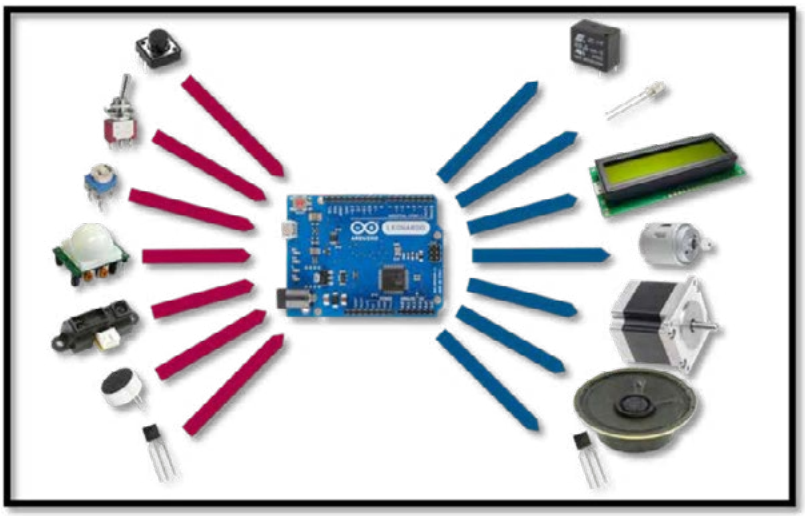
- *USB Jack* – služi za spajanje Arduino mikrokontrolera s računalom, kao sučelje za programiranje i napajanje prilikom testiranja;
- *Reset Button* – resetira mikrokontroler, odnosno aplikaciju učitanu u njega;
- *Fuse* – osigurač, morao vam je barem jednom doma iskočiti jedan, štiti uređaje od strujnih preopterećenja;
- *5V Power Supply* – naponski regulator koji limitira napon s DC konektora na 5 V;
- *DC Power Jack* – eksterno napajanje od 7 do 20 V DC;
- *Power Header* – sabirnica za napajanje senzora ili aktuatora;
- *Analog Header* – sabirnica za analogne senzore;
- *ON Led* – signalizacija da je platforma pod naponom;
- *Low & High Digital Header* – ulazi i izlazi za senzore i aktuatore;
- *Pin 13 LED* – svaka mikrokontrolerska razvojna platforma ima bar jednu LE diodu spojenu na pin mikrokontrolera, da bi se platforma mogla testirati;
- *USB Interface* – sučelje koje služi za komunikaciju mikrokontrolera i PC-a;
- *TX & RX LED* – diode koje nam omogućavaju da vidimo da postoji protok podataka – signala između PC-a i mikrokontrolera i obratno;
- *Crystal* – električni oscilator koji generira signal točne frekvencije – clock mikroračunala.

Senzori i aktuatori



Slika 5. I/O (Ulazni/izlazni) uređaji kod klasičnog PC-ja (računala)

Mi svakodnevno imamo interakciju sa senzorima i aktuatorima, možda je jednostavnije kada kažemo ulaznim i izlaznim uređajima. Ponovno ćemo se vratiti na PC. Kao što je to prikazano na slici iznad, ulazni uređaji su miš, skener, tipkovnica, kamera, dok su izlazni uređaji ekran, printer ili zvučnik.



Slika 6. I/O uređaji kod razvojne platforme Arduino

Ulazni uređaji u mikrokontrolerskim sustavima su tasteri, prekidači, potenciometri, digitalni senzori ili analogni senzori. Izlazni uređaji bili bi releji, LED, LCD, motori ili zvučnici. Mikrokontrolerski sustav prati promjene stanja na svojim ulazima i spram aplikacije radi promjene na svojim izlazima. Kako budemo prolazili kroz praktične primjere, malo detaljnije ćemo se upoznati sa svakom od komponenti koje ćemo koristiti.

IDE Software

Prije svega prvo moramo podesiti svoje računalo na taj način da odradimo instalaciju softvera Arduino IDE. Naravno, riječ je o besplatnoj aplikaciji koju koristimo da bismo kreirali ili razmjenjivali informacije s mikrokontrolerom Arduino. Aplikacija se može preuzeti sa sljedeće mrežne lokacije:

https://www.Arduino.cc/en/Main/Software

Slika 7. Arduino IDE odabir instalacijskoga dosjea

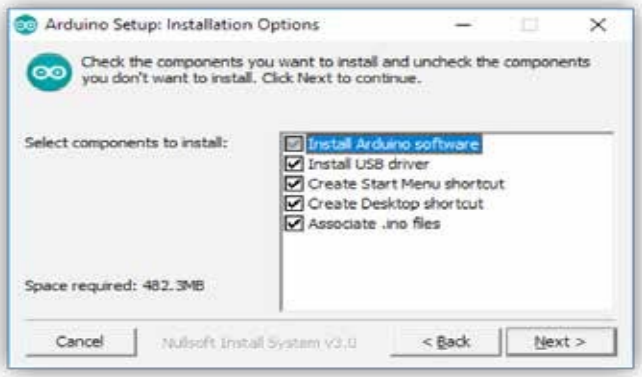
Softver Arduino stalno se revidira. U trenutku pisanja ovog priručnika aktualna je inačica 1.8.9, ali je vrlo moguće da dok ovaj priručnik dođe do krajnjih korisnika, novija inačica softvera bude dostupna za instalaciju.

Proći ćemo kroz korake za podešavanje i instalaciju Arduino IDE Softwarea. Kliknite na poveznicu Windows installer za preuzimanje instalacijskoga dosjea i potom kliknite na novoj stranici Just download.

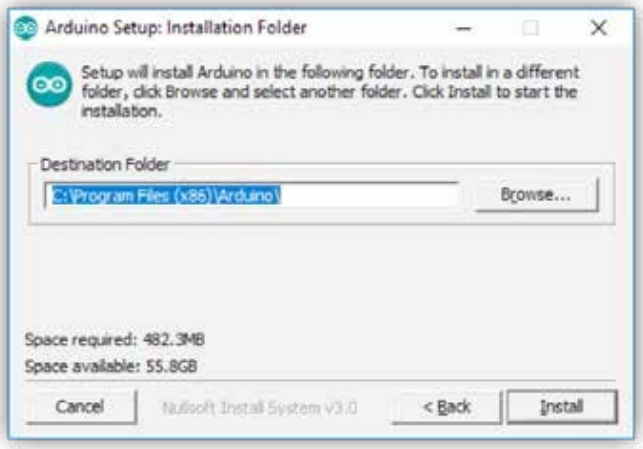
Morate potvrditi da se slažete s općim uvjetima pri uporabi softvera koji koristi tzv. licencu open source.

Slika 8. Arduino IDE potvrda općeg ugovora za licencu open source

Odaberite instalaciju USB drajvera, te kreiranje ikona i shortcuta, a potom potvrdite lokaciju za instaliranje aplikacije.

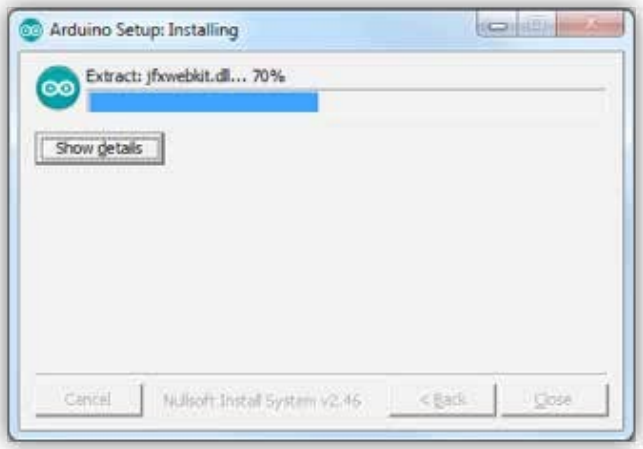


Slika 9. Arduino IDE odabiri instalacije USB drajvera



Slika 10. Arduino IDE potvrda lokacije instalacije

Proces instalacije u prosjeku traje oko minute, što opet zavisi od hardverske konfiguracije računala na koji se aplikacija instalira.



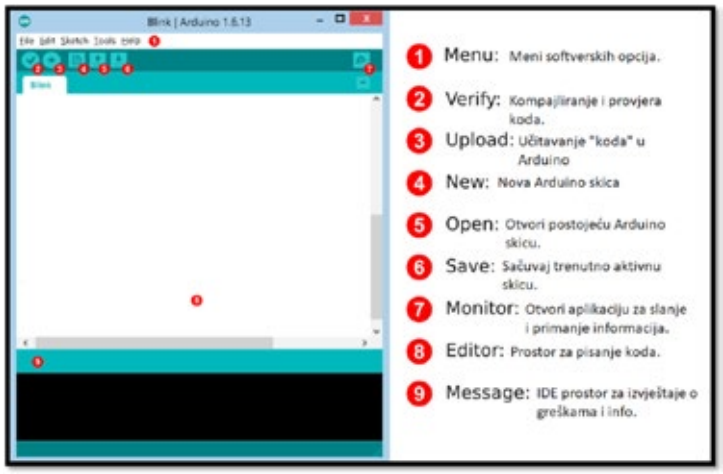
Slika 11. Proces instalacije Arduino IDE

Konačno, još nekoliko sitnih prepreka i spremni smo za naše prve linije koda. Ikona Arduino IDE-a je na zaslonu. Dakle, sve smo spremniji.

Slika 12. Ikona Arduino IDE

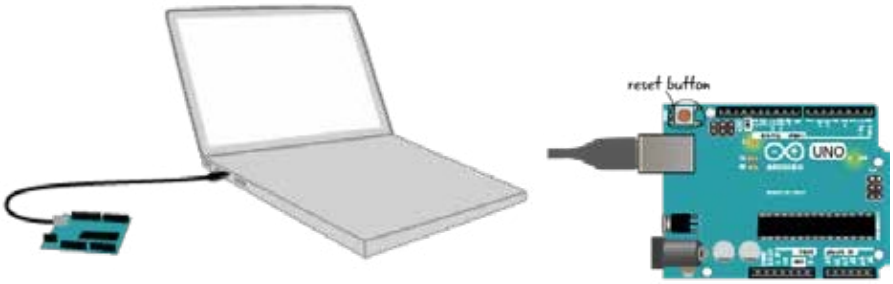
Arduino IDE okruženje

Prije no što krenemo u programiranje, pojasnit ćemo Arduino IDE okruženje.



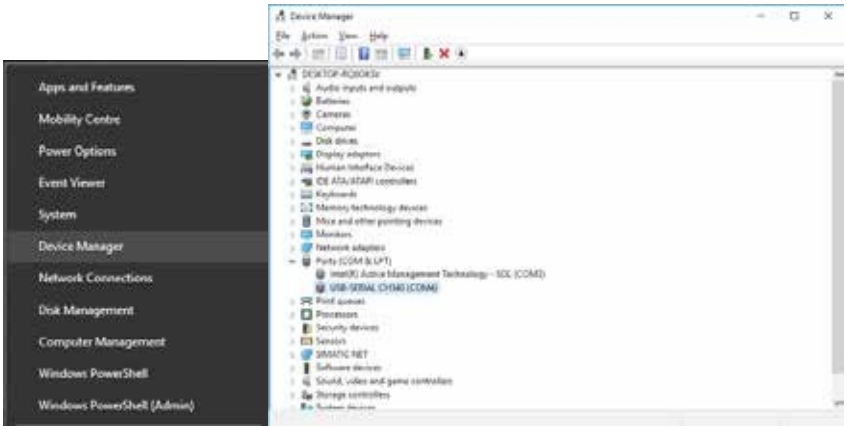
Slika 13. Arduino IDE opis

Prvo je potrebno, koristeći USB A – USB B kabel, spojiti Arduino Uno na slobodni USB port PC.



Slika 14. Spajanje Arduina na PC

Dobra je rutina u *Windows Device Manageru* provjeriti kojem COM (komunikacijskom sučelju) portu je pridružen Arduino. Na Win10 operativnom sustavu potrebno je desnim klikom na Windows ikoni naći opciju *Device Manager* i pretražiti komunikacijske portove (Ports – COM & LPT).



Slika 15. Arduino na COM4

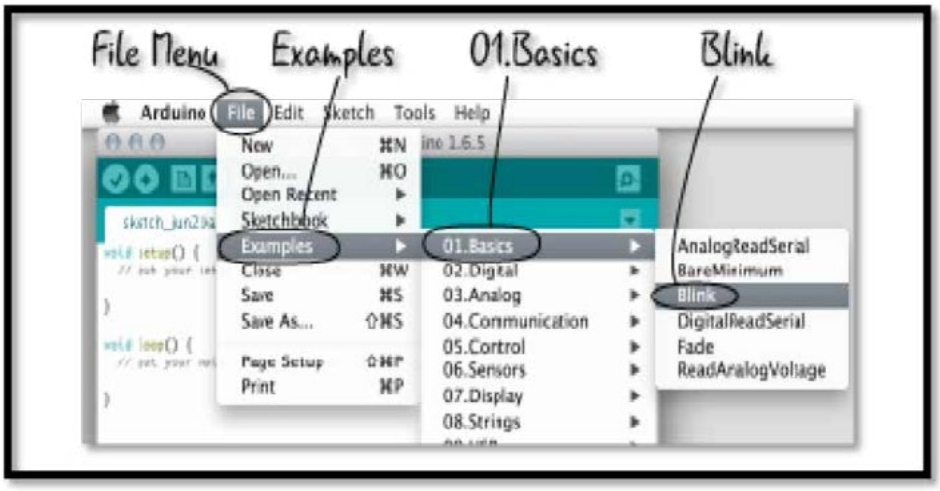
cout << „Hello, World!”;

„Hello, World!” ili „Zdravo, svijete!” je program u C++ programskom jeziku. U svijetu mikroračunarskog programiranja to je aplikacija „blink LE diode” na razvojnoj ploči. Kako ćemo u narednoj vježbi vidjeti, na PIN 13 platforme Arduino bit će spojena LE dioda, što će značiti da ta aplikacija ustvari mijenja stanje LE diode na PIN-u 13 iz uključenog u isključeno stanje.

Mora se naglasiti da je programski jezik koji se koristi u Arduino IDE pojednostavljeni C programski jezik, i smatra se da je takav način ulaska u svijet programiranja, gdje korisnik vidi rezultat svog koda u fizičkom svijetu, idealan za početnike.

Naravno, osnovnu sintaksu programskog jezika učit ćemo kroz primjere. Ne treba biti u zabludi da će ovaj „hands-on” priručnik biti dovoljan da korisnik savlada sve tehnike programiranja, ali će biti dovoljan da samostalno riješi zadatke koji se pred njega postavljaju.

Sljedeći korak bio bi otvaranje aplikacij*e example Blink,* kompajliranje i njeno učitavanje u Arduino, te detaljna analiza koda.



Slika 16. Otvaranje skice Arduino Blink

```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */


int ledPin = 13;                // LED spojena na digital pin 13

void setup()                    // jednom se pokreće, kada se skica pokrene
{
  pinMode(ledPin, OUTPUT);      // proglašavanje pina 13 IZLAZOM
}


void loop()                     // stalno se izvršava
{
  digitalWrite(ledPin, HIGH);   // uključi LED
  delay(1000);                  // čekaj sekundu
  digitalWrite(ledPin, LOW);    // isključi LED
  delay(1000);                  // čekaj sekundu
}
```

Pa krenimo redom. **Komentari** su dijelovi koda koje dobar programer uvijek ostavlja radi sebe ili drugih koji će čitati kod, te ga na taj način brže i lakše razumjeti. Arduino komentare tretira baš kao što je to i rečeno, kao komentare, dakle ne uzima ih u obzir prilikom izvršavanja koda.

Komentari se pojavljuju u dva oblika:

```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

//Komentar
/*
Komentar
*/
```

Dalje ćemo vidjeti jednu klasičnu **izjavu** u C programskom jeziku. Poslije izjave slijedi komentar koji nam je u potpunosti jasan, počinje velikim slovom, ima smisao i završava točkom. S druge strane, izjava na lijevoj strani nije ništa drugo do programska rečenica koja kaže da je ***varijabla*** imena ledPin tip podatka ***int*** (integer – „moderni naziv za cijeli broj"), kojoj je dodijeljena vrijednost 13.

```
int ledPin = 13;                    // LED spojena na digital pin 13.
```

Pa ovaj pojednostavljeni C programski jezik i nije tako strašan. Čisto radi vježbe, napišite dvije izjave koje vas opisuju.

```
                                    / moja je visina____cm
                                    / moja je težina_____kg
```

Sada postaje zanimljivo. Svaki mikrokontrolerski uređaj mora imati dvije defaultne specijalne procedure **„setup()"** i **„loop()".** Imena su im intuitivna i lako se da naslutiti njihova uloga.

```
void setup()                        // jednom se pokreće, kad se skica starta
starts
{
  pinMode(ledPin, OUTPUT);          // proglašavamo pin 13 IZLAZOM
}
```

Naime, mikrokontroler na razvojnoj platformi Arduino ima 13 pinova koji su deklarirani kao *Digital inputs/ outputs*. Ispravno ih je nazvati GPIO (*general purpose input output* – ili ulazi i izlazi opće namjene). U proceduri setup mi kontroleru damo uputu da ćemo neke pinove koristiti kao INPUTE, a neke kao OUTPUTE. Taj dio koda izvršava se samo jednom, kada se platforma stavi pod napon.

U našem konkretnom slučaju, kako je na pin 13 spojena LE dioda (koja spada u grupu izlaznih elemenata), logično je da pozovemo **funkciju pinMode,** koja proglašava pin 13 izlaznim pinom. Dakle, funkcija pinMode ima dva argumenta koja korisnik treba podesiti, a to su koji **pin** želimo koristiti i u kojem **modu**. I da ponovimo, ne smijemo zaboraviti staviti točku na kraju programske izjave.

Sada malo vježbe. Pogledajte proceduru setup u kojoj će se pin 2 proglasiti ulaznim pinom, a pin 8 izlaznim.

```
pinMode(2, INPUT);          / proglašavamo pin2 ULAZNIM pinom
pinMode(8, OUTPUT);         / proglašavamo pin8 IZLAZNIM pinom
```

```
void loop()                        // stalno se „vrti“
{
  digitalWrite(ledPin, HIGH);      // uključi LED na pinu 13
  delay(1000);                     // zadrži stanje 1 sekundu
  digitalWrite(ledPin, LOW);       // isključi LED na pinu 13
  delay(1000);                     // zadrži stanje 1 sekundu
}
```

Procedura **loop()** ciklično se izvršava i ustvari predstavlja aplikaciju koju korisnik u konačnici vidi. U našem primjeru unutar procedure loop koristimo nove dvije funkcije: **digitalWrite()** i **delay().** Funkcija DigitalWrite ima dva argumenta: ***pin*** i ***value***. Dakle u našem primjeru dajemo uputu kontroleru da LE diodu na pinu 13 uključi ili isključi.

Funkcija *delay* zaustavlja izvršavanje programa za definirani broj milisekundi.

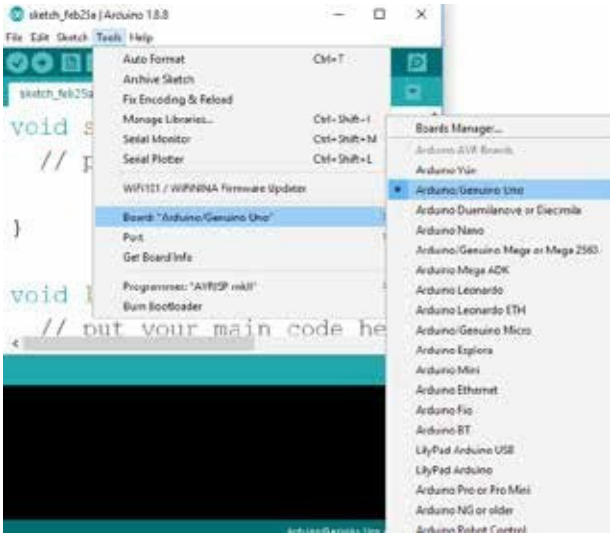
Da ne zaboravimo, procedure su skupovi programskih izjava. Da biste bolje razumjeli procedure, pogledajte sljedeći primjer.

```
clean cat wash the cat(dirty cat)   // procedure za pranje prljave mačke
{
      Nađite mačku.
      Uhvatite je.
      Odvrnite česmu.
      Stavite mačku ispod česme.
      Dobro operite mačku.         // dok ne bude čista.
      Pustite mačku da nastavi bezbrižan život.
}
```

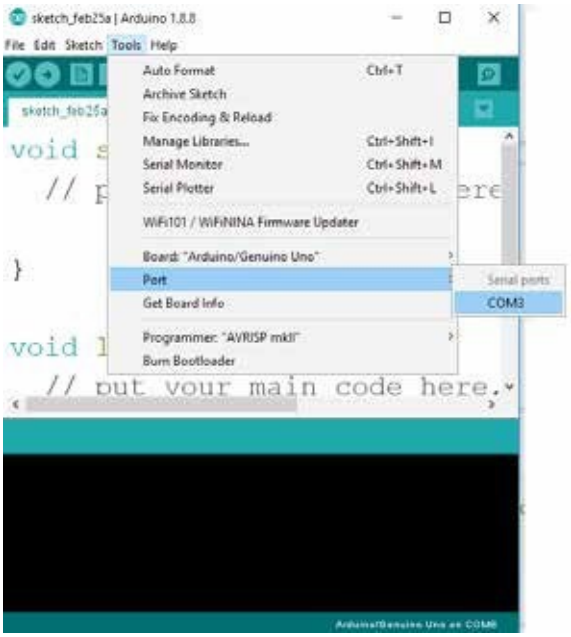
Ime procedure je *operite mačku.* Procedura ima jedan argument – prljavu mačku, koja nakon uspješno izvršenih programskih izjava vraća na kraju procedure čistu mačku. Jednostavno, zar ne!?

Verifikacija i učitavanje koda

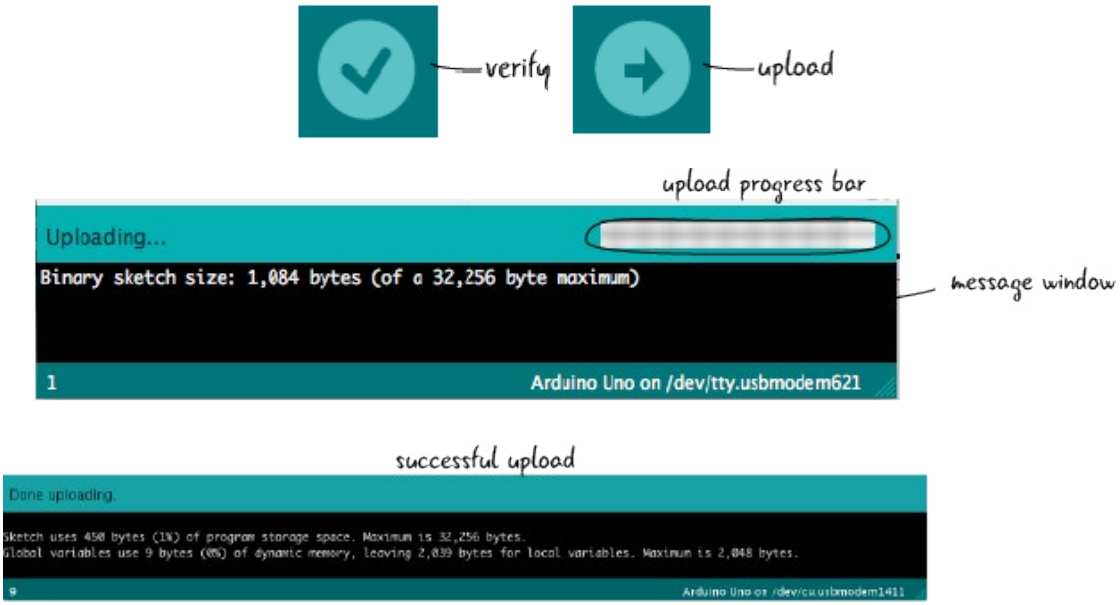
Prije same procedure učitavanja naše aplikacije u mikrokontroler potrebno je u izborniku *Tools* odabrati odgovarajuću razvojnu pločicu i pripadajući COM port.



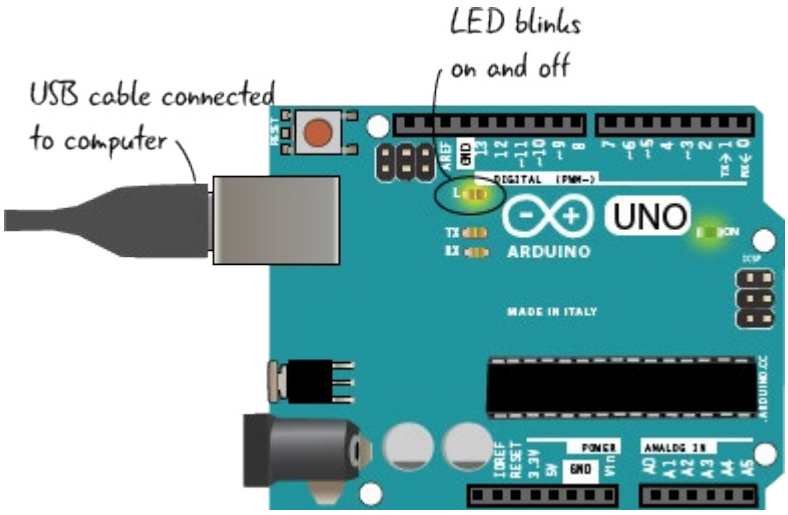
Slika 17. Odabir razvojne ploče Arduino



Slika 18. Odabir pripadajućeg porta



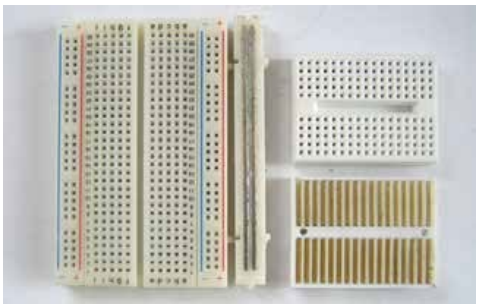
Slika 19. Procedura za verifikaciju i učitavanje koda



Slika 20. LE dioda na pinu 13 se uključuje i isključuje

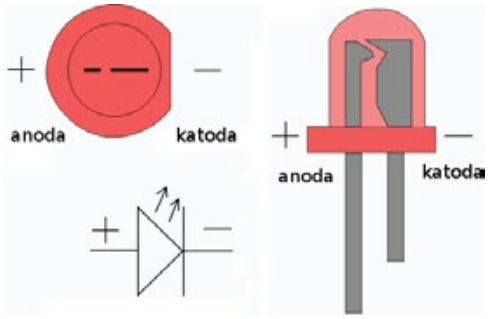
Osnovne elektronske komponente i pomagala

U narednim koracima upoznat ćemo se s osnovnim elektronskim komponentama i pomagalima, te pokušati samostalno kreirati malo kompleksnije aplikacije. Za realizaciju vježbe potrebna nam je ploča matador. Na slici ispod prikazana je organizacija ploče matador. Dakle, uzdužne sabirnice označene sa „+" i „-" koriste se za distribuciju napona, a horizontalne služe za spajanje komponenti.



Slika 21. Ploča matador (pogled odozgo i odozdo)

Naredni element koji ćemo koristiti da bismo realizirali vježbu je LE dioda ili svijetleća dioda, a to je specijalna vrsta diode koja uslijed protjecanja struje nju pretvara u svjetlost.



Slika 22. LE dioda – izgled i simbol

Otpornik je neophodan da bi se ograničila jačina struje kroz LED. Preporučena jačina struje u slučaju 5 mm crvene LED je oko 20 mA. Kada je izlaz mikrokontrolera aktivan, on na svom izlazu daje 5 V pa bismo po Omovom zakonu dobili sljedeću preporučenu vrijednost otpornika:

$$II = \frac{\#}{\$} \rightarrow RR = \frac{\#}{\cdot} = \frac{((*)}{,-(-/)} = \frac{((*)}{-.-,(/)} = 250\ (\Omega)$$

Naravno, uvijek je dobra praksa osigurati elektronsku opremu pa je odabrana vrijednost otpora 330 Ω.

ISHODI UČENJA

(vježbe 1 do 5)

ISHODI UČENJA	RAZINA POSTIGNUĆA
- Učenik/-ica koristi programsko okruženje za Arduino - Opisuje ulogu osnovnih dijelova na Arduinu - Razlikuje osnovne elektronske komponente (ploču matador, otpornike, LE diode) koje su korištene u vježbama - Učenik/-ica prepoznaje i opisuje ulogu elektronskih komponenti - Učenik/-ica primjenjuje svoje znanje iz elektronike za povezivanje Arduina i elektronskih komponenti - Učenik/-ica koristi električnu shemu spoja za realizaciju vježbe - Raspoznaje razlike između osnovnih algoritamskih struktura koje su korištene unutar vježbi - Povezuje Arduino s pločom matador i ostalim komponentama - Verificira i izvršava program	**Minimalna postignuća** Učenik/-ica imenuje sve potrebne komponente korištene u vježbi Učenik/-ica s pomoću nastavnika povezuje dijelove s Arduinom, verificira i izvršava program **Dovoljna postignuća** Učenik/-ica objašnjava ulogu osnovnih elektronskih komponenti korištenih u vježbi Povezuje osnovne elektronske komponente prema shemi spoja Učenik/-ica posjeduje osnovna znanja o pisanju programa za prethodne vježbe Samostalno verificira i izvršava program **Visoka postignuća** Učenik/-ica samostalno povezuje komponente (Arduino, ploču matador, otpornike, LE diode) korištene u vježbama prema shemi spoja Raspoznaje razlike između osnovnih algoritamskih struktura, te primjenjuje svoje znanje za pisanje koda Samostalno verificira i izvršava program, te po potrebi uspješno otklanja nastali bug

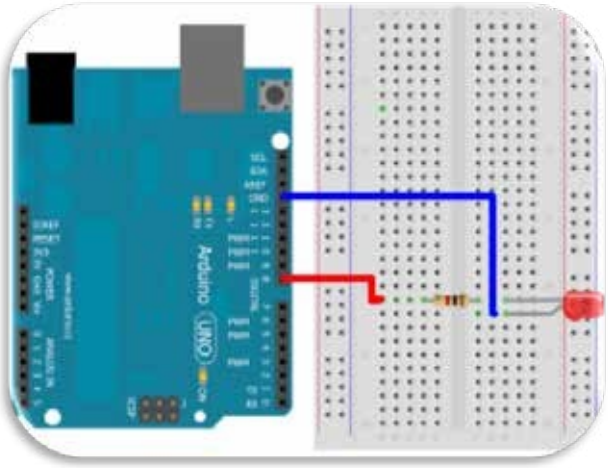
Vježba 1.
NEKA BUDE SVJETLO

Spojiti LE diodu na digitalni izlaz Arduino UNO pločice, te je paliti i gasiti u razmaku od 2 sekunde.

Potrebne komponente:

- Pločica Arduino UNO i USB 1 kom.
- Pločica matador 1 kom.
- LE diode 1 kom.
- Otpornik 220 Ω 1 kom

Slika 23. Potrebne komponente



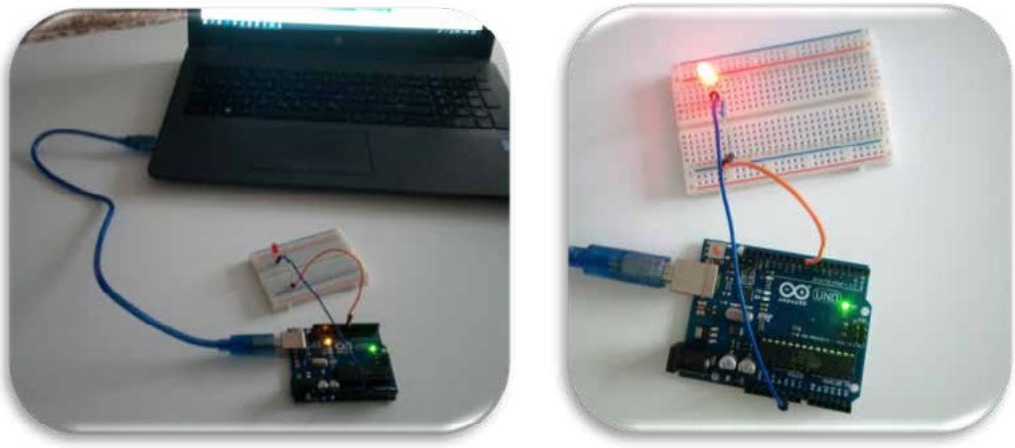
Slika 24. Shema spoja

4Koraci za realizaciju vježbe:

1. Povežite Arduino pin 8 s LE diodom na osnovu priložene sheme spoja.
2. Kreirajte *Sketch* kojim ćete upravljati radom LE diode na pinu 8:

Kod:

```
int izlaz = 8;
void setup()
{
  // put your setup code here, to run once:
  pinMode(izlaz, OUTPUT);
}
void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(izlaz, HIGH);
  delay(2000);
  digitalWrite(izlaz, LOW);
  delay(2000);
}
```



Slika 25. Verifikacija i testiranje

Primjer 1.

Modificirajte kod da LED bude uključena na 100 msek., a isključena na 900 msek.

Primjer 2.

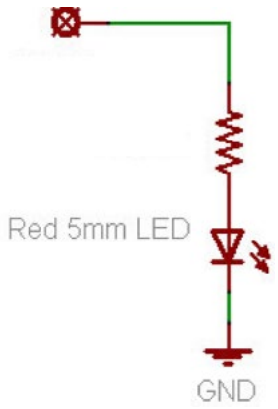
Modificirajte kod da LE dioda bude uključena na 50 msek. i isključena na 50 msek. Opišite kako se ponaša LE dioda.

Dobijemo na LE diodi tzv. učinak STROBE, odnosno „titranje”!

Primjer 3.

Modificirajte kod na taj način da LE dioda bude uključena i isključena u intervalima od po 10 msek. Opišite pojavu. Pokušajte mahati razvojnom platformom Arduino naprijed–nazad u zamračenoj prostoriji. Što se dešava?

LE diode ostavljaju trag u zraku. Na taj način oko je „prevareno" i ne može vidjeti promjenu stanja na LE diodi. Mahanjem dobijemo učinak linije u zraku.

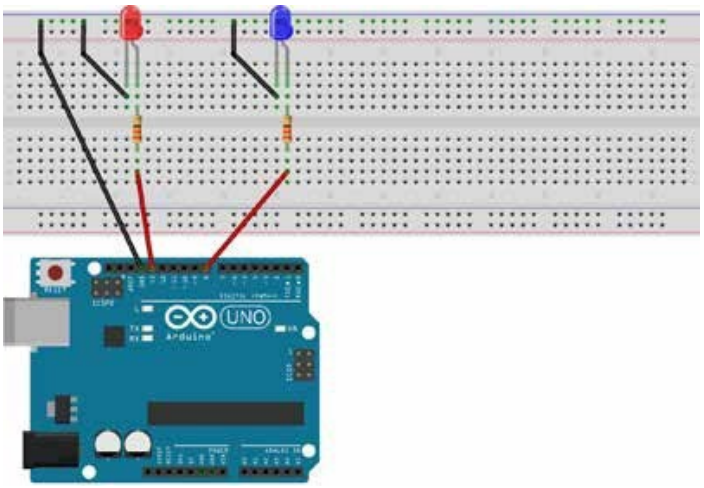


Slika 26. Električna shema kola za spajanje LE dioda na pin 8

Vježba 2.
TOPLO I HLADNO!

Kreirajte aplikaciju koja će naizmjenično paliti i gasiti LE diodu na pinu 13 i pinu 8. Potrebne komponente:

- Pločica Arduino UNO i USB 1 kom.
- Pločica matador 1 kom.
- LE diode 2 kom.
- Otpornik 220 Ω 2 kom.



Slika 27. Shema spoja

Kod:

```
int led1 = 13;                  // LED spojena na digital pin 13
int led2 = 8;                 // LED spojena na digital pin 8

void setup()
{
  pinMode(led1, OUTPUT);      // proglašavanje pina 13 IZLAZOM
  pinMode(led2, OUTPUT);      // proglašavanje pina 8 IZLAZOM

}

void loop()                     // stalno se izvršava
{
  digitalWrite(led1, HIGH);     // uključi LED1
  digitalWrite(led2, LOW);      // isključi LED2

  delay(1000);                  // čekaj sekundu
  digitalWrite(led1, LOW);      // isključi LED1
  digitalWrite(led2, HIGH);     // uključi LED2
  delay(1000);                  // čekaj sekundu
}
```

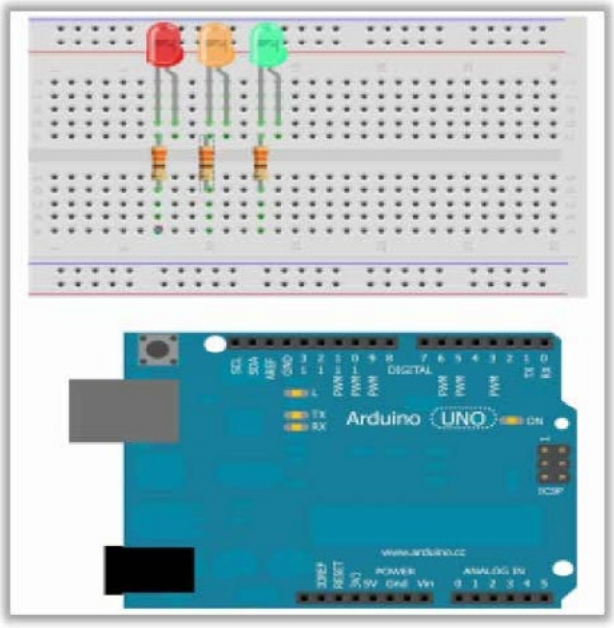
Vježba 3.
SEMAFOR

Kreirati *Sketch* kojim ćete simulirati rad semafora. Neka crveno svijetli 5 sek., zatim crveno i žuto 3 sek., a na kraju zeleno 8 sek. i žuto 3 sek.

Potrebne komponente za vježbu i shema spoja:

- Arduino Uno 1 kom.
- LE dioda 3 kom.
- Otpornik 330 Ω 3 kom.

Slika 28. Primjer izgleda semafora



Slika 29. Shema spoja

Kod:

```
void setup()
 {
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
 digitalWrite(13,1);        //crveno
 digitalWrite(12,0);
 digitalWrite(11,0);
 delay(3000);                     //3 sekunde

 digitalWrite(12,1);        //crveno i žuto
 delay(3000);

 digitalWrite(13,0);
 digitalWrite(12,0);
 digitalWrite(11,1);        //zeleno
 delay(8000);

 digitalWrite(12,1);              //žuto
 digitalWrite(11,0);
 delay(3000);
 }
```

Vježba 4.
DISKO SVJETLA

Disco lights – 5 lampica i 4 učinka

Kreirati *Sketch* za kontrolu rada LE diode. Ovaj se zadatak može riješiti na više načina. Za prvi treba jednostavno primijeniti logiku koju smo do sada primjenjivali i slijedno redati programske iskaze spram željene logike.

```
        digitalWrite(led1, HIGH);
          delay(500);
        digitalWrite(led2, HIGH);
          delay(500);
        digitalWrite(led3, HIGH);
          delay(500);
        digitalWrite(led4, HIGH);
          delay(500);
        digitalWrite(led5, HIGH);
          delay(500);
```

Na taj način dobit ćemo učinak da se LED pale jedna iza druge sa zadrškom od pola sekunde. Međutim, kako budemo razvijali nove učinke, to će se broj linija koda drastično povećavati.

Da bismo to izbjegli, možemo napraviti sljedeće: LED spojimo u niz na sljedeći način:

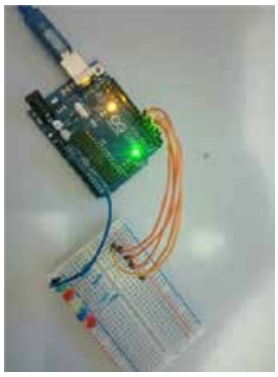
LED 1 – pin 2

LED 2 – pin 3

LED 3 – pin 4

LED 4 – pin 5 `

LED 5 – pin 6



Slika 30. LED spojene u niz

Dakle, kako imamo niz, možemo iskoristiti i ***for petlju*** koja se upotrebljava za ponavljanje programskih izjava unutar vitičaste zagrade, pri čemu se koriste inkrementalni ili dekrementalni brojači za prolazak od početnog do krajnjeg uvjeta za ponavljanje for petlje.

Zaglavlje *for* petlje ima tri argumenta.

```
for (inicijalizacija; uvjet; inkrement)
{
Programske_izjave(inkrement);
}
```

Kod iz posljednjeg primjera s for petljom izgledao bi ovako:

```
for(int i=2;i<=6;i++)
{
    digitalWrite(i, HIGH);
    delay(500);
}
```

Prevedeno, za vrijednost broja i = 2 do broja i = 6 s inkrementom od 1, izvrši programsku izjavu sa zadrškom od pola sekunde.

Za i = 2 izvršit će se `digitalWrite(2, HIGH);`
Za i = 3 izvršit će se `digitalWrite(3, HIGH);`
Za i = 4 izvršit će se `digitalWrite(4, HIGH);`
Za i = 5 izvršit će se `digitalWrite(5, HIGH);`
Za i = 6 izvršit će se `digitalWrite(6, HIGH);`

Pri čemu je naredba za inkrement

i++;

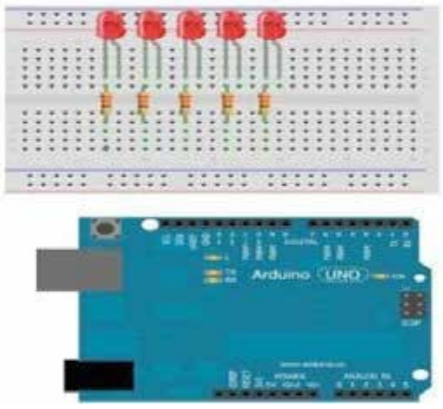
samo kraće napisan matematički izraz

i= i + 1;

Pinove 0 i 1 trebamo izbjegavati koristiti jer te pinove označene kao RX i TX koristi i sučelje za programiranje.

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. LE dioda 5 kom.
3. Otpornik 330 Ω 5 kom.



Slika 31. Shema spoja

Koraci za realizaciju vježbe:

1. Kreirajte shemu spoja koristeći prethodno stečena znanja.
2. Kreirajte Sketch kojim ćete upravljati radom LE diode kreirajući sljedeće učinke:
   - Diode se uključe sa zadrškom od pola sekunde na pločicu matador u portove od 2 do 6,
   - Diode se gase sa zadrškom od pola sekunde na pločicu matador u portove od 2 do 6,

- Dioda se uključi, zadrži stanje pola sekunde, ugasi se pa se proces ponovi i za ostale diode,
- Sve se diode uključe i zadrže stanje pola sekunde, isključe se te se proces ponovi 5 puta.

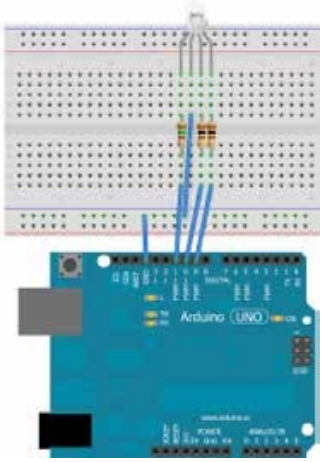
```
void setup()
 {
for(int i=2;i<=6;i++)
 {
      pinMode(i,OUTPUT); //pinovi od 2 do 6 izlazni
 }
 }
 void loop()
 {
 for(int i=2;i<=6;i++)      //učinak 1
 {
   digitalWrite(i, HIGH);
    delay(500);
 }
 for(int i=2;i<=6;i++)      //učinak 2
 {
    digitalWrite(i, LOW);
    delay(500);
 }
for(int i=2;i<=6;i++) //učinak 3
 {
    digitalWrite(i, HIGH);
    delay(500);
    digitalWrite(i, LOW);
 }
for(int i =0;i<=5;i++)      // učinak 4
 {
     for(int j=2;j<=6;j++)
         {
           digitalWrite(j,HIGH);
        }
    delay(500);
    for(int j=2;j<=6;j++)
         {
            digitalWrite(j,LOW);
         }
 }
}
```

Vježba 5.
RGB LED

Kreirati *Sketch* za upravljane radom RGB LED.

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. RGB LED 1 kom.
3. Otpornik 150 Ω 1 kom.
4. Otpornik 100 Ω 2 kom.



Slika 32. Shema spoja

Koraci za realizaciju vježbe:

1. Kreirajte testni sustav koristeći shemu spoja.
2. Kreirajte *Sketch* kojim ćete upravljati radom LE diode na taj način da se u jednakim vremenskim intervalima mijenjaju kombinacije boja.
3. Od papira napravite kućište te unutra ubacite zgužvanu maramicu i RGB LED.

Slika 33. Izrada kućišta za RGB LED

ISHODI UČENJA

(vježbe od 6 do 8)

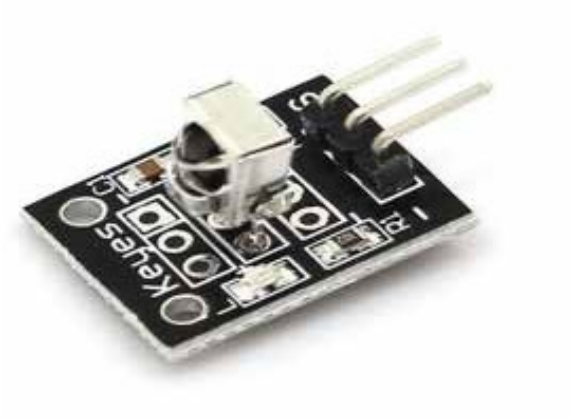
ISHODI UČENJA	RAZINA POSTIGNUĆA
- Učenik/-ica koristi programsko okruženje za Arduino - Razlikuje elektronske komponente koje su korištene u vježbama - Učenik/-ica opisuje ulogu elektronskih komponenti - Učenik/-ica definira svojstva signalnih uređaja (senzora) primijenjenih u vježbama - Učenik/-ica definira način rada sedam- segmentnog zaslona - Učenik/-ica primjenjuje svoje znanje iz elektronike za povezivanje Arduina i elektronskih komponenti - Učenik/-ica koristi električnu shemu spoja za realizaciju vježbe - Primjenjuje znanje iz programiranja za realizaciju određenog zadatka - Koristi odgovarajuće knjižnice unutar programa za realizaciju vježbe - Koristi shemu za povezivanje uređaja Arduino s pločicom matador i ostalim komponentama (senzori, sedam- segmentni zaslon, taster) - Verificira i izvršava program	**Minimalna postignuća** Učenik/-ica imenuje sve potrebne komponente korištene u vježbi S pomoću nastavnika/-ice povezuje dijelove s Arduinom, verificira i izvršava program **Dovoljna postignuća** Učenik/-ica objašnjava ulogu elektronskih komponenti korištenih u vježbi Povezuje elektronske komponente prema shemi spoja Učenik/-ica posjeduje osnovna znanja o pisanju programa za prethodne vježbe Koristi se odgovarajućim knjižnicama unutar programa Samostalno verificira i izvršava program **Visoka postignuća** Učenik/-ica samostalno povezuje komponente (Arduino, ploču matador, otpornike, LE diode, senzore, sedam-segmentni zaslon, tastere) korištene u vježbama prema shemi spoja Primjenjuje svoje znanje iz programiranja za pisanje koda Samostalno verificira i izvršava program, te po potrebi uspješno otklanja nastali bug

Vježba 6.
INFRACRVENI SENZOR I UPRAVLJAČ

Dešifrirati signale koje prima IR senzor pritiskom na dugmad upravljača te programirati Arduino UNO tako da se pritiskom na tastere 1, 2 i 3 pale odgovarajuće diode (zelena, crvena i žuta).

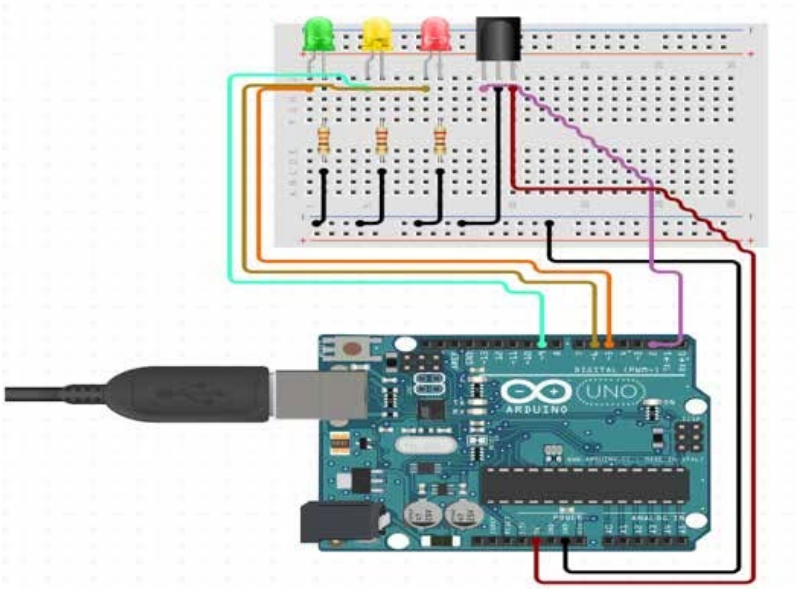
Potrebne komponente:

- Arduino UNO i USB
- Pločica matador
- Crvena, zelena, žuta LED s odgovarajućim otpornicima
- IR prijemnik
- Daljinski upravljač



Slika 34. IR receiver

* Obratiti pažnju na pinove IR senzora! Y – signal, G – ground, R – 5 V



Slika 35. Shema spoja

Za ovaj zadatak potrebno je koristiti knjižnicu *IRreceiver.h* , koja u sebi sadrži osnovne funkcije za korištenje IR senzora.

Najbitnije funkcije su:

Kreiranje instance IR senzora:
`IRrecv irrecv(broj pina na koji je priključen Y pin senzora);`

Deklariranje varijable u kojoj ćemo spremati primljeni signal od daljinskog upravljača:
`decode_results rezultat;`

Pokretanje IR senzora tj. pokretanje dekodiranja ulaznog signala:
`irrecv.enableIRIn();`

Provjera je li neki signal primljen:
`irrecv.decode(&rezultat)`

Čitanje sljedećeg signala:
`irrecv.resume();`

Kod:

```
#include "IRremote.h"
int receiverPIN = 3;

IRrecv irrecv(receiverPIN);
decode_results rezultat;

void setup()
{
  Serial.begin(9600);
  Serial.println("Primljeni signal: ");
  irrecv.enableIRIn();
}
void loop()
{
  if (irrecv.decode(&rezultat))   {
      Serial.println(rezultat.value);
      delay(500);
      irrecv.resume();
  }
}
```

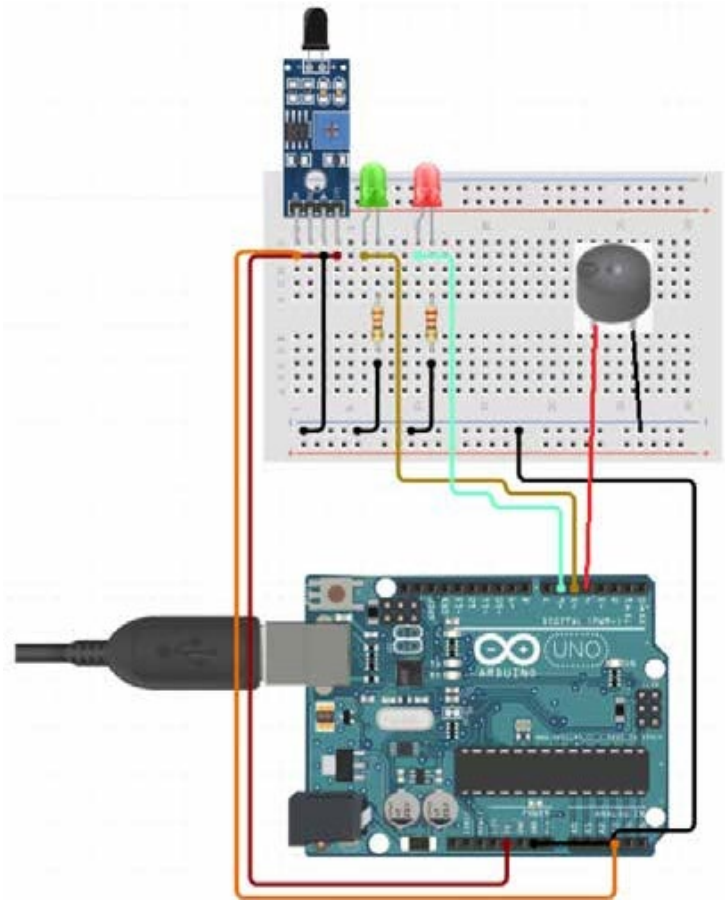
Vježba 7.
SENZOR ZA DETEKCIJU PLAMENA

Programirati Arduino UNO tako da se pri detekciji plamena oglašava buzzer i pali crvena LED. Ako plamen nije detektiran, upaliti zelenu LED, a u slučaju da je detektiran plamen, ali se ne nalazi u blizini, upaliti samo crvenu LED bez buzzera.

* Za ovaj će projekt biti bitna funkcija `map(value, fromLow, fromHigh, toLow, toHigh)`

Potrebne komponente:

- Pločica Arduino UNO i USB
- Pločica matador
- Senzor za detekciju plamena
- 1x zelena LED
- 1x crvena LED
- 2x otpornik 220 Ω



Slika 36. Šema spoja

Kod:

```
int buzzer = 4;
int zelenaLED = 5;
int crvenaLED = 6;

const int senzorMin = 0;
const int senzorMax = 1023;

void setup() {
   pinMode(buzzer, OUTPUT);
   pinMode(zelenaLED, OUTPUT);
   pinMode(crvenaLED, OUTPUT);
}
void loop() {
    int ocitanjeSenzora = analogRead(A0);

    int raspon = map(ocitanjeSenzora, senzorMin, senzorMax, 0, 2);

  switch (range) {
  case 0:
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(zelenaLED, LOW);
    digitalWrite(buzzer, HIGH);
    break;
  case 1:
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(zelenaLED, LOW);
    digitalWrite(buzzer, LOW);
    break;
  case 2:
    digitalWrite(crvenaLED, LOW);
    digitalWrite(zelenaLED, HIGH);
    digitalWrite(buzzer, LOW);
    Break;
  }
  delay(1);  // delay between reads
}
```

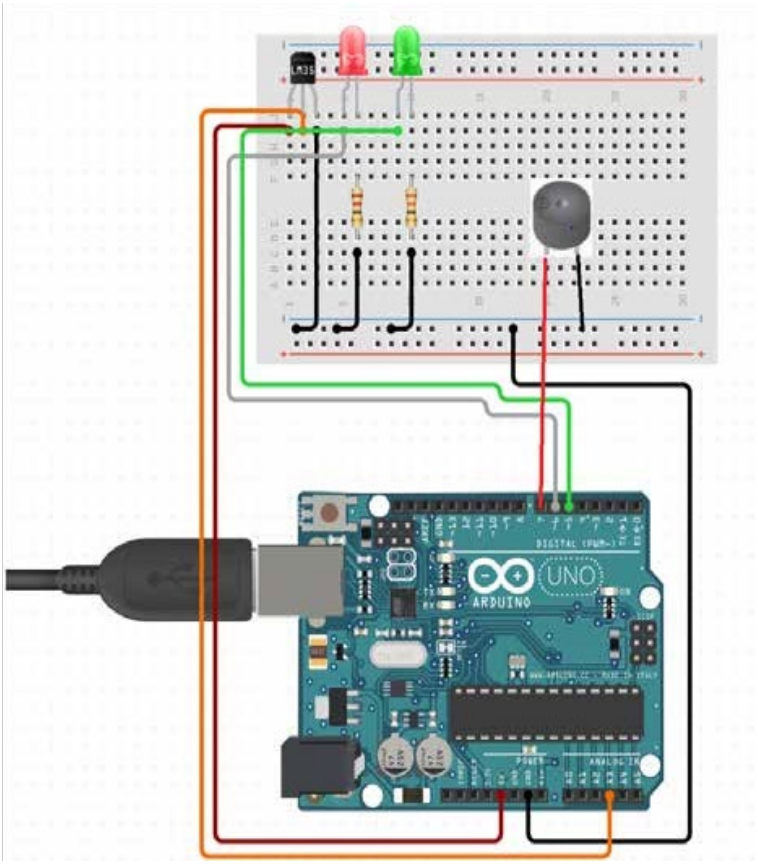
Vježba 8.
MJERENJE TEMPERATURE

S pomoću senzora TMP36 mjeriti temperaturu te upaliti zelenu LED ako je temperatura iznad 23° C. Kada je temperatura niža od 23° C, upaliti crvenu LED i buzzer.

Potrebne komponente:

- Pločica Arduino UNO i USB
- Pločica matador
- Crvena LED
- Zelena LED
- 2x otpornik 220 Ω
- Senzor TMP 36
- Buzzer

Spojiti komponente kao na slici:



Slika 37. Shema spoja

Kod:

```
int zelenaLED = 5;
int crvenaLED = 6;
int senzorPin = A3;
int buzzer = 7;

const float sobnaTemperatura = 28.0;

void setup() {
  Serial.begin(9600);
  // put your setup code here, to run once:
  pinMode(zelenaLED, OUTPUT);
  pinMode(crvenaLED, OUTPUT);
  pinMode(buzzer,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  int vrijednostSenzora = analogRead(senzorPin);
  float napon = (vrijednostSenzora / 1024.0) * 5.0;
  float temperatura = (napon - .5) * 100;
  Serial.print("stepeni C: ");
  Serial.println(temperatura);

  if (temperatura < sobnaTemperatura - 2) {
    digitalWrite(zelenaLED, LOW);
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(buzzer,HIGH);
  }
  else if (temperatura > sobnaTemperatura - 2 && temperatura < sob-
naTemperatura + 2) {
    digitalWrite(zelenaLED, HIGH);
    digitalWrite(crvenaLED, LOW);
    digitalWrite(buzzer,LOW);
  }
  else if (temperatura > sobnaTemperatura + 2) {
    digitalWrite(zelenaLED, LOW);
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(buzzer,HIGH);
  }
}
```

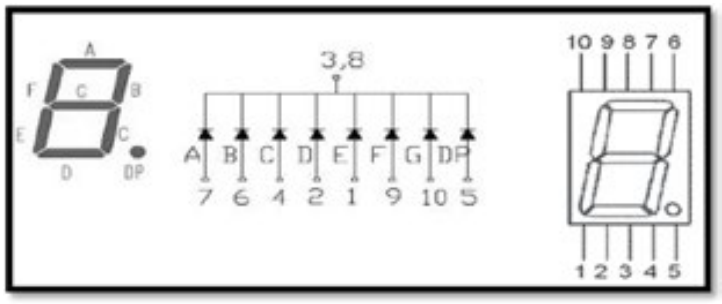
ISHODI UČENJA

ISHODI UČENJA	RAZINA POSTIGNUĆA
- Učenik/-ica razlikuje elektronske komponente koje su korištene u vježbama - Učenik/-ica opisuje ulogu elektronskih komponenti korištenih u vježbama - Učenik/-ica definira svojstva senzora za temperaturu primijenjenih u vježbama - Učenik/-ica definira način rada potenciometra - Učenik/-ica koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti - Primjenjuje znanje o uporabi funkcija iz programiranja za realizaciju zadataka - Učenik/-ica primjenjuje odgovarajuću vrstu podataka (*int*, *float*, *double*...) - Koristi odgovarajuće knjižnice unutar programa za realizaciju vježbe - Koristi shemu za povezivanje uređaja Arduino s pločicom matador i ostalim komponentama (senzori, sedam- segmentni zaslon, taster) - Koristi aplikaciju *Serial Monitor* prilikom izvršavanja vježbe - Primjenjuje procedure iz knjižnice *Serial* za izradu zadatka - Verificira i izvršava program	**Minimalna postignuća** Učenik/-ica imenuje sve potrebne komponente korištene u vježbi Učenik/-ica s pomoću nastavnika/-ice povezuje dijelove s Arduinom, verificira i izvršava program **Dovoljna postignuća** Učenik/-ica objašnjava ulogu elektronskih komponenti korištenih u vježbi Povezuje elektronske komponente prema shemi spoja Učenik/-ica posjeduje osnovna znanja o pisanju programa za prethodne vježbe Koristi se odgovarajućim knjižnicama unutar programa Samostalno verificira i izvršava program te po potrebi s pomoću nastavnika/-ice otklanja nastali bug **Visoka postignuća** Učenik/-ica samostalno povezuje komponente korištene u vježbama prema shemi spoja Primjenjuje svoje znanje iz programiranja za pisanje koda Uočava primjenu Arduino uređaja u drugim oblastima (fizika, matematika, kemija...) Samostalno verificira i izvršava program, te po potrebi uspješno otklanja nastali bug

Sedam-segmentni zaslon

Sedam-segmentni zaslon predstavlja izlazni uređaj na kojemu je najlakše ljudima prikazati informacije. Može prikazati sve brojeve, ali i neka slova. Dolazi u dvije konfiguracije i to sa zajedničkom katodom ili sa zajedničkom anodom, na slici ispod prikazana je konfiguracija sa zajedničkom katodom.

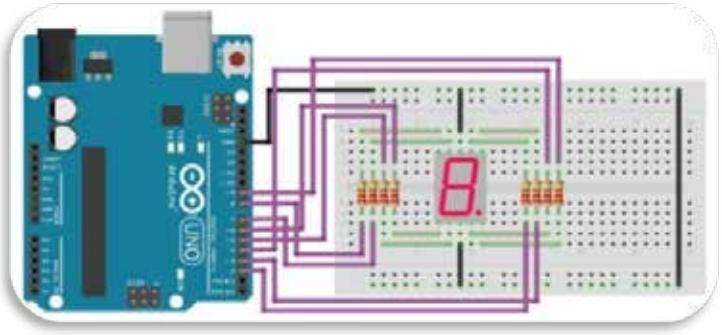
Dakle, anode LE dioda spajaju se na pinove mikrokontrolera preko predotpora, a zajednički pin na *masu-gnd* na razvojnoj platformi Arduino. HIGH signal na pinu mikrokontrolera aktivira jedan segment na zaslonu. Na taj način kombinacijom aktivnih pinova dobijemo prikaz brojeva na sedam-segmentnom zaslonu.



Slika 38. Pinout za sedam-segmentni zaslon

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. Sedam-segmentni zaslon 1 kom.
3. Otpornik 330 Ω 7 kom.



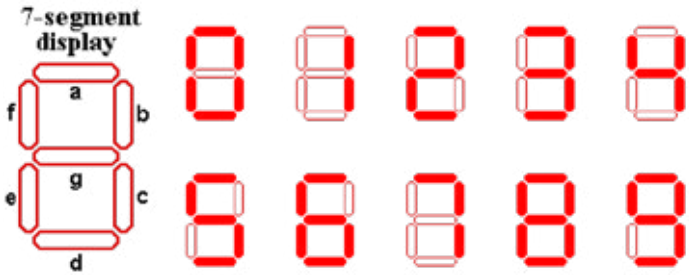
Slika 39. Šema spoja

Koraci za realizaciju vježbe:

1. Kreirajte shemu spoja koristeći skicu *Fritzing*.
2. Kreirajte *Sketch* kojim ćete upravljati radom sedam-segmentnog zaslona.

Više je načina da se taj problem riješi. Jedan od njih je da ponovno bez uporabe drugih programskih struktura riješimo problem. Ali prvo trebamo definirati pinout koji će nam olakšati samo programiranje i rješavanje problema:

PIN2 -> a
PIN3 -> b
PIN4 -> c
PIN5 -> d
PIN6 -> e
PIN7 -> f
PIN8 -> g
PIN9 -> dp



Slika 40. Prikaz brojeva na segmentnom zaslonu

Prema gornjoj slici kod za prikaz broja „1" na našem zaslonu bio bi sljedeći:

```
        digitalWrite(a, LOW);
        digitalWrite(b, HIGH);
        digitalWrite(c, HIGH);
        digitalWrite(d, LOW);
        digitalWrite(e, LOW);
        digitalWrite(f, LOW);
        digitalWrite(g, LOW);
```

Odnosno aplikacija za prikaz broja „1” na sedam-segmentnom zaslonu bila bi:

```
int a=2;
int b=3;
int c=4;
int d=5;
int e=6;
int f=7;
int g=8;

void setup ()
{
   for(int i=2;i<=9;i++)
    {
          pinMode(i,OUTPUT);//pinovi od 2 do 9 OUTPUT
    }
}

void loop()
{
digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, HIGH);
digitalWrite(d, LOW);
digitalWrite(e, LOW);
digitalWrite(f, LOW);
digitalWrite(g, LOW);
}
```

Naravno, jednostavniji način bio bi da se za svaki broj kreira funkcija custom za prikaz broja „1” te da se ona pozove u loop petlju. To je moguće napraviti na sljedeći način:

```
vovoid jedan() //kreiranje funkcije za prikaz broja jedan
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
}
```

Odnosno naš cijeli kod za prikaz broja „1” izgledao bi na sljedeći način:

```
int a=2;
int b=3;
int c=4;
int d=5;
int e=6;
int f=7;
int g=8;

void jedan()//kreiranje funkcije za prikaz broja “1”
{
digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, HIGH);
digitalWrite(d, LOW);
digitalWrite(e, LOW);
digitalWrite(f, LOW);
digitalWrite(g, LOW);

}
void setup ()
{
   for(int i=2;i<=9;i++)
    {
          pinMode(i,OUTPUT);//pinovi od 2 do 9 OUTPUT
    }
}

void loop()
{
    jedan();  //pozivanje funkcije za prikaz broja “1”
}
```

Ako bismo željeli pozvati funkciju za prikaz broja „2”, prije pozivanja funkcije potrebno je „ugasiti” zaslon; funkcija za gašenje zaslona izgledala bi ovako:

```
void gasi() //kreiranje funkcije za "isključenje" zaslona
{
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
```

```
void dva() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
void tri() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void cetiri() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void pet() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
```

```
void sest() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
void sedam() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void osam() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void devet() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
```

Potrebno je gornje izraze dopuniti i kreirati aplikaciju koja će svake sekunde prikazati drugi broj na zaslonu. Dakle, pravimo brojač od 0 do 9.

Ovo je možda najbolji primjer da proširimo svoje znanje iz programiranja, jer se ovaj isti zadatak može riješiti na više različitih načina. Jedan je od načina uporaba ***if*** testa.

```
if (uvjet ispunjen)

        {

  //napravi nešto

        }
```

If test se koristi s komparacijskim operatorima.

x == y (x jednako y)
x != y (x nije jednako y)
x < y (x manje od y)
x > y (x veće od y)
x <= y (x manje ili jednako od y)
x >= y (x veće ili jednako y)

Jedan tipični *if* test glasio bi:

if(ocjenaUcenika < 2) upaliAlarm();

Kako nam *if* test može pomoći da svoj brojač riješimo na ljepši način? Kao prvo, potrebna nam je deklaracija jedne globalne varijable brojač. Globalne varijable deklariraju se izvan tijela funkcije i vidljive su svim funkcijama u programu. Lokalne varijable deklariraju se unutar neke funkcije i samo ta funkcija vidi tu varijablu.

```
int brojac=0;   // globalno deklarirana varijabla, svi je

"vide" void setup ()

{

    int i=0;    // lokalno deklarirana varijabla, samo setup je
                "vidi"
}
```

Pokušajmo iskoristiti novostečena znanja na praktičnom primjeru.

```
      int a=2;
      int b=3;
      int c=4;
      int d=5;
      int e=6;
      int f=7;
      int g=8;
      int brojac = 0;
      void jedan()//kreiranje funkcije za prikaz broja "1"
      {
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);

      }
      void setup ()
      {
         for(int i=2;i<=9;i++)
          {
                pinMode(i,OUTPUT);//pinovi od 2 do 9 OUTPUT
          }
      }

      void loop()
      {
      if (brojac==0)nula();//ako je stanje brojača 0, pozovi funkciju za
prikaz
                           //broja 0, ako poslije if testa imamo samo
jedan
                           //programski iskaz, nisu potrebne vitičaste
zagrade

          brojac++;     // za svaki prolaz kroz loop napravi
brojača
          inkrement delay(1000); // sačekaj sekundu

          if (brojac ==9) //kada izbrojimo do 9
          {
                 brojac=0;    //vratimo stanje brojača na 0 da ponovno
brojimo
          }
      }
```

Dodajte linije koda da stanje na sedam-segmentnom zaslonu odgovara stanju globalnoga brojača. Kompajlirajte i testirajte. Kreirajte dodatnu opciju da brojač broji unazad.

```
      int a=2;
      int b=3;
      int c=4;
      int d=5;
      int e=6;
      int f=7;
      int g=8;

      int brojac = 0;

      void jedan()//kreiranje funkcije za prikaz broja "1"
      {
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);

      }
      void setup ()
      {
         for(int i=2;i<=9;i++)
          {
                pinMode(i,OUTPUT);//pinovi od 2 do 9 OUTPUT
          }
      }

      void loop()
      {
      if (brojac==0)nula();//ako je stanje brojača 0, pozovi funkciju za
prikaz
                           //broja 0, ako poslije if testa imamo samo
jedan
                           //programski iskaz, nisu potrebne vitičaste
zagrade

          brojac++;     // za svaki prolaz kroz loop napravi
brojač
          inkrement delay(1000); // sačekaj sekundu

          if (brojac ==9) //kada izbrojimo do 9
          {
                 brojac=0;    //vratimo stanje brojača na 0 da ponovno
brojimo
          }
      }
```

Drugi način da se isti ovaj zadatak riješi je korištenjem strukture switch... case.

```
int brojac=0;
void loop ()
{
switch (brojac)
{
    case 1:
      jedan();      // kada je stanje brojača 1, uđi u case 1
      break;
    case 2:
      dva();       // kada je stanje brojača 2, uđi u case 2
      break;
    default:
      nula();     // kada je stanje brojača 0, uđi u default

    break;
  }

}
```

Pokušajte riješiti aplikaciju brojanja i na ovaj način!

Digitalni ulazi

Već smo na početku pisali o ulazima u mikrokontroler. Kada govorimo o digitalnim ulazima, mislimo na ulaze koji mogu imati samo dva stanja: stanje logičke 1 i logičke 0, odnosno ako je riječ recimo o prekidaču, onda kažemo da možemo imati dva stanja:

prekidač zatvoren -> logička „1“
otvoren -> logička „0“.

Prije svega moramo svom mikrokontroleru reći da više nećemo imati interakciju samo s izlazima već da će na jedan pin biti spojen jedan input, tako da sada za taj input na koji je spojen taster, prekidač ili neki senzor, *pinMode* funkcija glasi:

```
int inPin = 2;
int outPin = 13;
void setup()
{
 pinMode(inPin, INPUT);           //pin 2 je input
 pinMode(outPin, OUTPUT);         //pin 13 je output
}
```

Provjeru stanja na input pinu 2 radimo s pomoću funkcije

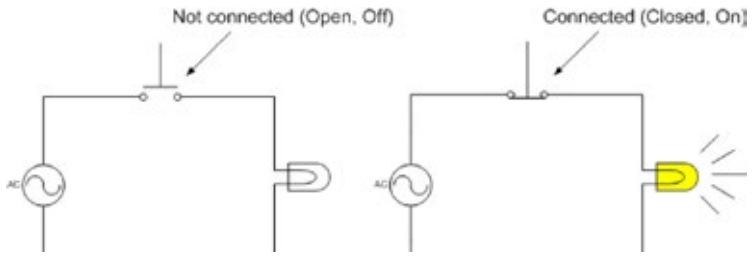
digitalRead(pin)

Funkcija ima samo jedan argument „pin” s kojega čitamo stanje.

```
    /*
    Praćenje stanja na digitalnom ulazu te promjena stanja na LED na
pinu 13 spram stanja na digitalnom ulazu-> ako je korisnik pritisnuo
taster, upali LED; ako taster nije pritisnut, ugasi LED
    */
    int inPin = 2;
    int outPin = 13;
    int inPinstate = 0; //varijabla u koju ćemo spremiti stanje na pinu
2
    void setup()
    {
     pinMode(inPin, INPUT);           //pin 2 je input
     pinMode(outPin, OUTPUT);         //pin 13 je output
    }


    void loop()
    {
      inPinstate = digitalREad(inPin);//spremi stanje na pinu 2 u vari-
jablu
      if(inPinstate == 1)
            {
            digitalWrite(outPin, HIGH);
                  }
      else
                  {
            digitalWrite(outPin, LOW);
                  }

    }
```

Time smo obradili softversku stranu problema, idemo sada vidjeti kako ćemo riješiti hardver za digitalni ulaz.

Mi stalno imamo interakciju s prekidačima i tasterima, doma, u stubištu zgrade, dizalu, raznim kućanskim aparatima. Kratko ćemo opisati ponašanje klasičnog prekidača za sijalicu. Taj je prekidač jedan jednostavan uređaj koji ima dvije pozicije, i to uključeno i isključeno. To je slikovito prikazano na slici ispod.

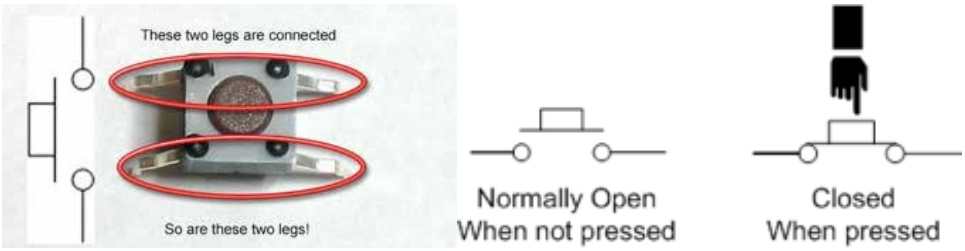


Slika 41. Taster aktivan i taster neaktivan

Naravno, prekidači koje koristimo u svojim domovima su superpouzdani, ali jednostavno su gabaritima veliki za primjenu u mikrokontrolerskim sustavima. Mi ćemo za svoje eksperimente koristiti mikrotaster od 6 mm.

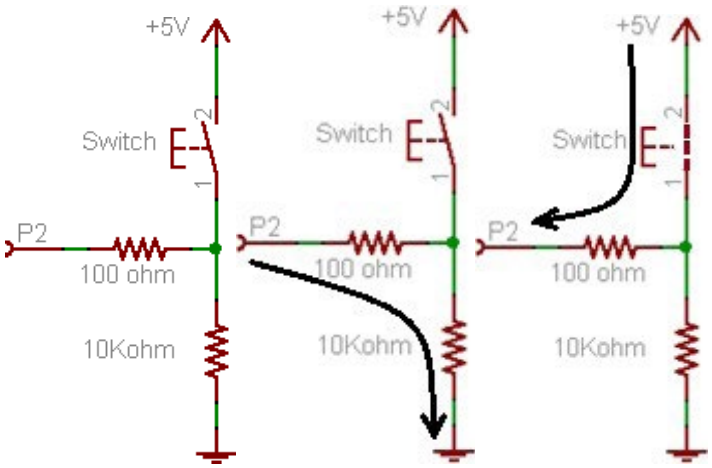
Slika 42. Mikrotaster od 6 mm

Ovi su mali tasteri jeftini, praktični za uporabu na matador pločama, četiripinski su, s tim da su po dva pina spojena zajedno tako da su bez obzira na četiri nožice ovo dvožični prekidači.



Slika 43. Mikrotaster od 6 mm, unutarnja struktura

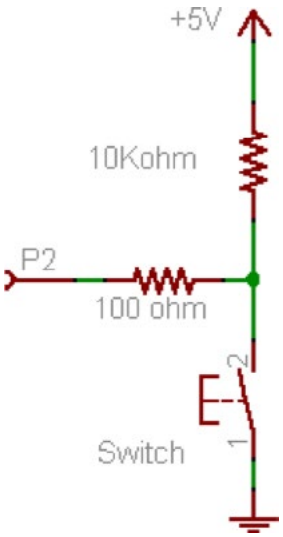
Dobra je praksa nožice ispraviti jer na taj se način lakše pozicioniraju na matador ploči. Ali kako formirati električno kolo na ispravan način? Naime, dvije su kombinacije kako spojiti taster na ulaz mikrokontrolera. To su konfiguracija pull-down i pull-up. Konfiguracija pull-down prikazana je na slici ispod.



Slika 44. Konfiguracija pull-down

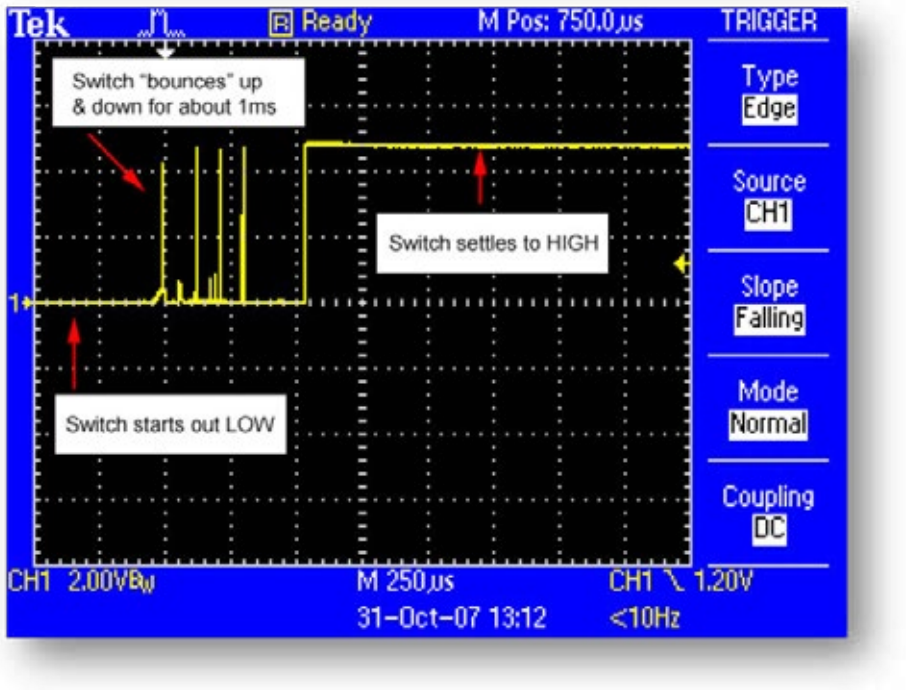
Objasnit ćemo prethodnu sliku. Kada taster nije pritisnut, pin 2 mikrokontrolera je preko 100-omskog i 10-kiloomskog otpora spojen na masu. Stoga kontroler to vidi kao logičku 0 (nulu). S druge strane, kada se taster pritisne, imamo situaciju da na ulaz pina 2 mikrokontrolera preko 100-omskog otpora dolazi 5 V, što mikrokontroler vidi kao logičku 1 (jedinicu).

Druga je konfiguracija tzv. konfiguracija *pull-up*. Koncept je sličan, a logika obratna. Opišite je.



Slika 45. Konfiguracija pull-up

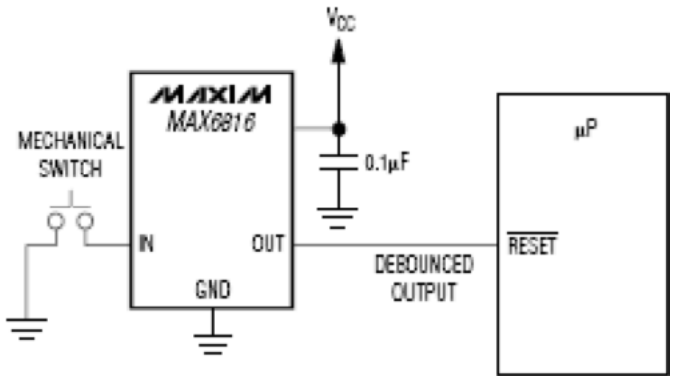
Drugi problem kod digitalnih ulaza s mehaničkim prekidačima je tzv. ***debouncing*** ili odskakanje kontakata, koje se javlja prilikom pritiska tastera. Učinak odskakanja kontakata najbolje se vidi snimanjem signala oscilosko-pom.



Slika 46. Debouncing – odskakanje kontakata

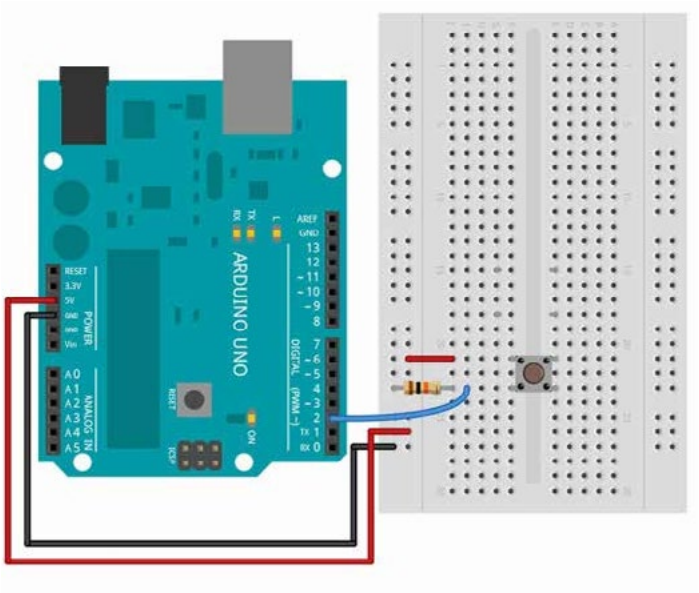
Sada je jasno da će mikrokontroler umjesto da jednom očita stanje logičke jedinice to u slučaju sa slike napraviti pet puta, tako da ako bismo imali aplikaciju koja broji pritiske tastera korisnika, svaki bismo put imali pogrešno brojanje.

Više je načina da se riješi taj problem. Jedan od njih je uporaba debouncing integralnog kola, kao na slici ispod.



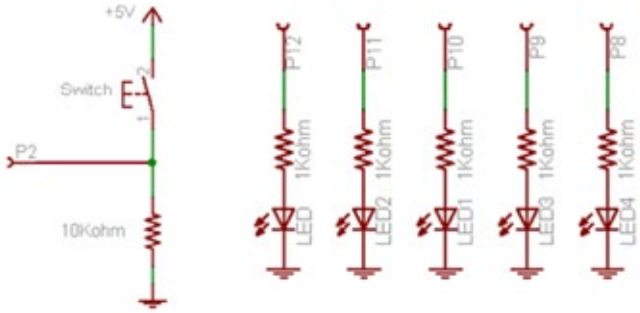
Slika 47. Debouncing integralno kolo

Naravno, to bi bilo neko profesionalno rješenje. Mi možemo problem riješiti tako što ćemo dati vremena da se kontakti smire, te ćemo poslije pritiska tastera sačekati do 10 sekundi. Naravno, postoje i druga rješenja, ali mi ćemo ovdje iskoristiti najjednostavnije. Pa idemo probati.



Slika 48. Pojednostavljena shema spajanja tastera na pin 2

Prvo je potrebno formirati spajanje za kontrolu LED na ploči matador prema sljedećoj elektro-shemi.



Slika 49. Shema spajanja za kontrolu LED

```
     /*
     Praćenje stanja na digitalnom ulazu te promjena stanja na LED na
pinovima od 12 do 8. Brojanje koliko je puta korisnik pritisnuo taster,
povećavanje vrijednosti globalne varijable counter te uključenje odgova-
rajuće LED diode spram stanja brojača.
     */
     int inPin = 2;
     int ledOut1 = 12;
     int ledOut2 = 11;
     int ledOut3 = 10;
     int ledOut4 = 9;
     int ledOut5 = 8;
     int counter = 0;
     int inPinstate = 0;               //varijabla stanja pina 2

     void gasi()                       // funkcija za gašenje svih LED
     {
         for(int i=8; i<=12; i++)
         {
               digitalWrite(i, LOW);         //ugasi led od 8 do 12
         }
     }
     void setup()
     {
      pinMode(inPin, INPUT);                 //pin 2 je input
         for(int i=8; i<=12; i++)
         {
               pinMode(i, OUTPUT);     //pinovi od 8 do 12 OUTPUT
         }
     }

     void loop()
     {
         if(counter == 0) gasi();      // gasi sve za count =0

         inPinstate = digitalREad(inPin);  //spremi stanje na pin2 u
varijablu

         if(inPinstate == 1) counter++;// povećaj stanje kada delay
         (250);                        // sačekajmo malo

         if(counter == 1) digitalWrite(ledOut1, HIGH);
             if(counter == 2) digitalWrite(ledOut2, HIGH);
             if(counter == 3) digitalWrite(ledOut3, HIGH);
             if(counter == 4) digitalWrite(ledOut4, HIGH);
             if(counter == 5) digitalWrite(ledOut5, HIGH);
         if(counter >5) counter = 0;
     }
```

Iskoristite sedam-segmentni zaslon te prikažite vrijednost varijable counter na sedam-segmentnom zaslonu.

```
/*
*/
int inPin = 2;
int ledOut1 = 12;
int ledOut2 = 11;
int ledOut3 = 10;
int ledOut4 = 9;
int ledOut5 = 8;
int counter = 0;
int inPinstate = 0;                //varijabla stanja pina 2

void gasi()                        // funkcija za gašenje svih LED
{
    for(int i=8; i<=12; i++)
    {
          digitalWrite(i, LOW);          //ugasi led od 8 do 12
    }
}
void setup()
{
 pinMode(inPin, INPUT);                  //pin 2 je input
    for(int i=8; i<=12; i++)
    {
          pinMode(i, OUTPUT);      //pinovi od 8 do 12 OUTPUT
    }
}

void loop()
{
    if(counter == 0) gasi();       // gasi sve za count =0

    inPinstate = digitalREad(inPin);  //spremi stanje na pin2 u
varijablu

    if(inPinstate == 1) counter++;// povećaj stanje kada
    delay (250);                   // sačekajmo malo

    if(counter == 1) digitalWrite(ledOut1, HIGH);
        if(counter == 2) digitalWrite(ledOut2, HIGH);
        if(counter == 3) digitalWrite(ledOut3, HIGH);
        if(counter == 4) digitalWrite(ledOut4, HIGH);
        if(counter == 5) digitalWrite(ledOut5, HIGH);
    if(counter >5) counter = 0;
}
```

Izazov za nastavnike

Cilj vježbe:

Uporaba funkcije *digitalRead()*

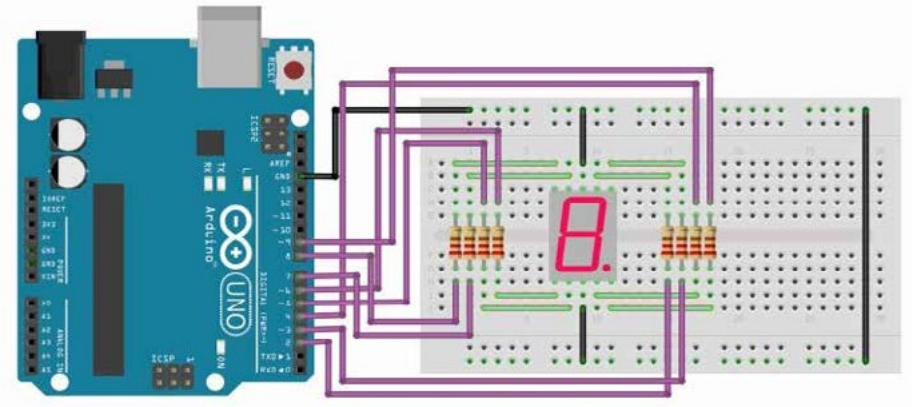
Zadatak vježbe:

Kreirati *Sketch* za kontrolu rada sedam-segmentnog zaslona -> *If uvjeti!*

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. Sedam-segmentni zaslon 1 kom.
3. Otpornik 10 kΩ 1 kom.
4. Otpornik 330 Ω 1 kom.
5. Taster 1 kom.

Shema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte sklop koristeći skicu Fritzing.
2. Kreirajte Sketch kojim ćete upravljati radom sedam-segmentnog zaslona na taj način da se svakim pritiskom tastera mijenja prikaz na njemu od 0 do 9.
3. Upisati rješenje u za to predviđen prostor.
4. Koristite primjere i gotove funkcije iskodirane u primjerima za digital output.

Napisati kod:

Cilj vježbe:

Uporaba funkcije *digitalRead()*

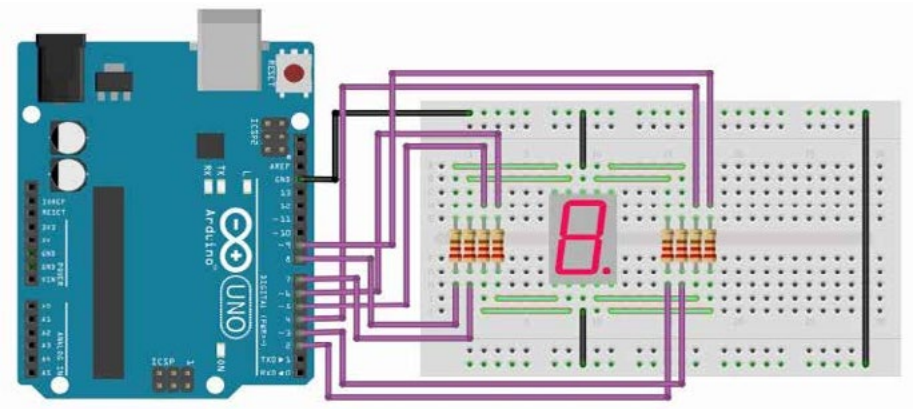
Zadatak vježbe:

Kreirati *Sketch* za kontrolu rada sedam-segmentnog zaslona (***switch case***)

Potrebne komponente za vježbu:

Arduino Uno	1 kom.
Sedam-segmentni zaslon	1 kom.
Otpornik 10 kΩ	1 kom.
Otpornik 330 Ω	1 kom.
Taster	1 kom.

Shema spoja:



Koraci za realizaciju vježbe

1. Kreirajte sklop koristeći skicu *Fritzing*.
2. Kreirajte *Sketch* kojim ćete upravljati radom sedam-segmentnog zaslona na taj način da se svakim pritiskom tastera mijenja prikaz na njemu od 0 do 9.
3. Upisati rješenje u za to predviđen prostor.
4. Koristite primjere i gotove funkcije iskodirane u primjerima za *digital output*.

Napisati kod:

Cilj vježbe:

Uporaba funkcije *digitalRead()*

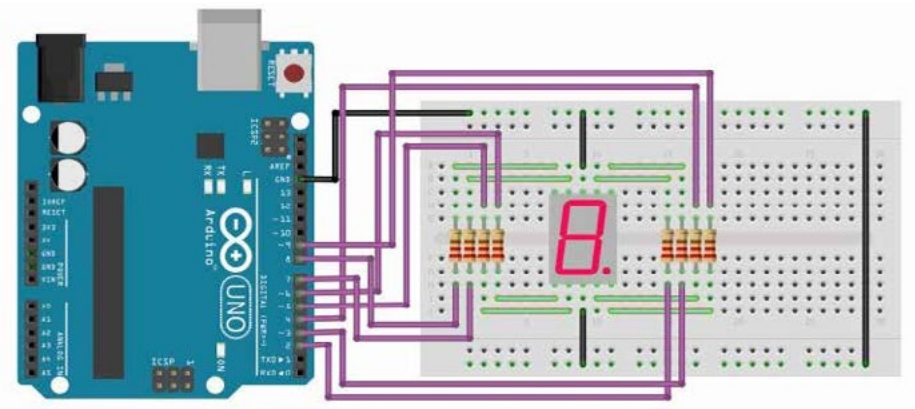
Zadatak vježbe:

Kreirati *Sketch* za kontrolu rada sedam-segmentnog zaslona.-> up/down brojač!

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. Sedam-segmentni zaslon 1 kom.
3. Otpornik 10 kΩ 2 kom.
4. Otpornik 330 Ω 1 kom.
5. Taster 2 kom.

Shema spoja:



Koraci za realizaciju vježbe

1. Kreirajte sklop koristeći skicu *Fritzing*.
2. Kreirajte *Sketch* kojim ćete upravljati radom sedam-segmentnog zaslona na taj način da se svakim pritiskom tastera za brojanje na gore i na dolje mijenja stanje brojača koje je prikazano na sedam-segmentnom zaslonu.
3. Upisati rješenje u za to predviđen prostor.
4. Koristite primjere i gotove funkcije iskodirane u primjerima za *digital output*.

Napisati kod:

Serial knjižnica – software debugging

Jedna od mana razvojne platforme Arduino je to što nema hardverski debugger, koji omogućava mrežno praćenje i izvršavanje programskih instrukcija, tako da je lakše pronaći greške u kodu. Kreatori Arduina stoga su kreirali skup procedura koje omogućavaju da imamo izvještavanje, što se trenutačno dešava s našim kodom.

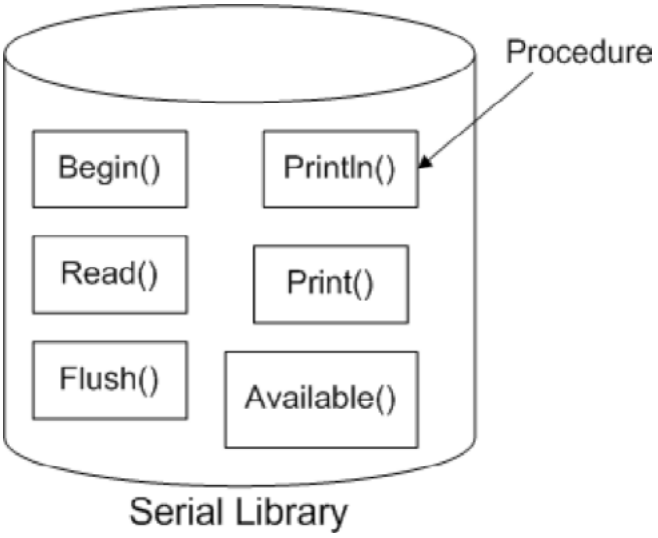
Taj skup procedura zove se knjižnica, u našem konkretnom slučaju riječ je o Serial Library, koja omogućava da PC aplikacija komunicira s Arduino kroz USB port.

Prije no što nastavimo, reći ćemo nekoliko riječi o knjižnicama. Recimo, kada budete imali zahtjev da napravite aplikaciju za kontrolu stepper ili servo motora, vjerojatno ćete trebati knjižnicu servo.h ili knjižnicu stepper.h. Pozivanje knjižnice radi se na sljedeći način:

#include <servo.h>

Na taj način pozivamo i koristimo sve procedure iz knjižnice servo te tako olakšavamo sebi rješavanje problema, odnosno izbjegavamo pisanje svojih procedura za kontrolu servo motora.

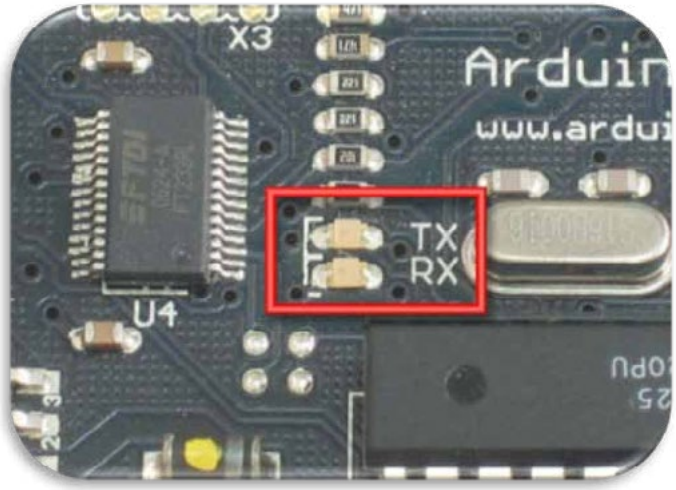
Knjižnica koju prvu trebamo naučiti koristiti je *Serial knjižnica*. Na slici ispod prikazane su osnovne procedure koje ćemo koristiti u narednim primjerima.



Slika 50. Procedure iz knjižnice Serial

Mi smo već koristili serijsku komunikaciju, jer svaki put kada radimo **Compile/Verify** naše skice, kod pretvaramo u binarni zapis (nule i jedinice), a kada uradimo **Upload**, u stvari šaljemo niz tih istih nula i jedinica ka Arduinu, koje se onda pohranjuju na prostor predviđen za aplikaciju.

Primijetili ste da svaki put kada radimo *upload* koda na Arduino, dvije LED označene sa RX i TX blinkaju kada imamo protok informacija, odnosno nula i jedinica. RX linija blinka kada imamo proces primanja podataka na kontroler, a TX linija kad imamo proces slanja podataka od kontrolera ka PC-u.

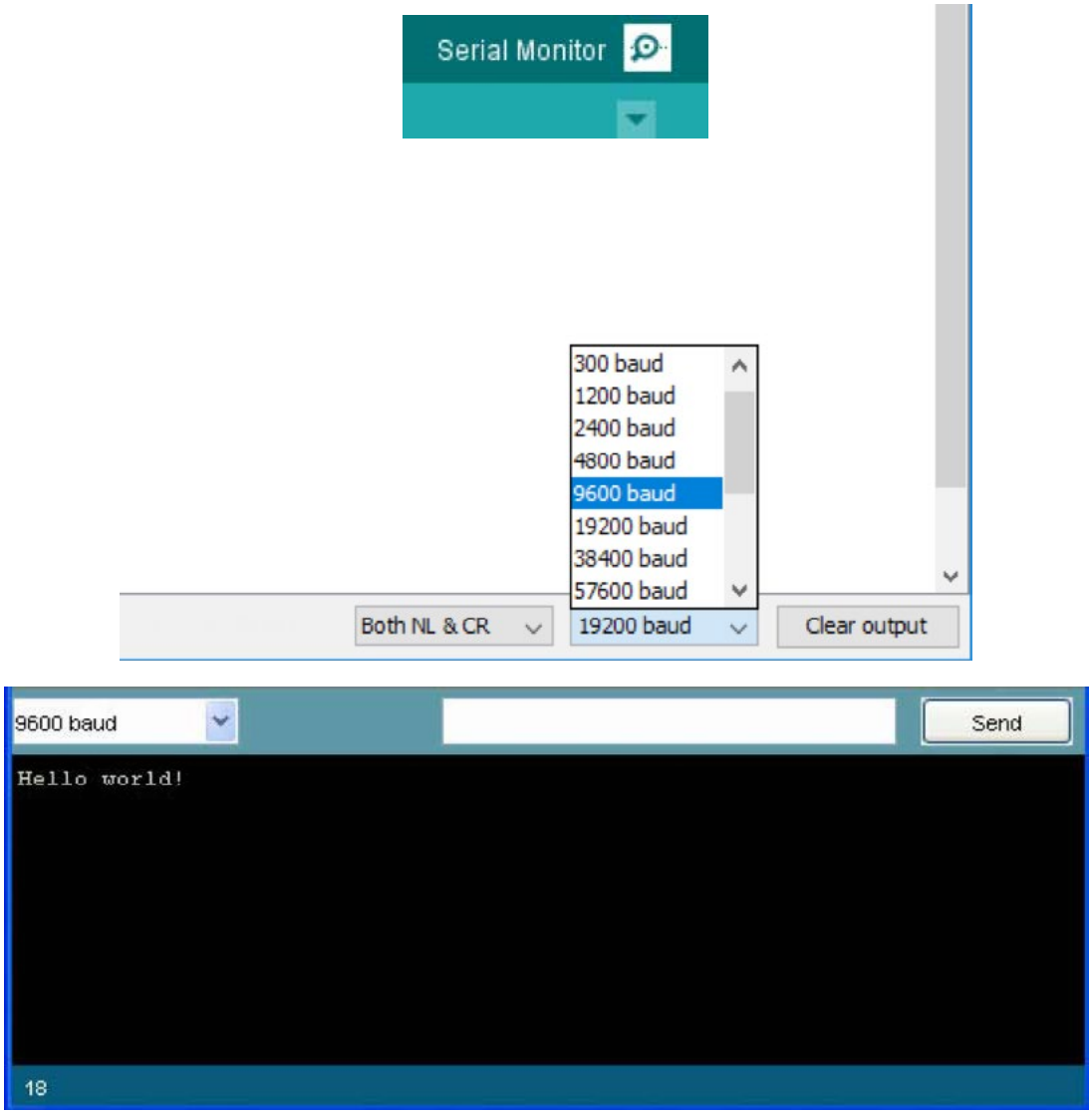


Slika 51. RX i TX LED

Idemo napraviti još jednu aplikaciju „Zdravo, svijete!", ali ovaj put komunikacijsku. Kako sve više postajemo programeri, značenje procedura učit ćemo iz komentara.

```
    /*
    Aplikacija „Zdravo, svijete!", u kojoj ćemo vidjeti kako Arduino
šalje informacije na PC. Nećemo morati uraditi #include <Serial.h> jer
je ova knjižnica build-in, dakle već ugrađena
    */
    void setup()
    {
     Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps
     Serial.println("Zdravo, svijete!");// pošalji na PC „Zdravo,
svijete!"
    }

    void loop()
    {
    }
```

Nakon što ste kompajlirali i učitali aplikaciju u Arduino, na Arduino IDE-u nađite kraticu za aplikaciju *Serial Monitor,* podesite brzinu kao u aplikaciji na 9600 bps. Po potrebi resetirajte kontroler na tasteru *reset* na razvojnoj platformi.

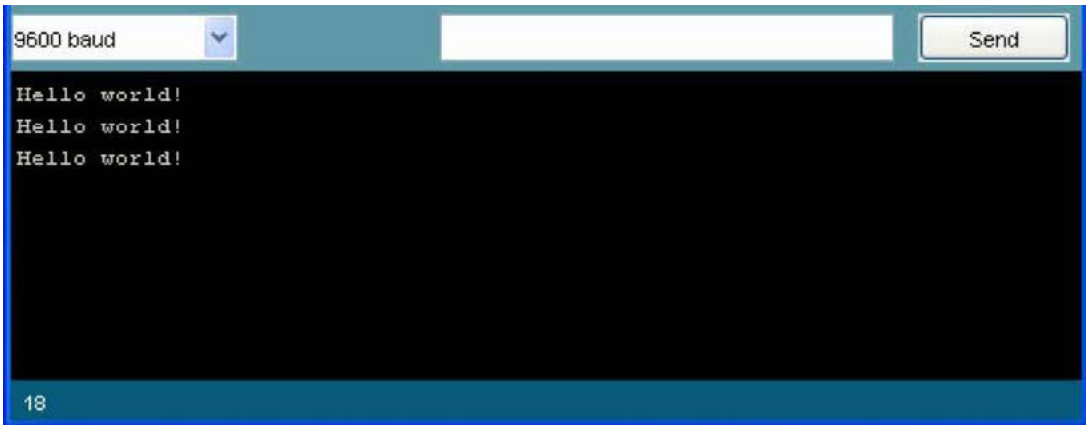


Slika 52. Aplikacija „Hello, World!" za komunikaciju

Kako smo proceduru *println()* pozvali u petlji *setup,* koja se izvršava samo jedanput, to će na serijskom ekranu biti ispisano da je string *„Zdravo, svijete!"* poslan na PC samo jednom. Idemo to probati u petlji *loop*.

```
     /*
     Aplikacija „Zdravo, svijete!”, u kojoj ćemo vidjeti kako
Arduino šalje informacije na PC
     */

     void setup()
     {
      Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps

     }

     void loop()
     {
         Serial.println(“Zdravo, svijete!”);// pošalji na PC „Zdravo,
svijete!”
         delay(1000);                  // čekaj 1 sekundu
     }
```

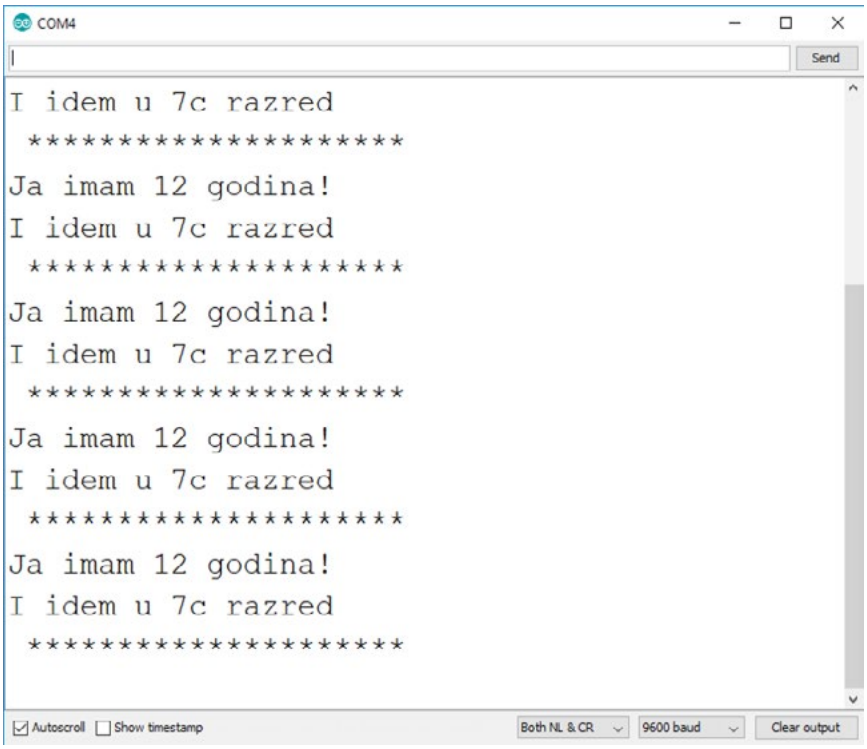
Rezultat ovog koda prikazan je na slici ispod, dakle svake sekunde mikrokontroler na PC pošalje poruku „Hello, World!”.



Slika 53. „Hello, World!”

Vidi se da funkcija println šalje na PC poruku svaki put u novom redu. Ako želimo kreirati neku kompleksniju poruku, možemo koristiti i proceduru print, koja poruku ispisuje u istom redu, pa idemo to testirati.

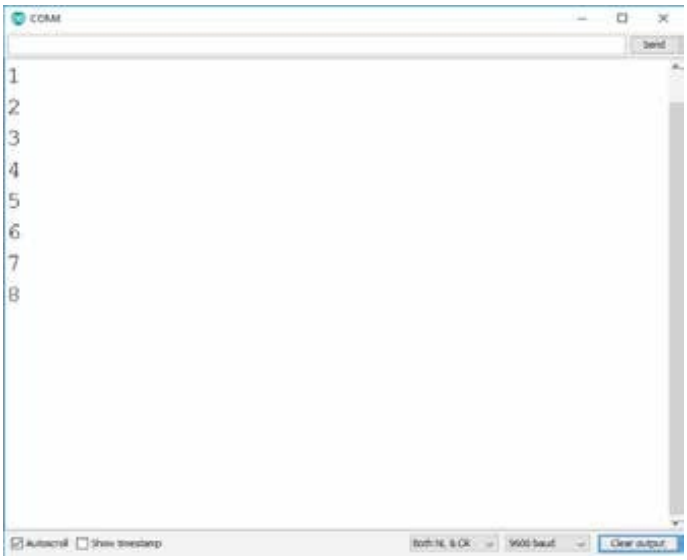
```
     /*
     Aplikacija „Zdravo, svijete!”, u kojoj ćemo vidjeti kako
Arduino šalje informacije na PC
     */

     void setup()
     {
      Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps

     }

     void loop()
     {
         Serial.print(“Ja imam”);
             Serial.print(“  “);       // upisati godine
             Serial.println(“godina!”);
             Serial.print(“I idem u”);
             Serial.print(“  “);       // upisati razred
             Serial.println(“razred”);
         Serial.println(“ **********************  “);
         delay(1000);                  // čekaj 1 sekundu
     }
```



Slika 54. Možete napisati i pismenu zadaću u Arduinu

Dakle, iz gornjeg primjera vidi se da je moguće kreirati i malo kompleksnije informacije koje se mogu slati na PC. Prije no što pokušamo raditi kompleksnu matematiku u Arduinu, idemo vidjeti kako možemo *Serial Monitor* koristiti kao debugger odnosno kako možemo pratiti stanje neke varijable u aplikaciji.

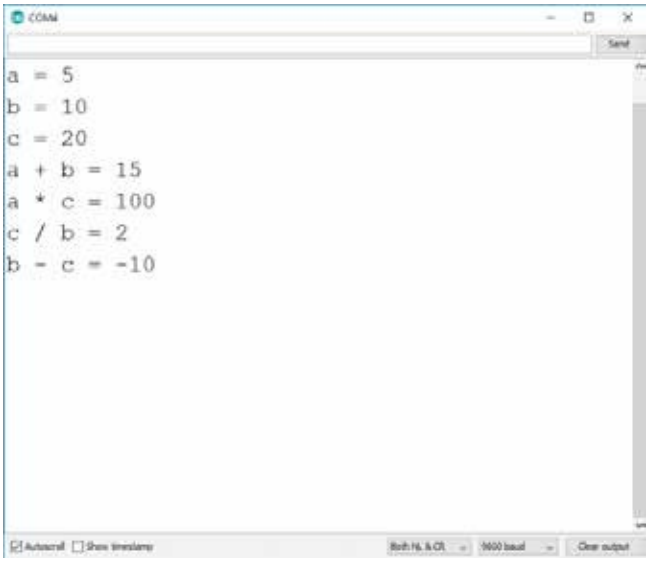
```
     /*
     Aplikacija „Zdravo, svijete!”, u kojoj ćemo vidjeti kako
Arduino šalje informacije na PC
     */
     int i=0;

     void setup()
     {
      Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps

     }

     void loop()
     {
         i++;               // radi inkrement
         Serial.println(i);

         delay(1000);             // čekaj 1 sekundu
     }
```



Slika 55. Prikaz stanja bojača u Serial Monitoru

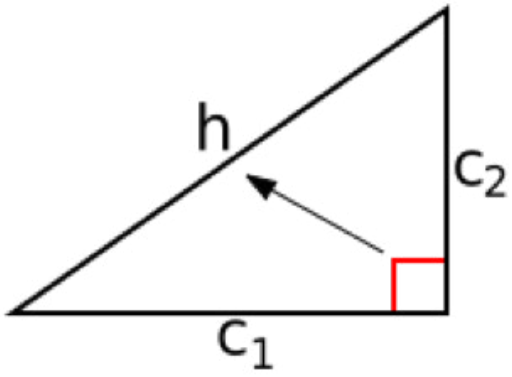
Idemo probati raditi matematiku koristeći Arduino.

```
/*
 Domaća zadaća iz matematike na malo drugačiji način!
 */

int a = 5;
int b = 10;
int c = 20;

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

  Serial.println("Evo malo matematike: ");

  Serial.print("a = ");
  Serial.println(a);
  Serial.print("b = ");
  Serial.println(b);
  Serial.print("c = ");
  Serial.println(c);

  Serial.print("a + b = ");       // sabiranje
  Serial.println(a + b);

  Serial.print("a * c = ");       // množenje
  Serial.println(a * c);

  Serial.print("c / b = ");       // dijeljenje
  Serial.println(c / b);

  Serial.print("b - c = ");       // oduzimanje
  Serial.println(b - c);
}

void loop()
{
}
```



Slika 56. Rješenje u Serial Monitoru

DIY – serial



Slika 57. Pitagorina teorema

Izračunajte hipotenuzu koristeći Pitagorinu teoremu i pripadajuće formule. Za izračunavanje korijena koristite funkciju ***sqrt***.

$$a2 + b\,2 = h\,2$$

$$h = \sqrt{(a2 + b2)}$$

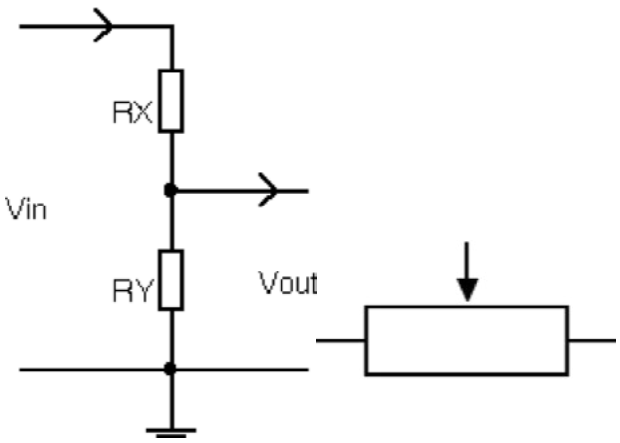
Prepisati rješenje:

```
/*
 * Math is fun!
 */



int a = 3;
int b = 4;
int h;

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

  Serial.println("Proračun hipotenuze:");

  Serial.print("a = ");
  Serial.println(a);

  Serial.print("b = ");
  Serial.println(b);

  h = sqrt( a*a + b*b );

  Serial.print("h = ");
  Serial.println(h);
}

void loop()
{
}
```

AnalogRead

Na platformi Arduino imamo šest naponskih analognih ulaza. Za razliku od digitalnih ulaza, na koje se mogu spojiti uređaji koji mogu imati samo dvije vrijednosti „0" ili „1", na analogni ulaz možemo spojiti senzore ili elektronske komponente koje na svom izlazu mogu dati napon od 0 od 5 V. Najjednostavnija elektronska komponenta koju možemo spojiti na analogni ulaz mikrokontrolera je potenciometar.

Potenciometar je tropinski promjenjivi otpornik koji ima klizni ili rotirajući kontakt, tako da se u kolu, kada je spojen kao na slici ispod, ponaša kao djelitelj napona tj. ako na njegove krajeve dovedemo napon Vin, napon Vout bit će duplo manji od Vin ako je ***Rx=Ry.***



Slika 58. Djelitelj napona i simbol potenciometra

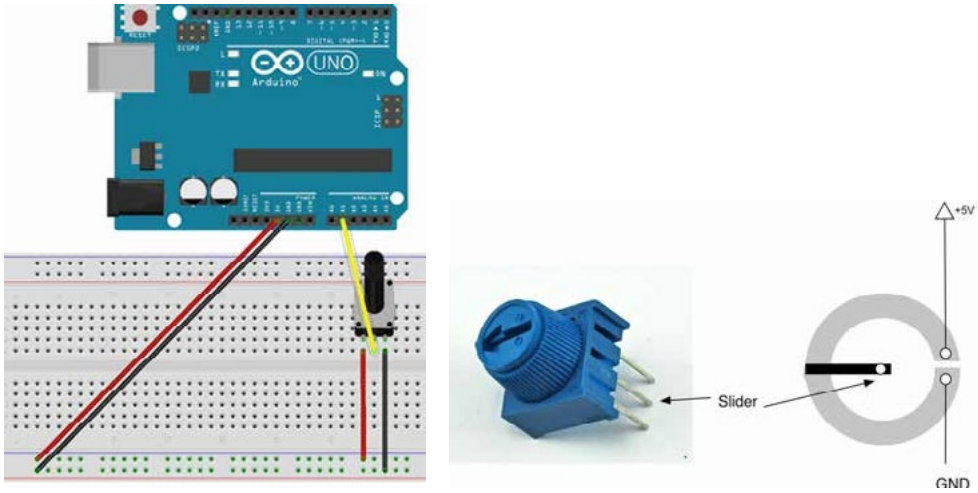
Tako ako je napon na njegovim krajevima 5 V, na kliznom kontaktu možemo imati vrijednosti od 0 do 5 V, te ovaj element kao takav predstavlja idealan elektronski element za testiranje analognog ulaza mikrokontrolera.

Naravno, mikrokontroler ne vidi napon na svom ulazu, već preko svog internog ADC (konvertera *analog to digital*) kola pretvara vrijednost napona u digitalnu vrijednost tipa integer.

Pa kako je rezolucija – preciznost našeg analognog ulaza 10-bitna, to se vrijednosti koje funkcija **analogRead** vraća kreću u granicama od 0 do 1023.

Dakle, Arduino će mapirati ulazni napon potenciometra od 0 do 5 V u integer vrijednost od 0 do 1023. To znači vrijednost „1" integera u naponu je 0,0049 V ili 4,9 mV, a to je najmanja promjena koju mikrokontroler može osjetiti.

Testirat ćemo sve do sada rečeno. Koristit ćemo knjižnicu Serial za softverski *debugging*. Spojite shemu na matadoru prema slici ispod.



Slika 59. Potenciometar na ploči matador i kao element

```
     /*
     Upotreba funkcije analogRead za čitanje vrijednosti sirovih podataka
na analognom ulazu mikrokontrolera.
     */
     int analogPin=1;
     int val=0;
     void setup()
     {
      Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps

     }

     void loop()
     {
         val = analogRead(analogPin);// čitaj stanje s A1

         Serial.println(val);        // pošalji podatke na Serial Monitor

     }
```

Pokušajmo sada dobiti i vrijednost napona na našem analognom ulazu. Formula za pretvaranje sirovih podataka na analognom ulazu u napon je jednostavna i glasi:

float voltage = val*(5.0/1023.0);

Ovdje koristimo novi tip podatka *float*. To je tip podatka namijenjen za prikaz decimalnih brojeva.

```
     /*
     Upotreba funkcije analogRead za čitanje vrijednosti sirovih podataka
na analognom ulazu mikrokontrolera i proračun napona.
     */
     int analogPin=1;
     int val=0;
     void setup()
     {
      Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps

     }

     void loop()
     {
         val = analogRead(analogPin);// čitaj stanje s A1

         Serial.println(val);        // pošalji podatke na Serial Monitor









     }
```

Dopišite formulu i dio koda za slanje vrijednosti napona na serijski ekran svakih 250 milisekundi u sljedećem formatu:

Raw data: 512 Voltage: 2.5 (V)

DiY – analogRead

Cilj vježbe:

Uporaba funkcije *analogRead(pin)*

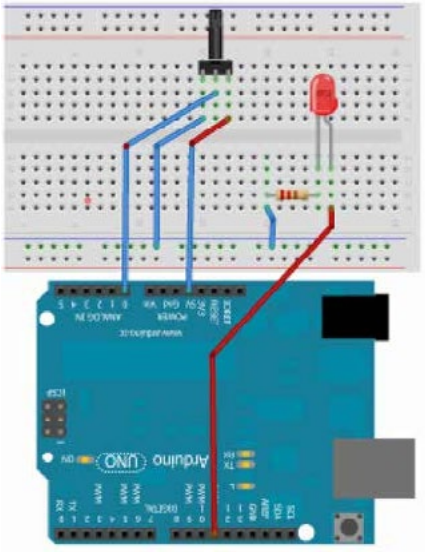
Zadatak vježbe:

Kreirati *Sketch* za upravljanje brojem uključenih LE dioda spram vrijednosti sirovih podataka na analognom ulazu.

0-127 **LED 1 ON**
128-255 **LED 2 ON**
256-383 **LED 3 ON**
384-511 **LED 4 ON**
512-639 **LED 5 ON**
640-766 **LED 6 ON**
767-895 **LED 7 ON**
896-1023 **LED 8 ON**

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. Potenciometar 10 kΩ 1 kom.
3. LE diode 8 kom.
4. Otpornik 330 Ω 8 kom.



Slika 60. Shema spoja

Koraci za realizaciju vježbe:

1. Kreirajte testni sustav koristeći skicu *Fritzing*, vodeći računa da je potrebno spojiti 8 LE dioda.

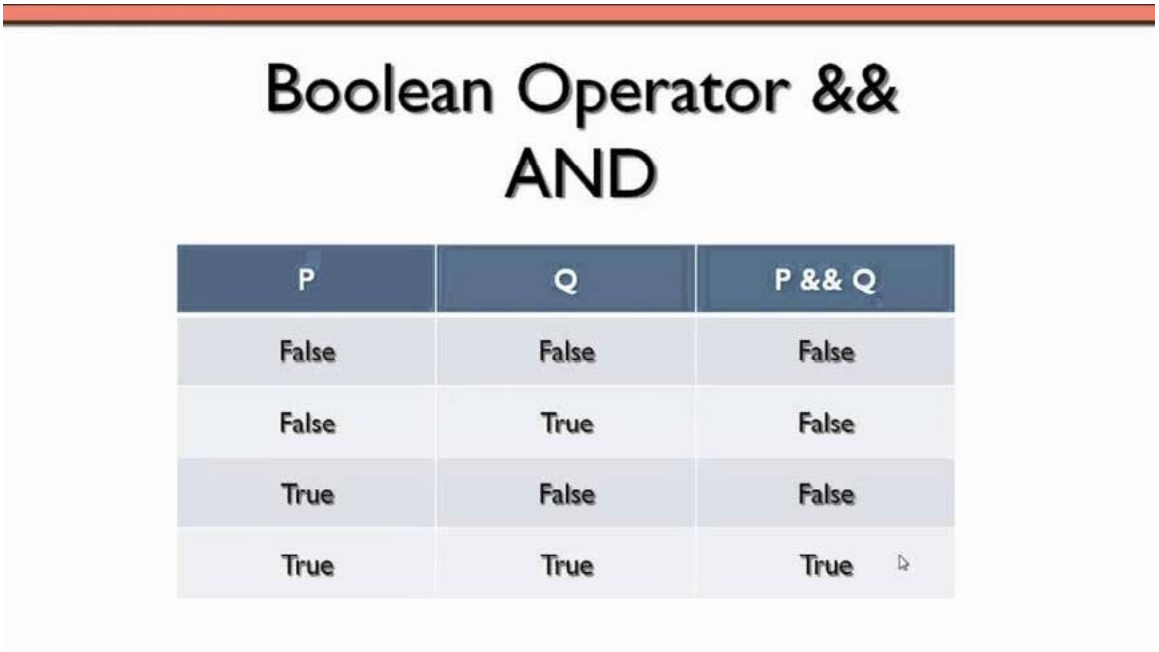
2. Kreirajte *Sketch* kojim ćete čitati analognu vrijednost s analognog pina 0, te vrijednost prikazati u *Serial Monitoru* i spram zadatka vježbe upaliti odgovarajuće LE diode.

3. Zadatak se može riješiti *if* upitima, koristeći operatore prosljeđivanja; upotrijebite *Serial Monitor* za praćenje vrijednosti analognog ulaza.

x == y (x jednako y)
x != y (x nije jednako y)
x < y (x manje od y)
x > y (x veće od y)
x <= y (x manje ili jednako od y)
x >= y (x veće ili jednako y)

Također za rješenje zadatka trebate koristiti i *Boolean* ili logičke operatore

&& (logical and)

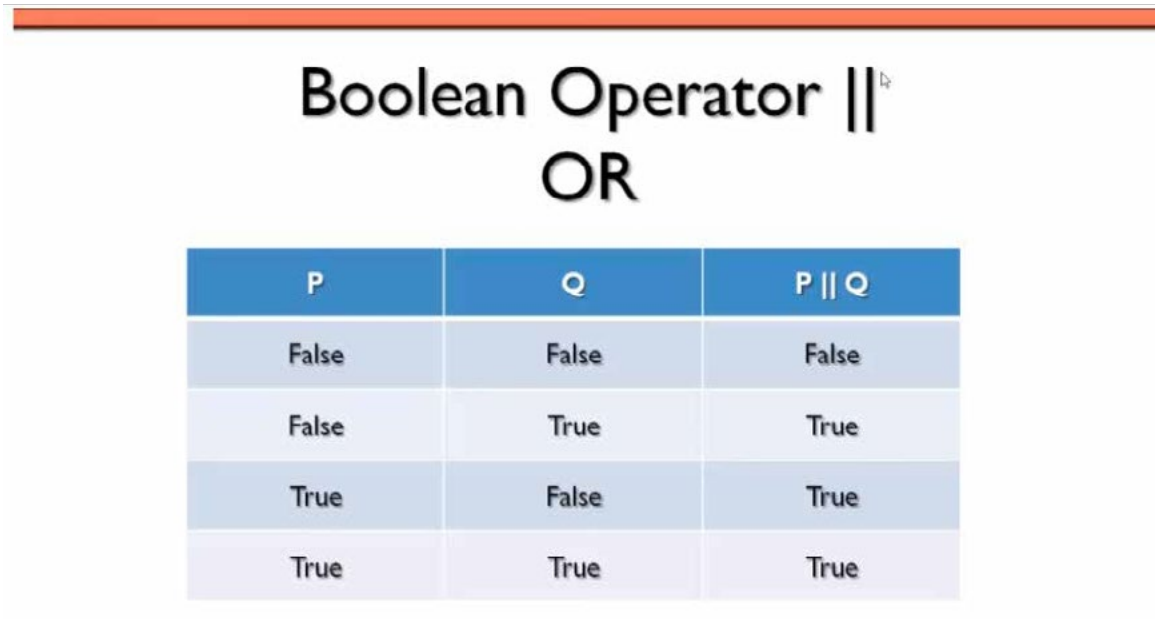


Logički AND ili logički „i" operator zahtijeva da oba iskaza budu istinita da bi rezultat upita bio istinit. Npr.:

```
if (val1 = = 100 && val2 = = 200)
    {
        //uradi nešto
    }
```

Dakle, uvjet je ispunjen samo ako je vrijednost varijable val1 = 100 i varijable val2 = 200. Za sve ostale uvjete nećemo izvršiti ono što je unutar if uvjeta.

|| (logical or)



Logički OR ili logički „ili" operator zahtijeva da bilo koji iskaz ili oba iskaza budu istiniti da bi rezultat upita bio istinit. Npr:

```
if (val1 == 100 || val2 == 200)
    {
        //uradi nešto
    }
```

Dakle, uvjet je ispunjen ako je vrijednost varijable val1 = 100 ili varijable val2 = 200 ili ako su obje varijable točne.

Prepiši rješenje:

```
        int analogPin=1;
        int val=0;
        void setup()
        {
        for(int i =2;i<=10,i++)
        {
            pinMode(i, OUTPUT);
        }
         Serial.begin(9600);              //Koristi komunikaciju na brzini
    od 9600 bps

        }
        void loop()
        {
            val = analogRead(analogPin);// čitaj stanje s A1

            Serial.println(val);        // pošalji podatke na Serial Monitor

            if ((val>=0)&&(val<=127))
            {
                  digitalWrite(2, HIGH);
            }
            else
            {
                  digitalWrite(2, LOW);
            }


        }
```

Cilj vježbe:

Uporaba funkcije *analogRead()* i *serialPrintln()*.

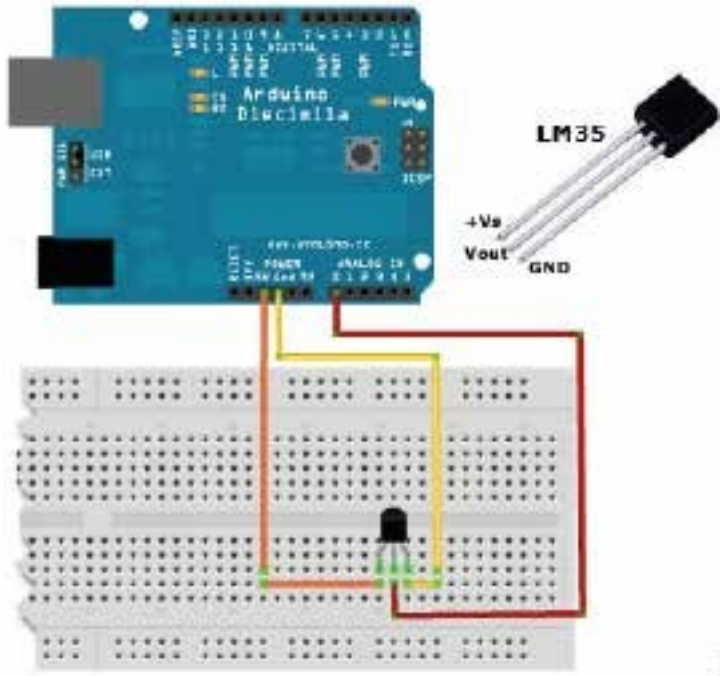
Zadatak vježbe:

Kreirati *Sketch* za proračun temperature.

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. LM35 1 kom.

Shema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sustav koristeći skicu iznad.

2. Kreirajte *Sketch* kojim ćete čitati analognu vrijednost s analognog pina 0, te vrijednost prikazati u *Serial Monitoru*. Koristeći jednadžbu za proračun temperature izračunati temperature u °C, K i F te ih prikazati u *Serial Monitoru*.

3. Ispisati temperature u *Serial Monitoru* u sljedećem formatu:

Temperatura °C -> 23.5
Temperatura K -> 296.65
Temperatura °F -> 74.3

Osnovne teorijske postavke:

Temperatura temperaturnog senzora LM35 može se izračunati prema sljedećem izrazu:

T = (Vcc * ADC * 100.0) / 1024;
Celsius = Kelvin - 273.15
Celsius = 5/9 x (Fahrenheit -32)

Prepiši rješenje:

```
      int analogPin=0;
      int val=0;
      void setup()
      {
      Serial.begin(9600);              //Koristi komunikaciju na brzini od
  9600 bps

      }
      void loop()
      {
          val = analogRead(analogPin);// čitaj stanje s A1

          Serial.println(val);        // pošalji podatke na Serial Monitor
      // primijeniti formule za temperaturu i ispisati ih
```

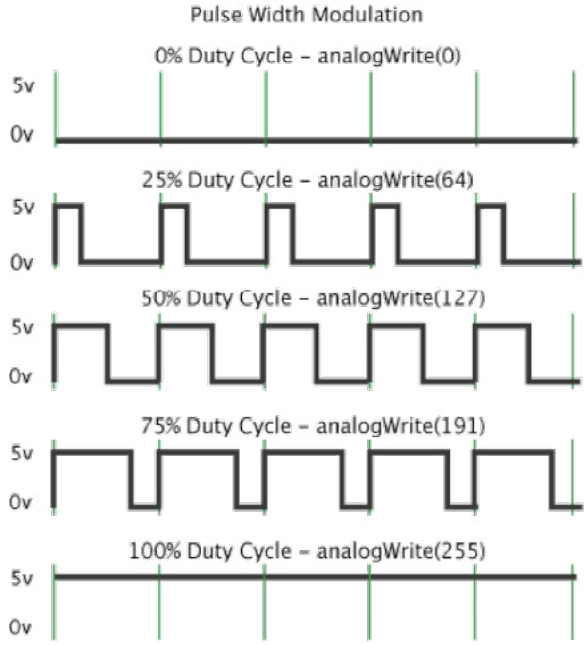
PWM – analogWrite

Vjerojatno ste imali priliku da na raznim rasvjetnim tijelima vidite učinak zvani „fading" ili postepeno gašenje. Taj učinak možemo realizirati i s Arduinom jer on ima 6 PWM pinova (PWM je skraćenica od *Pulse Width Modulation*), a to je tehnika za dobivanje analognih rezultata uz pomoć digitalnog signala.

Naime, PWM podrazumijeva da na izlazu mikrokontrolera generiramo pravokutni impuls, te na taj način možemo simulirati napone od 0 do 5 V.

Vrijeme „on time" signala zove se širina impulsa. Da bismo dobili različite analogne vrijednosti, na izlazu mikrokontrolera mijenjamo širinu tog impulsa odnosno njegovo trajanje. Ako takav signal ponavljamo dovoljno brzo te ga primijenimo na spojenu LED, rezultat će biti promjena intenziteta svjetlosti na LED.

Na grafikonu ispod prikazan je PWM signal. Razdoblje (T) je označeno zelenom bojom i traje 2 milisekunde, pa je frekvencija f = 1/T, 500 Hz.



Slika 61. PWM signal

Za nas je dovoljno da znamo da se PWM signal može generirati funkcijom

analogWrite(pin, value);

Jako je važno napomenuti da je tu funkciju moguće koristiti samo na posebno označenim pinovima Arduina i to sa sljedećim oznakama „~" , „#" ili samo „PWM".

Funkcija ima dva argumenta. Pin na kojem želimo da generiramo PWM signal i *value*, koja može biti u intervalu od 0 do 255. Naš PWM signal ima 8-bitnu rezoluciju pa je 28 = 256.

Testirat ćemo do sada naučeno na sljedećem primjeru. Izaberite željeni PWM pin, spojite predotpor i LED te testirajte naredni kod.

```
/*
Upotreba funkcije analogWrite za kreiranje PWM signala
*/
int pwmPin = 5;

void setup()
{
 Serial.begin(9600);              //Koristi komunikaciju na brzini
od 9600 bps

}

void loop()
{
    analogWrite(pwmPin, 0);   // ugasi LED
    delay(1000);

    analogWrite(pwmPin,64);   // 25% intenziteta
    delay(1000);

    analogWrite(pwmPin,128);        // 50% intenziteta

        delay(1000);

    analogWrite(pwmPin,192);        // 75% intenziteta
        delay(1000);

    analogWrite(pwmPin,255);        // 100% intenziteta

        delay(1000);
```

Pratite promjenu na LED, ona postepeno pojačava intenzitet svjetlosti koju isijava. Idemo malo vježbati.

DiY – analogWrite

Cilj vježbe:

Uporaba funkcije *analogWrite()*

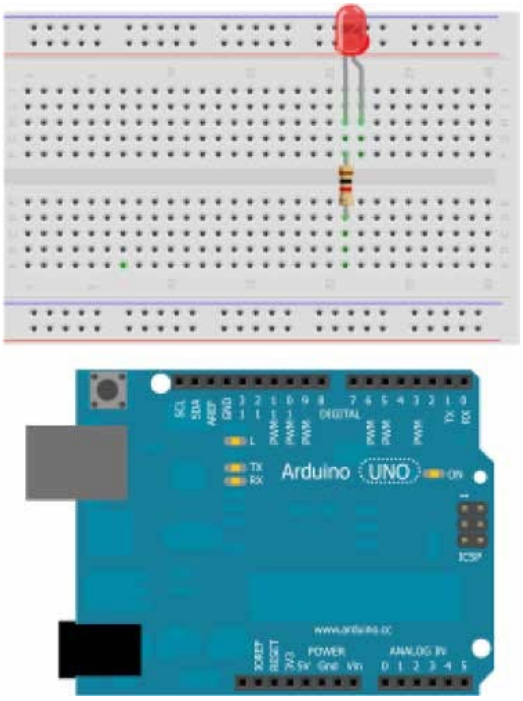
Zadatak vježbe:

Kreirati *Sketch* za upravljane radom LE diode s učinkom *fade in* i *fade out*.

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. LE diode 1 kom.
3. Otpornik 220 Ω 1 kom.

Shema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte shemu spoja koristeći skicu *Fritzing*.

2. Kreirajte *Sketch* kojim ćete upravljati radom LE diode na taj način da će se u jednakim vremenskim intervalima pojačavati intenzitet svjetlosti LE diode do maksimalne vrijednosti (255) i obrnuto do (0), sve s inkrementom i dekrementom od 1.

3. Zadatak je moguće riješiti uporabom for petlje. Obavezno unutar for petlje pozvati funkciju *delay()*.

```
for(int i=0;i<=255;i++)
        {
        analogWrite(pin, Value);
        delay(50);
        }
```

Prepisati rješenje:

```
/*
Upotreba funkcije analogWrite za kreiranje PWM signala
*/
int pwmPin = 5;

void setup()
{

}

void loop()
{
```

Cilj vježbe:

Uporaba funkcije *analogWrite()*

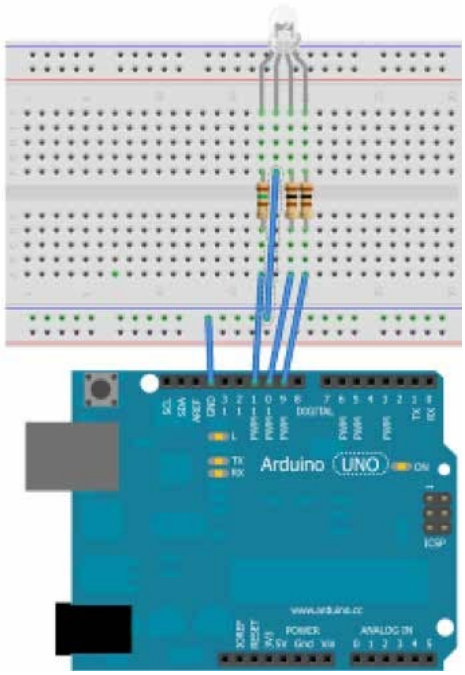
Zadatak vježbe:

Kreirati *Sketch* za upravljane radom RGB LED s učinkom *fade in* i *fade out*.

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom.
2. RGB LED 1 kom.
3. Otpornik 150 Ω 1 kom.
4. Otpornik 100 Ω 2 kom.

Shema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sustav koristeći shemu spoja Fritzing.

2. Kreirajte Sketch kojim ćete upravljati radom LE diode na taj način da će se u jednakim vremenskim intervalima pojačavati intenzitet svjetlosti RGB LED pojedinačno do maksimalne vrijednosti (255) i obrnuto do (0).

Prepisati rješenje:

```
/*
Upotreba funkcije analogWrite za kreiranje PWM signala
*/
int pwmPin = 5;

void setup()
{

}

void loop()
{
```

Zaključak

Iskreno se nadamo da Vam je priručnik razumljiv. Ako ste pomno pratili ovaj priručnik, polako ćete u svakodnevnom životu početi primjećivati mikrokontrolerske sustave, a možda čak i razumjeti kako oni funkcioniraju.

Nadam se da ćete pokušati napraviti i neki svoj mikrokontrolerski sustav, jer je kraj ovog priručnika u suštini samo novi početak. Ne žurite da odmah pravite „svemirsku" aplikaciju, krenite laganim koracima, za početak pokušajte naučiti sve senzore iz seta Arduino.

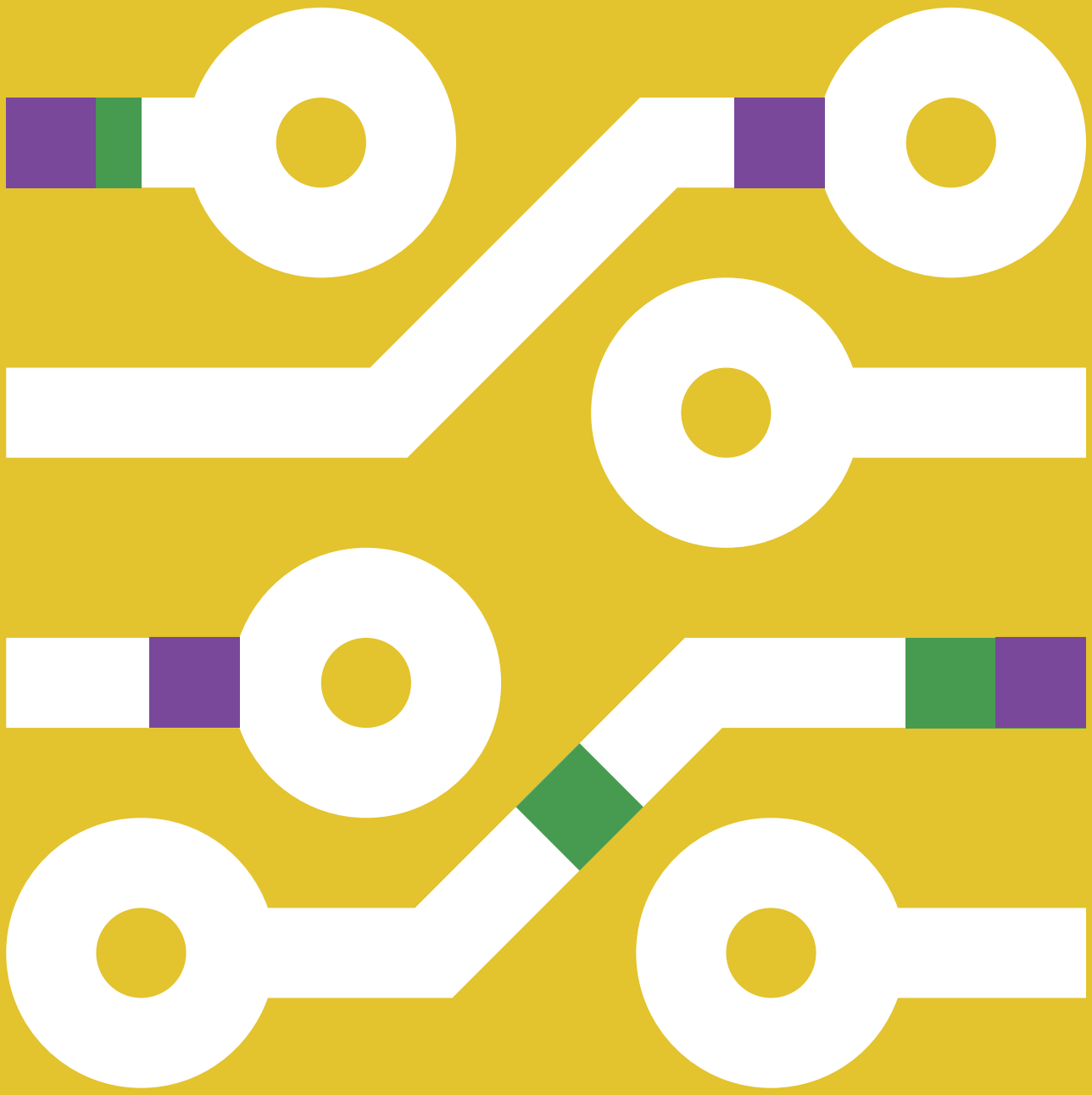
Pokušajte uvidjeti s kojim se problemom srećete svakodnevno. Razmišljajte je li moguće za rješenje vaših problema osmisliti neki mikrokontrolerski sustav.

Pokušajte, pa nemali broj ljudi upravo se tako i obogatio. Naravno, ne treba Vam biti samo to vodilja. Samo rješavanje problema i poslije bezbroj pokušaja pronađeno je rješenje nagrada samo za sebe. I, naravno, ne zaboravite uživati u svemu tome.

Mr. sc. Muamer Halilović, dipl. ing. el.

Literatura

1. https://learn.adafruit.com/
2. https://www.Arduino.cc/en/Tutorial/HomePage
3. https://www.instructables.com/technology/Arduino/
4. https://www.makerspaces.com/Arduino-uno-tutorial-beginners/



ARDUINO ZA SVE

DODATAK PRIRUČNIKU ZA NASTAVNIKE
Osnovna škola

Podržano od:

< IT Girls >

UJEDINJENE NACIJE
BOSNA I HERCEGOVINA

LABORATORIJSKA VJEŽBA BR.1

ARDUINO UNO – LM35

UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....

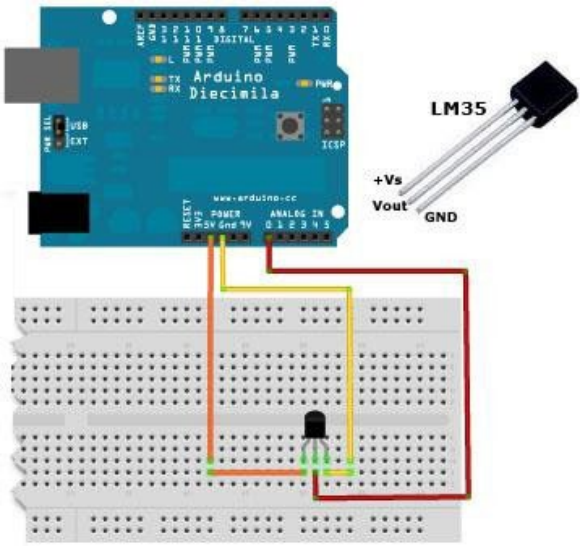
Cilj vježbe:
Uporaba funkcija analogRead(), analogWrite() , serialBegin() i serialPrintln()

Zadatak vježbe:
Kreirati Sketch za proračun temperature s analognog izlaza LM35 senzora.

Potrebni elementi za vježbu:

1. Arduino Uno 1 kom.
2. LM35 1 kom.
3. Ploča matador 1 kom.

Shema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sustav koristeći skicu *Fritzing*.

2. Kreirajte *Sketch* kojim ćete čitati analognu vrijednost s analognog pina 0 te vrijednost prikazati u *Serial Monitoru*. Koristeći jednadžbu za proračun temperature izračunati temperature u oC, K i F te ih prikazati u Serial Monitoru.

Osnovne teorijske postavke:

Temperatura temperaturnog senzora LM35 može se izračunati prema sljedećem izrazu:

T = (Vcc * ADC * 100.0) / 1024;
Celsius = Kelvin - 273.15 Celsius = 5/9
x (Fahrenheit -32)

analogRead()

Opis
Čita vrijednost sa definsanog analognog pina (10-bitni A/D konverter). Ulazni napon od 0 do 5 V mapira se u integer vrijednost od 0 do 1023.

Sintaksa
analogRead(pin)
Parametri
pin: broj analognog pina (0 do 5)
Vraća (return)
int (0 do 1023)

IT Girls - Arduino za sve

Rješenje:

```
/*
 * Zadatak ove vježbe je uporaba funkcije analogRead()
 * te uporaba tvz. arduino softverskog debugera
 * za očitanje i prikaz temperature prostorije
 */

int val=0; //deklaracija Integer tipa varijable
           //u koju ćemo spremiti sirove podatke s
           //analognog ulaza
float T=0; //Varijabla za spremanje vrijednosti temperature

void setup()
{
  Serial.begin(9600);// koristit ćemo app Serial Monitor za provjeru
temperature
  Serial.println("IT-Girls->2019");//pošalji Serial Monitor app
string "IT -Girls"
}

void loop()
{
  val = analogRead(A0);// pročitaj sirovi podatak s pina A0
                       // moguće vrijednosti su: 0-1023

 T=(5*val*100)/1023;  //formula za proračun temperature za LM35
senzor

Serial.print("Trenutna temperatura:")
 Serial.print(T);
  Serial.println("°C");
}
```

Zaključak:

LABORATORIJSKA VJEŽBA BR.2

ARDUINO UNO – 2x16 LCD zaslon

UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....

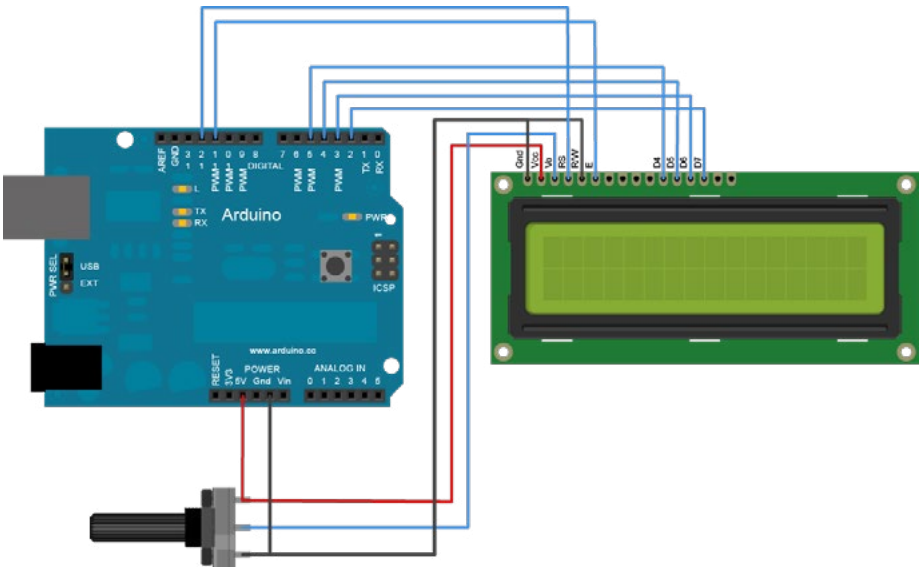
Cilj vježbe:
Uporaba 2x16 LCD zaslona, način spajanja

Zadatak vježbe:
Kreirati *Sketch* za prikaz teksta.

Potrebni elementi za vježbu:

1. Arduino Uno 1 kom.
2. 2x16 LCD zaslon 1 kom.
3. Potenciometar 10KΩ 1 kom.
4. Ploča matador 1 kom.

Shema spoja:



Kreirajte testni sustav koristeći skicu FRITZING.

1. Kreirajte *Sketch* u kojemu ćete pozvati knjižnicu LiquidCrystal.h naredbom *include.h* i koristeći HELP upoznati se s funkcijama potrebnim za ispis teksta na LCD zaslon.

2. U prvom redu napisati svoje ime, a u drugom ime škole ili razred i odjeljenje.

IT Girls - Arduino za sve

Rješenje:

```
/*
 * Primjer korištenja 2x16 lcd zaslona
 * za prikaz informacija korisniku
 */

//poziv knjižnice za LCD

#include <LiquidCrystal.h>

//kreiranje novog LCD objekta -> s konfiguracijom kao u argumentima
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  // nas LCD ima 16 stupaca i dva
  reda lcd.begin(16,2);

  //postavi kursor na prvo polje
  lcd.setCursor(0,0);

  //printaj IT Girls
  lcd.print("IT - Girls :D");
  delay(1000);
}

void loop()
{
  lcd.setCursor(0,1);
  lcd.print("Sarajevo 2019");
  }
```

Zaključak:

LABORATORIJSKA VJEŽBA BR.3

ARDUINO UNO – 2x16 LCD zaslon i LM35

UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....

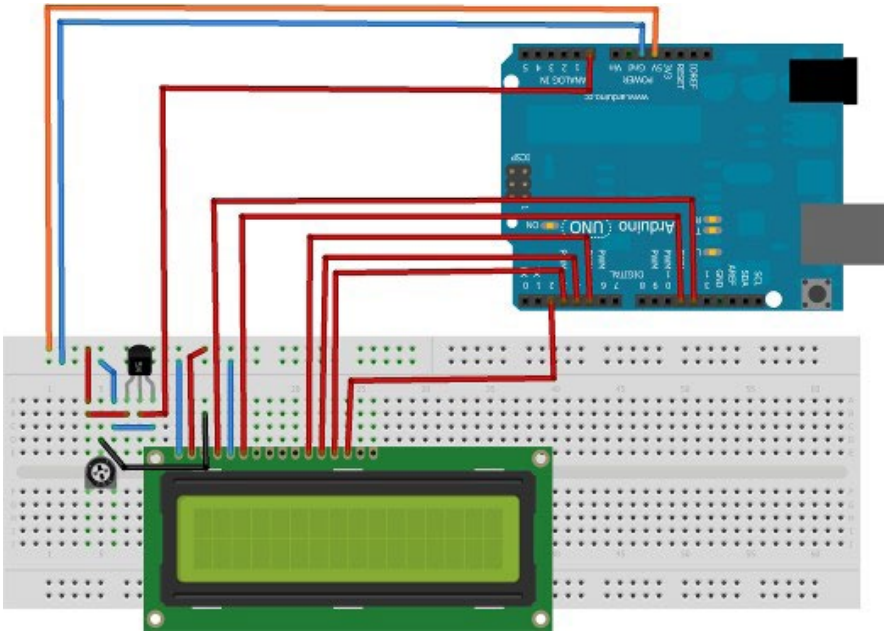
Cilj vježbe:
Uporaba 2x16 LCD zaslona, način spajanja

Zadatak vježbe:
Kreirati *Sketch* za prikaz temperature.

Potrebni elementi za vježbu:

1. Arduino Uno 1 kom.
2. 2x16 LCD zaslon 1 kom.
3. Potenciometar 10KΩ 1 kom.
4. LM35 1 kom.
5. Ploča matador 1 kom.

Shema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sustav koristeći skicu *Fritzing*.

2. Kreirajte *Sketch* u kojemu ćete pozvati knjižnicu LiquidCrystal.h naredbom include.h

3. Kreirati app za prikaza temperature u oC u prvom redu, a u drugom vrijednost napona na analognom izlazu A0. Kreirati tablicu vrijednosti napona i odgovarajuće temperature.

IT Girls - Arduino za sve

Rješenje:

```
                                        /*
 * Primjer korištenja 2x16 LCD zaslona
 * za prikaz informacija korisniku
 */

//poziv knjižnice za LCD

#include <LiquidCrystal.h>
int val=0;
float T=0;
//kreiranje novog LCD objekta -> s konfiguracijom kao u argumentima
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  // nas LCD ima 16 stupaca i dva
  reda lcd.begin(16,2);

  //postavi kursor na prvo polje
  lcd.setCursor(0,0);

  //printaj IT Girls
  lcd.print("IT - Girls :D");
  delay(1000);
}

void loop()
{
   val = analogRead(A0);

   T=(5*val*100)/1023;

  lcd.setCursor(0,1);
  lcd.print("Temp:");

  lcd.setCursor(5,1);
  lcd.print(T);

  lcd.setCursor(12,1);
  lcd.print("°C");

  delay(1000);

}
```

Zaključak:

LABORATORIJSKA VJEŽBA BR.4

ON-OFF SENZORI

UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....

Jednostavni ON-OFF senzori su uređaji koji mjere neku fizičku veličinu i kao izlaz daju digitalno stanje – uključeno ili isključeno.

Senzor zvuka aktivira se kada detektira jačinu zvuka veću od one podešene s pomoću potenciometra na njemu.

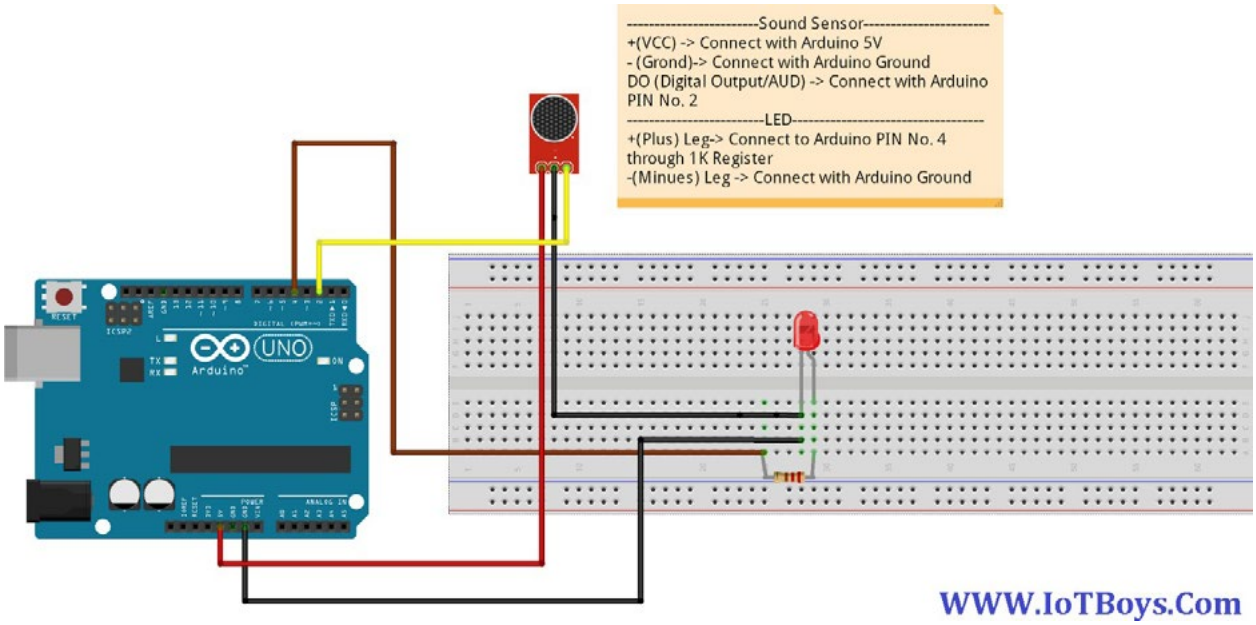
Zadatak

Napraviti sklop za uključivanje i isključivanje svijetleće diode zvukom. Svaki put kada kraj senzora pljesnete npr. rukama, neka se dioda uključi ako je bila isključena, odnosno isključi ako je bila uključena.

Potrebne komponente:

- Pločica Arduino UNO i USB
- Pločica matador
- Senzor zvuka
- 1x crvena LED
- 1x otpornik 220 Ω

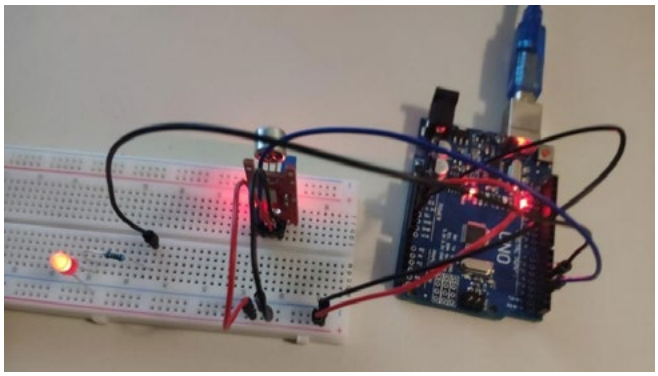
Prikaz spajanja



Rješenje:

```
  int soundSensor=2;
int LED=4;
boolean LEDStatus=false;

void setup()
{
 pinMode(soundSensor,INPUT);                    //postavi izvod SenzorZvuk (2)
kao ulazni
pinMode(LED,OUTPUT);                              //postavi izvod LedCrvena (4)
kao izlazni
}
void loop()
{
  int SensorData=digitalRead(soundSensor);
  if(SensorData==1)                                 //ako je detektiran zvuk
  {
    if(LEDStatus==false)
    {
       LEDStatus=true;
       digitalWrite(LED,HIGH);
    }
    else
    {
       LEDStatus=false;
       digitalWrite(LED,LOW);
    }
  }
 }
```

Zaključak:



LABORATORIJSKA VJEŽBA BR.5

ON-OFF SENZORI

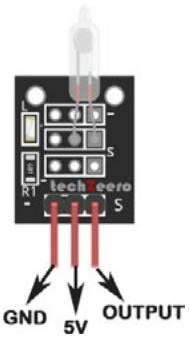
UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....



Senzor nagiba radi na principu dva kontakta i možemo grubo odrediti je li objekt postavljen uspravno/vodoravno na Zemlju ili nije.

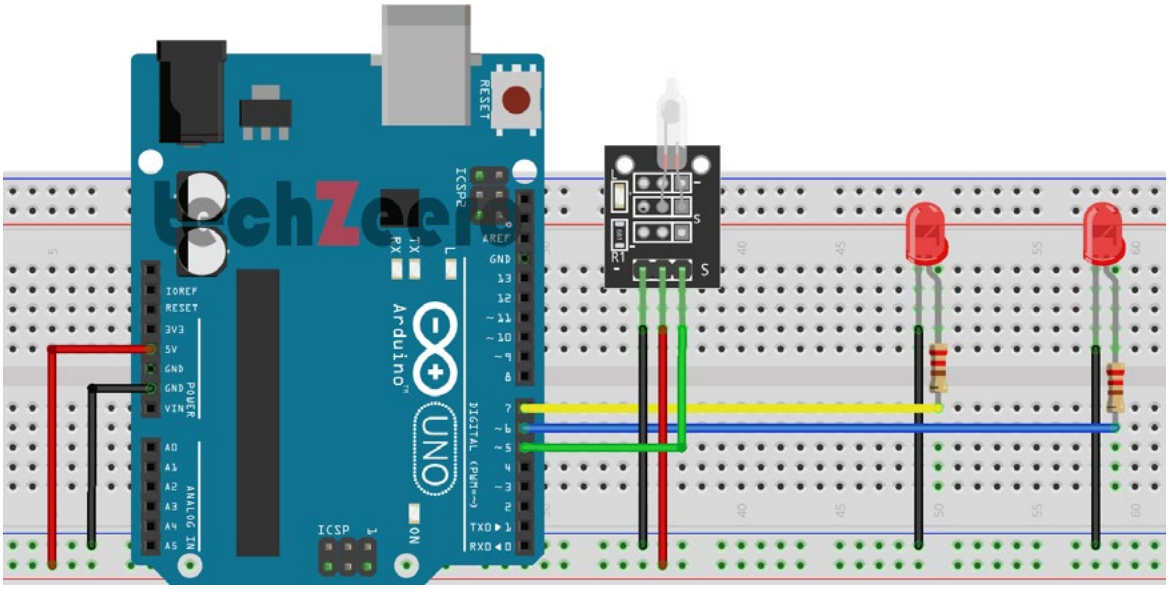
Zadatak:

Napraviti sklop za uključivanje i isključivanje svjetlećih dioda zavisno o položaju kuglice unutar tilt senzora. Neka oba svjetla budu uključena kada se kuglica nalazi u položaju ravnoteže. Kada je položaj tilt senzora uspravno, uključuje se jedno svjetlo, a kada je vodoravno, uključuje se drugo svjetlo.

Potrebne komponente:

- Pločica Arduino UNO i USB
- Pločica matador
- Senzor nagiba
- 1x crvena LED
- 1x zelena LED
- 2x otpornik 220 Ω

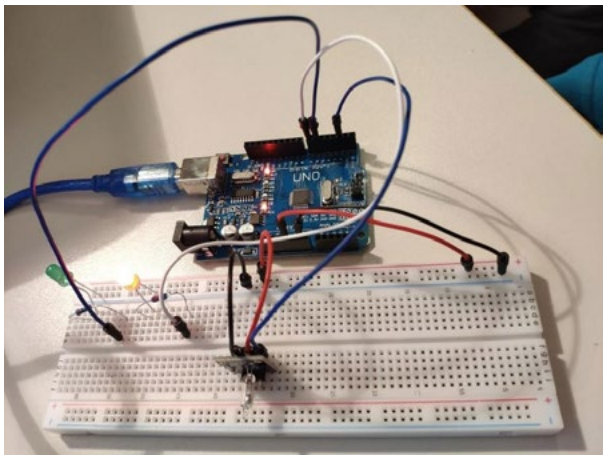
Prikaz spajanja:

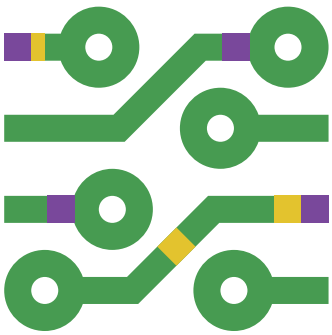


Rješenje:

```
  int tiltSensorPin = 2;          //Pin za senzor nagiba int
notTiltLED = 7;
int TiltLED = 6;

void setup()

{
    pinMode(tiltSensorPin, INPUT);

    digitalWrite(tiltSensorPin, HIGH);

    pinMode(notTiltLED, OUTPUT);
    pinMode(TiltLED, OUTPUT);
}
void loop()
{
  if(digitalRead(tiltSensorPin))
  {

    digitalWrite(notTiltLED, HIGH);
    digitalWrite(TiltLED, LOW);
  }
  else
  {
    digitalWrite(notTiltLED, LOW);
    digitalWrite(TiltLED, HIGH);
  }
}
```

Zaključak:





2019.