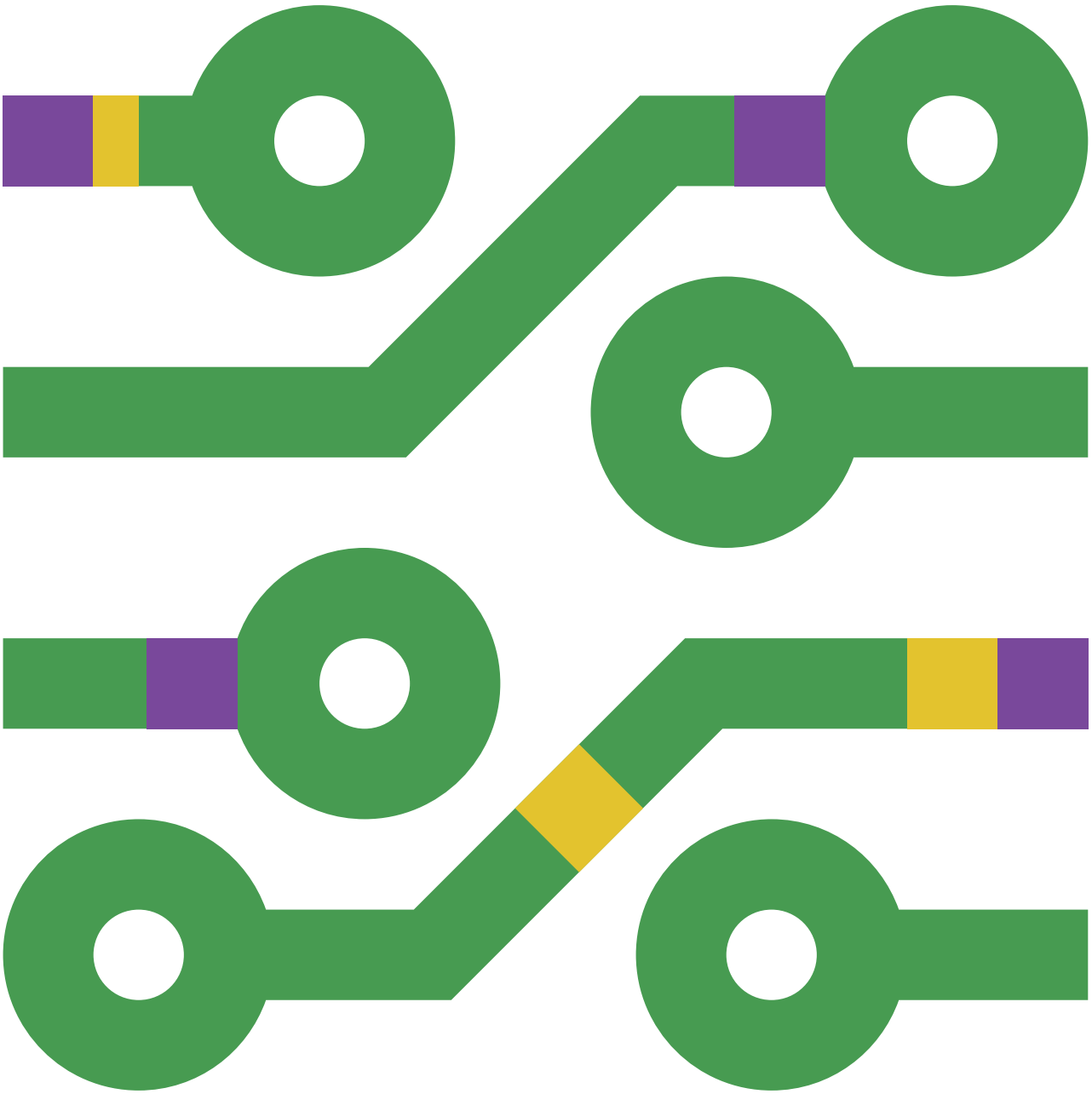


ARDUINO ZA SVE

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 da krene. 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 učenici 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 mikroračunara.

Naši životi se 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 novog doba.

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

Arduino projekat je počeo 2003. godine kao program za studente na Interaction Design Institute Ivrea u Ivrea, Italija. Cilj je bio kreirati jeftinu i jednostavnu platformu za izučavanje i kreiranje mikrokontrolerskih sistema kako za početnike tako i za profesionalce. Platforma je osmišljenja 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 milionski, a broj onih kloniranih vjerovatno nikada nećemo ni saznati. Dakle, konačno smo u prilici da uđemo u jedan novi svijet u kojem je bukvalno samo nebo granica.

Mr.sci. Muamer Halilović, dipl.ing.el

Arduino – Main chip

Prije nego što uđemo u svijet mikrokontrolera, moramo napraviti poređenje između „mozga" Arduino platforme i klasičnog CPU-a iz desktop ili laptop računara.

Arduino platforma je izrađena oko ATMEL ATMEGA 328p kontrolera čije karakteristike su sljedeće:

- 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 Arduino platformi i van nje

Moderni PC vrlo vjerovatno ima CPU 6, 7. ili 8. generacije, a za poređenje 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 računara danas ima HDD s minimalno 250 GB
- Nema interni RAM, ali većina računara 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 računara, koji košta nekoliko stotina konvertibilnih maraka. Jednostavno nema smisla.

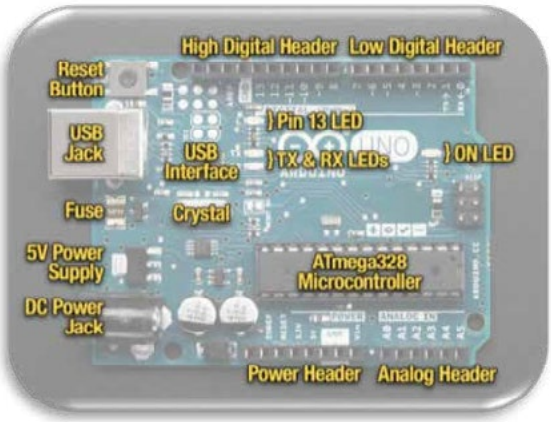
Mora se naglasiti još jedna osnovna razlika između PC računara i mikroračunara, a to je da na PC-u nema ograničenja za instalaciju aplikacija. Kad nestane HDD prostora, uradi se *upgrade* i doda novi HDD. Dok kod mikroračunara imamo samo jednu aplikaciju, a to je ona za koju je mikroračunarski sistem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

Arduino platforma

Slika 3. Golf MKII

Prosječan Bosanac i Hercegovac vjerovatno bi bez problema mogao navesti sve dijelove omiljenog VW auto- mobila GOLF MKII, poznatijeg na našim prostorima kao „Golf 2".

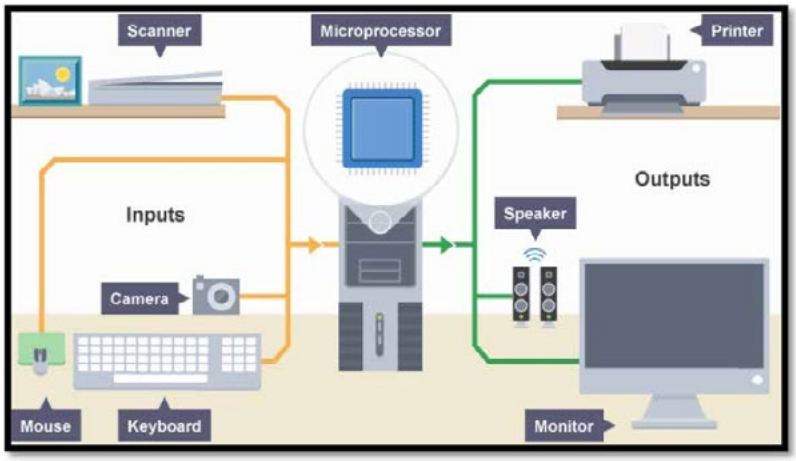
Iako je „Golf 2" proizveden u višestruko manjoj seriji od Arduino platforme, vjerovatno na prste ruke možemo pobrojati naše prosječne sugrađane koji bi isto to mogli uraditi i za Arduino platformu, pa idemo to malo promijeniti.



Slika 4. Arduino platforma

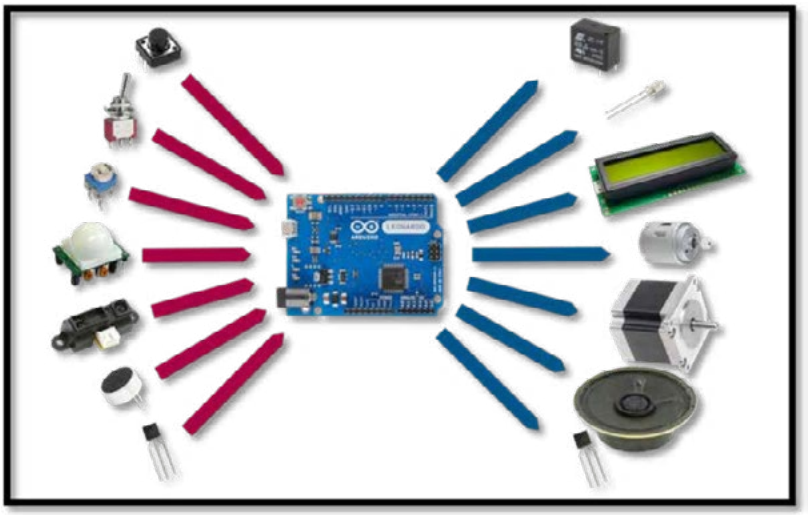
- *USB Jack* – služi za spajanje Arduino mikrokontrolera s računarom, kao interfejs za programiranje i napajanje prilikom testiranja;
- *Reset Button* – resetuje mikrokontroler, odnosno aplikaciju učitanu u njega;
- *Fuse* – osigurač, morao vam je barem jednom kod kuće iskočiti jedan, štiti uređaje od strujnih preop- terećenja;
- *5V Power Supply* – naponski regulator koji limitira napon sa 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* – interfejs (sučelje) koji 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 generiše signal tačne frekvencije – clock mikroračunara.

Senzori i aktuatori



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

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, tastatura, kamera, dok su izlazni uređaji monitor, printer ili zvučnik.



Slika 6. I/O uređaji kod Arduino razvojne platforme

Ulazni uređaji u mikrokontrolerskim sistemima su tasteri, prekidači, potenciometri, digitalni senzori ili analogni senzori. Izlazni uređaji bi bili releji, LED, LCD, motori ili zvučnici. Mikrokontrolerski sistem 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.

Arduino IDE Software

Prije svega prvo moramo podesiti svoj PC računar na taj način da odradimo instalaciju **Arduino IDE Softwarea**. Naravno, riječ je o besplatnoj aplikaciji koju koristimo da bismo kreirali ili razmjenjivali informacije s Arduino mikrokontrolerom. Aplikacija se može preuzeti sa sljedeće web lokacije:

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

Slika 7. Arduino IDE odabir instalacionog fajla

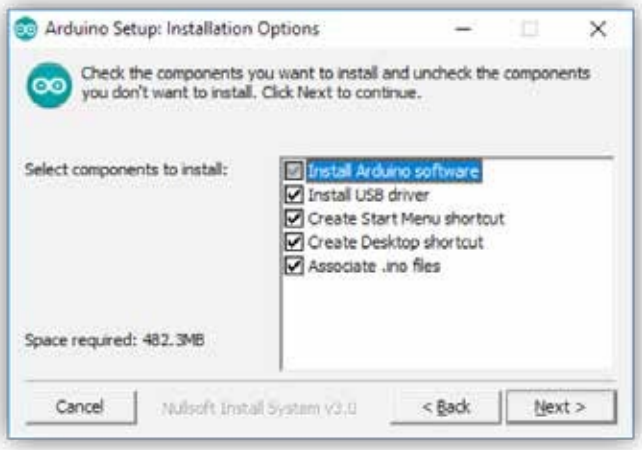
Arduino software se stalno revidira. U trenutku pisanja ovog priručnika aktuelna je verzija 1.8.9, ali je vrlo moguće da dok ovaj priručnik dođe do krajnjih korisnika novija verzija softwarea bude dostupna za instalaciju.

Proći ćemo kroz korake za podešavanje i instalaciju Arduino IDE Softwarea. Kliknite na *Windows installer link* za download instalacionog fajla i potom kliknite na novoj stranici *Just download*.

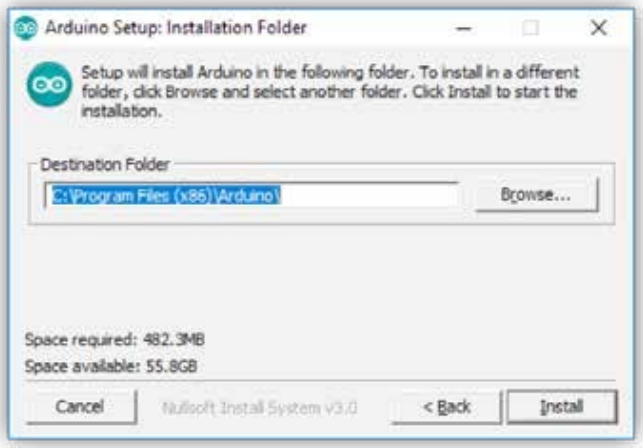
Morate potvrditi da se slažete s opštim uslovima pri upotrebi softwarea koji koristi tzv. *open source* licencu.

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

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

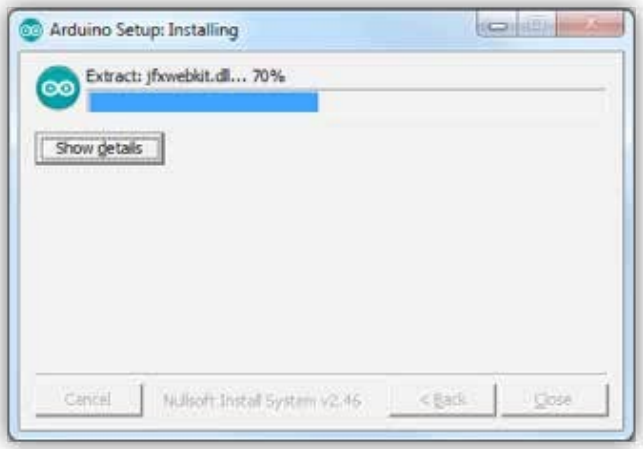


Slika 9. Arduino IDE odabiri instalacije USB drivera



Slika 10. Arduino IDE potvrda lokacije instalacije

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



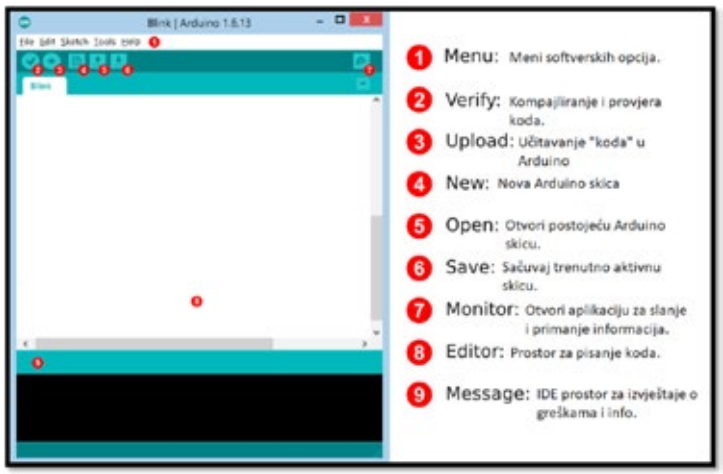
Slika 11. Arduino IDE proces instalacije

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

Slika 12. Arduino IDE ikona

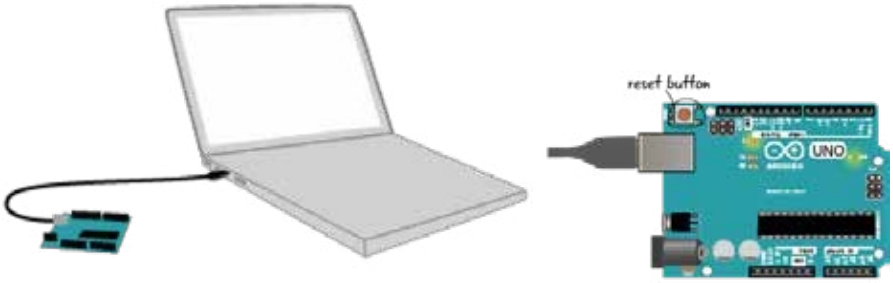
Arduino IDE okruženje

Prije nego što krenemo u programiranje, pojasnit ćemo Arduino IDE okruženje.



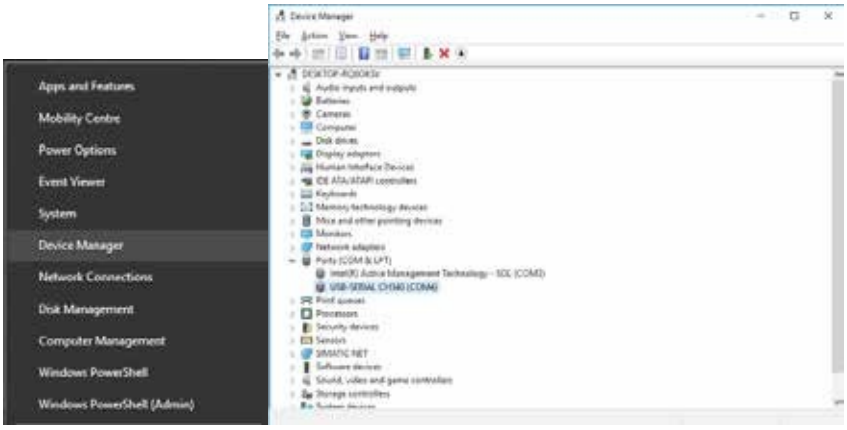
Slika 13. Arduino IDE opis

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



Slika 14. Spajanje Arduina na PC

Dobra rutina je u *Windows Device Menageru provjeriti kojem COM (komunikacijskom interfejsu) portu je pridružen Arduino. Na Win10 o*perativnom sistemu potrebno je desnim klikom na Windows ikoni naći *Device Menager* opciju i pretražiti komunikacijske portove (Ports – COM & LPT).



Slika 15. Arduino na COM4

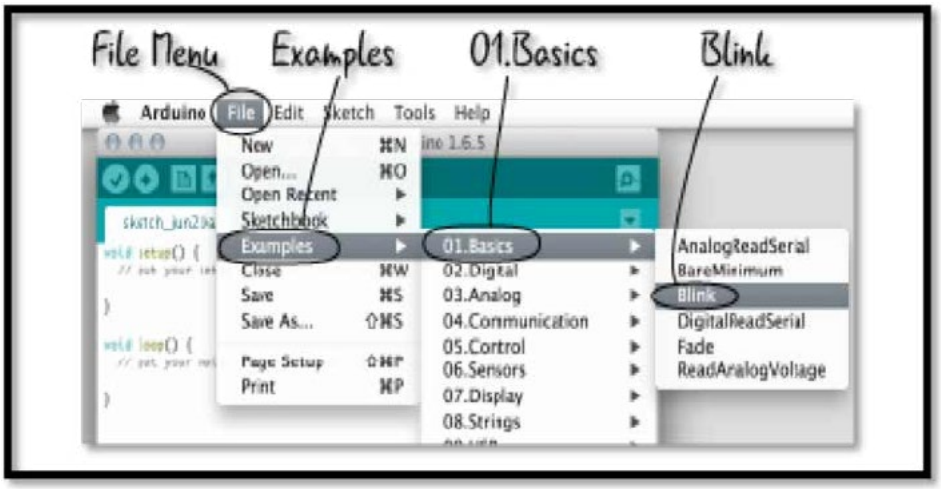
cout << „Hello, World!”;

„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 Arduino platforme 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 ćemo učiti 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 bi bio otvaranje *example Blink* aplikacije, kompajliranje i učitavanje iste u Arduino, te detaljna analiza koda.



Slika 16. Otvaranje Arduino Blink skice

```
/*
 * 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 pokrece, kada se skica startuje
{
  pinMode(ledPin, OUTPUT);      // proglasavanje pina 13 IZLAZOM
}


void loop()                     // stalno se izvršava
{
  digitalWrite(ledPin, HIGH);   // uključi LED
  delay(1000);                  // cekaj sekundu
  digitalWrite(ledPin, LOW);    // iskljuci LED
  delay(1000);                  // cekaj 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 tačkom. 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 visina je____cm
                                    / moja težina je_____kg
```

Sada postaje interesantno. 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 pokrece, kad se skica starta
starts
{
  pinMode(ledPin, OUTPUT);          // proglasavamo pin 13 IZLAZOM
}
```

Naime, mikrokontroler na Arduino razvojnoj platformi ima 13 pinova koji su deklarisani kao Digital inputs/ outputs. Ispravno ih je nazvati GPIO (*general purpose input output* – ili ulazi i izlazi opšte namjene). U *setup* proceduri mi kontroleru damo uputu da ćemo neke pinove koristiti kao INPUTE a neke kao OUTPUTE. Taj dio koda se izvršava 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 tačku na kraju programske izjave.

Sada malo vježbe. Pogledajte setup proceduru 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);      // ukljuci LED na pinu 13
  delay(1000);                     // zadrži stanje 1 sekundu
  digitalWrite(ledPin, LOW);       // iskljuci LED na pinu 13
  delay(1000);                     // zadrži stanje 1 sekundu
}
```

Procedura ***loop()*** se ciklično izvršava i u stvari predstavlja aplikaciju koju korisnik u konačnici vidi. U našem primjeru unutar loop procedure koristimo nove dvije funkcije ***digitalWrite()*** i ***delay().*** DigitalWrite funkcija 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 definisani broj milisekundi.

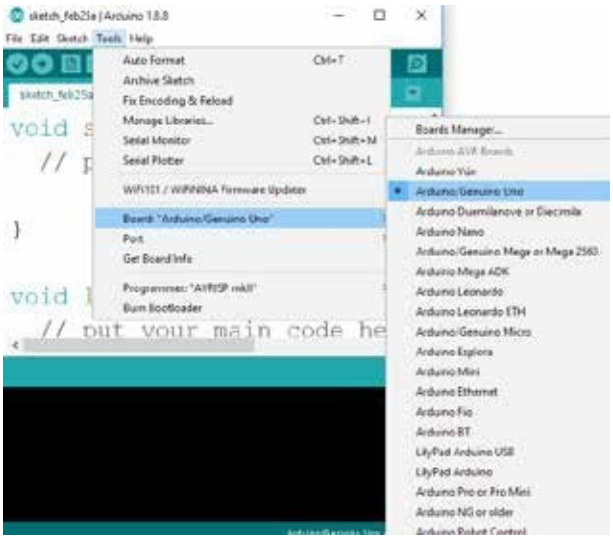
Da ne zaboravimo, procedure su skupovi programskih izjava. Da biste bolje razumjeli procedu re pogledajte sljedeći primjer.

```
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.
}
```

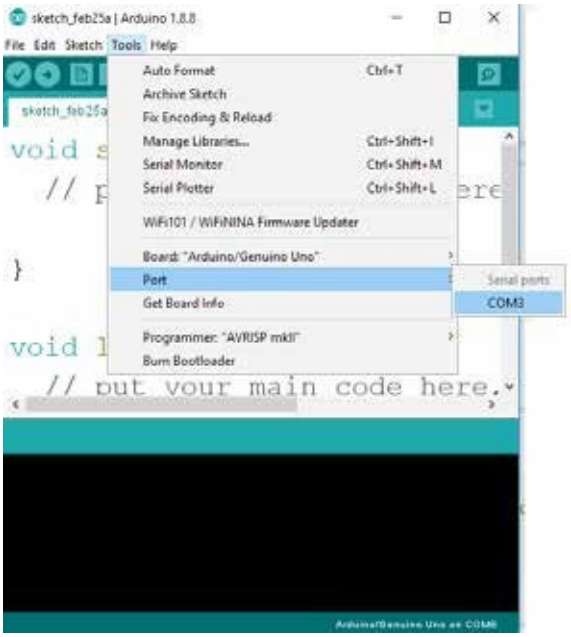
Ime procedure je *operite mačku*. Procedura ima jedan argument, prljavu mačku, koja nakon uspješno izvršenih programskih izjava vraća na kraju procedure čistu mačku. Jednostavno, zar ne!?

Verifikacija i učitavanje koda

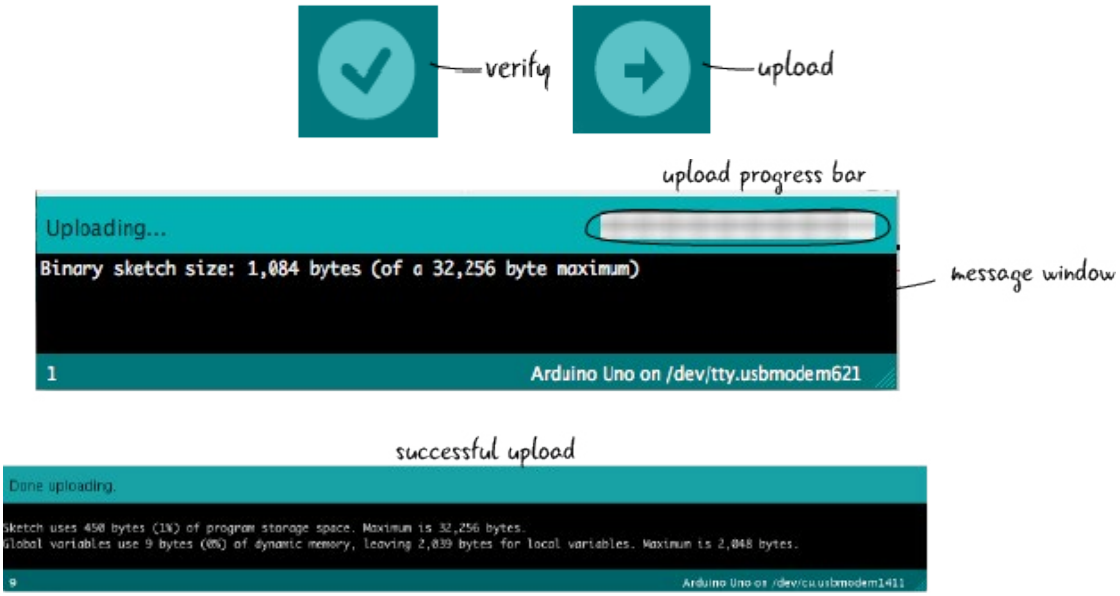
Prije same procedure učitavanja naše aplikacije u mikrokontroler potrebno je u meniju *Tools* odabrati odgovarajuću razvojnu pločicu i pripadajući COM port.



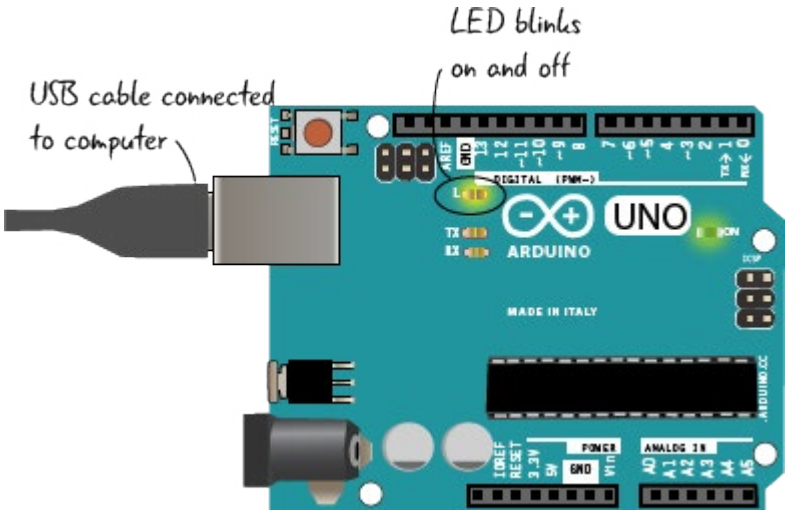
Slika 17. Odabir Arduino razvojne ploče



Slika 18. Odabir pripadajućeg porta



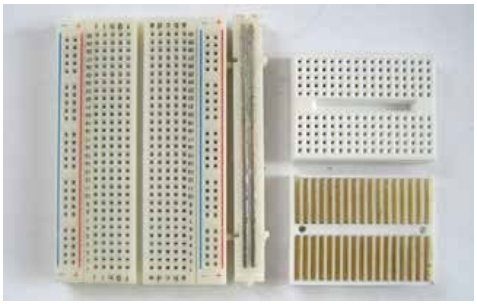
Slika 19. Procedura za verifikaciju i učitavanje koda



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

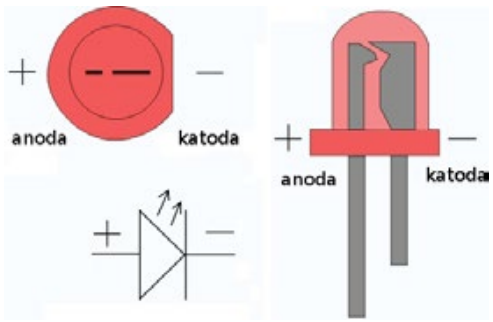
Osnovne elektronske komponente i pomagala

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



Slika 21. Matador ploča pogled odozgo i odozdo

Naredni element koji ćemo koristiti da bi realizovali vježbu je LE dioda ili svijetleća dioda, a to je specijalna vrsta diode koja usljed proticanja struje istu 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 bi po Omovom zakonu dobili sljedeću preporučenu vrijednost otpornika.

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

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

ISHODI UČENJA

(vježbe 1 do 5)

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

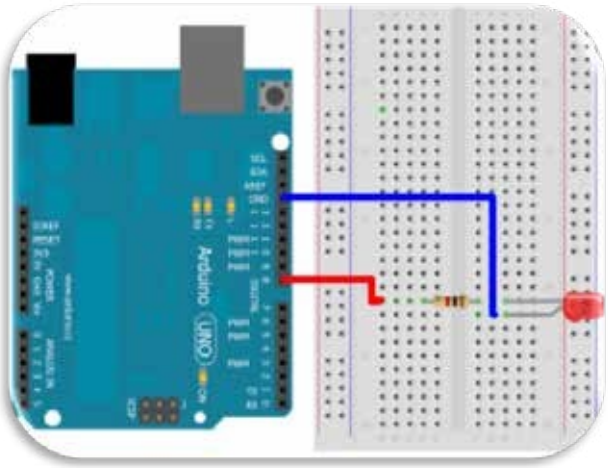
Vježba 1.
NEKA BUDE SVJETLO

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

Potrebne komponente:

- Arduino UNO pločica i USB 1 kom
- Matador pločica 1 kom
- LE diode 1 kom
- Otpornik 220 Ω 1 kom

Slika 23. Potrebne komponente



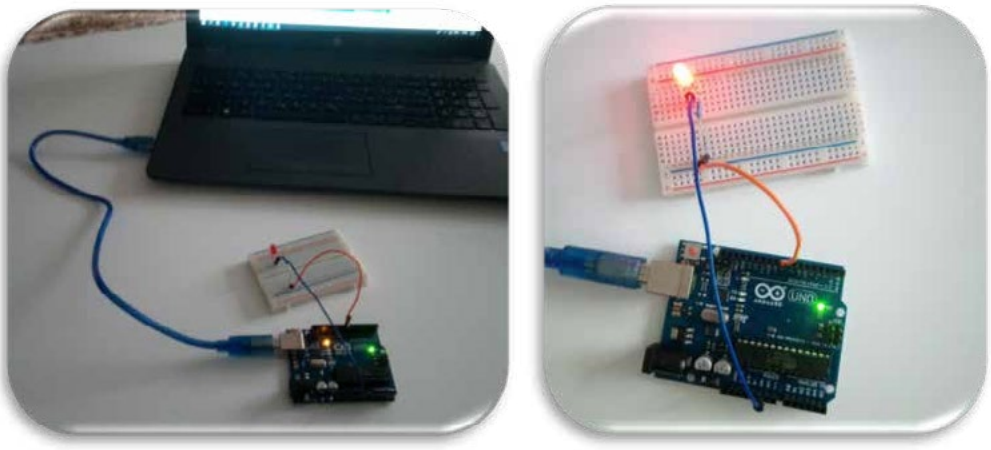
Slika 24. Šema spoja

Koraci za realizaciju vježbe:

1. Povežite Arduino pin 8 sa LE diodom na osnovu priložene šeme 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.

Modifikujete kod da je LED uključena na 100 msec, a isključena na 900 msec.

Primjer 2.

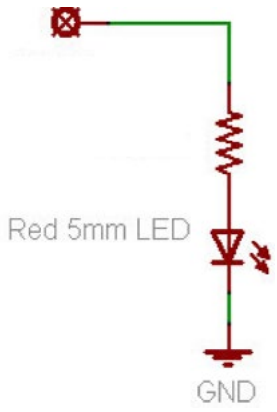
Modifikujte kod na taj način da je LE dioda uključena na 50 msec i isključena na 50 msec. Opišite kako se ponaša LE dioda.

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

Primjer 3.

Modifikujte kod na taj način da je LE dioda uključena i isključena u intervalima od po 10 msec. Opišite pojavu. Pokušajte mahati Arduino razvojnom platformom naprijed–nazad u zamračenoj prostoriji. Šta se dešava?

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

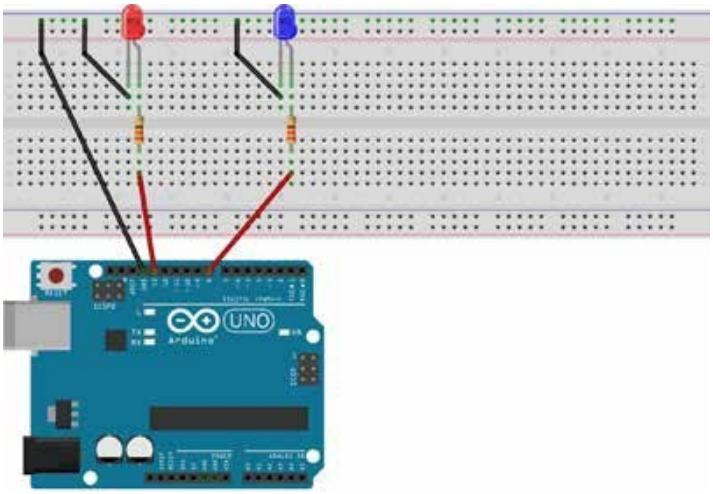


Slika 26. Električna šema kola za spajanje LE dioda na pin 8

Vježba 2.
TOPLO I HLADNO!

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

- Arduino UNO pločica i USB 1 kom
- Matador pločica 1 kom
- LE diode 2 kom
- Otpornik 220 Ω 2 kom



Slika 27. Šema 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
}
```

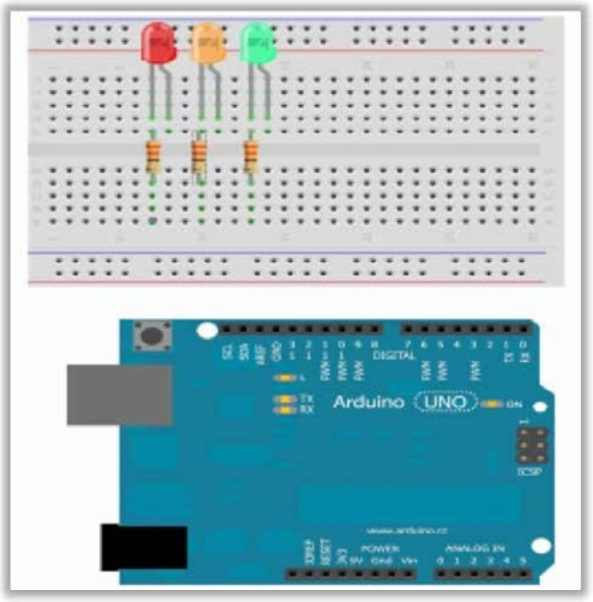
Vježba 3.
SEMAFOR

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

Potrebne komponente za vježbu i šema spoja:

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

Slika 28. Primjer izgleda semafora



Slika 29. Šema 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.
DISCO SVJETLA

Disco lights – 5 lampica i 4 efekta

Kreirati *Sketch* za kontrolu rada LE diode. Ovaj zadatak se može riješiti na više načina. Prvi je 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 ćemo dobiti efekat da se LED pale jedna iza druge sa zadrškom od pola sekunde. Međutim, kako budemo razvijali nove efekte, to će se broj linija koda drastično povećavati.

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

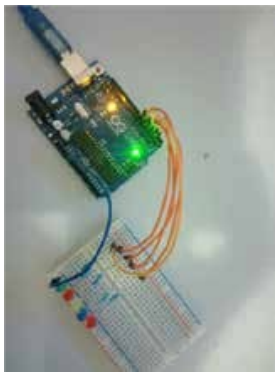
LED 1 – pin 2

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 uslova za ponavljanje *for* petlje.

Zaglavlje *for* petlje ima tri argumenta.

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

Kod iz posljednjeg primjera sa *for* petljom bi izgledao ovako:

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

Prevedeno, za vrijednost broja i = 2 do broja i = 6 sa 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++;

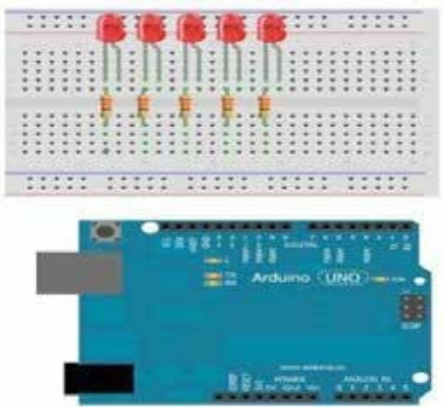
samo kraće napisan matematički izraz

i= i + 1;

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

Potrebne komponente za vježbu:

- Arduino Uno 1 kom
- LE dioda 5 kom
- 330 Ω 5 kom



Slika 31. Šema spoja

Koraci za realizaciju vježbe:

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

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

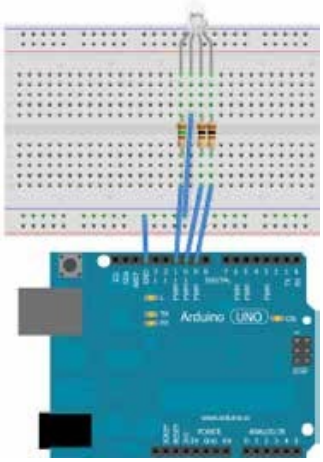
```
void setup()
 {
for(int i=2;i<=6;i++)
 {
      pinMode(i,OUTPUT); //pinovi od 2 do 6 izlazni
 }
 }
 void loop()
 {
 for(int i=2;i<=6;i++)      //efekat 1
 {
   digitalWrite(i, HIGH);
    delay(500);
 }
 for(int i=2;i<=6;i++)      //efekat 2
 {
    digitalWrite(i, LOW);
    delay(500);
 }
for(int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

- Arduino Uno 1 kom
- RGB LED 1 kom
- Otpornik 150 Ω 1 kom
- Otpornik 100 Ω 2 kom



Slika 32. Šema spoja

Koraci za realizaciju vježbe:

1. Kreirajte testni sistem koristeći šemu 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)

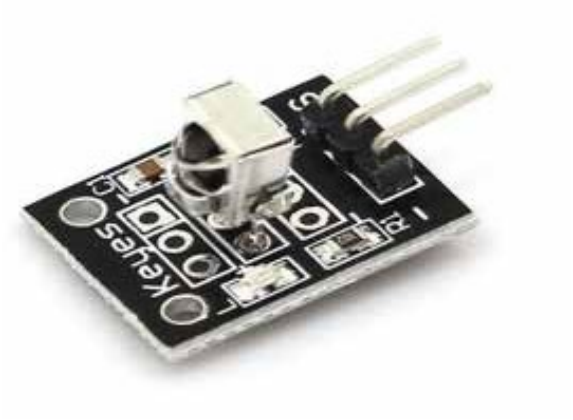
ISHODI UČENJA	NIVO POSTIGNUĆA
- Učenik koristi programsko okruženje za Arduino - Razlikuje elektronske komponente koje su korištene u vježbama - Učenik opisuje ulogu elektronskih komponenti - Učenik definiše svojstva signalnih uređaja (senzora) primijenjenih u vježbama - Učenik definiše način rada sedam-segmentnog displeja - Učenik primjenjuje svoje znanje iz elektronike za povezivanje Arduina i elektronskih komponenti - Učenik koristi električnu šemu spoja za realizaciju vježbe - Primjenjuje znanje iz programiranja za realizaciju određenog zadatka - Koristi odgovarajuće biblioteke unutar programa za realizaciju vježbe - Koristi šemu za povezivanje Arduino uređaja s matador pločicom i ostalim komponentama (senzori, sedam- segmentni displej, taster) - Verifikuje i izvršava program	**Minimalna postignuća** Učenik imenuje sve potrebne komponente korištene u vježbi Uz pomoć nastavnika povezuje dijelove s Arduinom, verifikuje i izvršava program **Dovoljna postignuća** Učenik objašnjava ulogu elektronskih komponenti korištenih u vježbi Povezuje elektronske komponente prema šemi spoja Učenik posjeduje osnovna znanja o pisanju programa za prethodne vježbe Koristi se odgovarajućim bibliotekama unutar programa Samostalno verifikuje i izvršava program **Visoka postignuća** Učenik samostalno povezuje komponente (Arduino, matador ploču, otpornike, LE diode, senzore, sedam-segmentni displej, tastere) korištene u vježbama prema šemi spoja Primjenjuje svoje znanje iz programiranja za pisanje koda Samostalno verifikuje i izvršava program, te po potrebi uspješno otklanja nastali bug

Vježba 6.
INFRACRVENI SENZOR I UPRAVLJAČ

Dešifrovati 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).

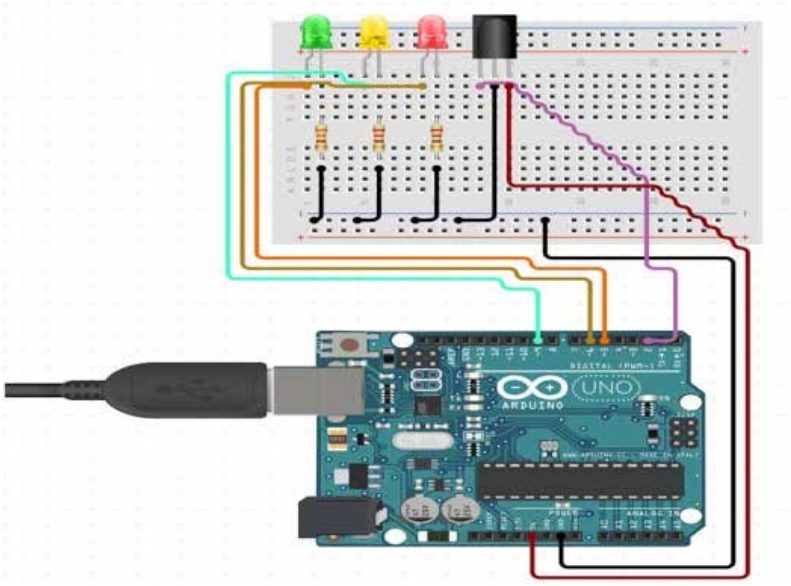
Potrebne komponente:

- Arduino UNO i USB
- Matador pločica
- Crvena, zelena, žuta LED s odgovarajućim otpornicima
- IR receiver
- Daljinski upravljač



Slika 34. IR receiver

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



Slika 35. Šema spoja

Za ovaj zadatak potrebno je koristiti biblioteku *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);`

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

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

Provjera da li je primljen neki signal:
`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();
  }
}
```

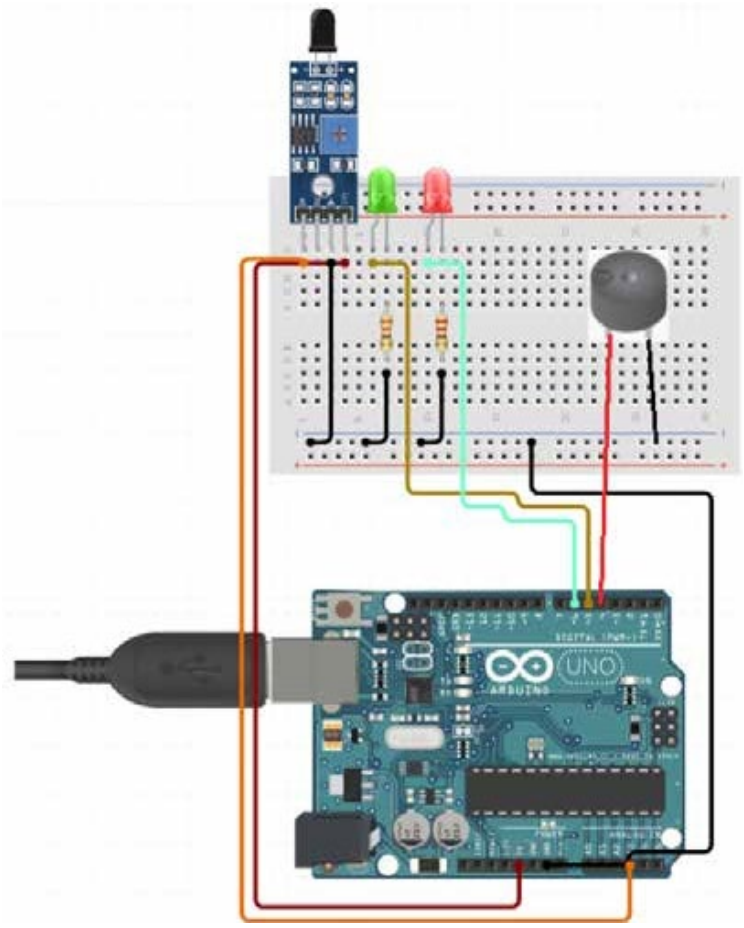
Vježba 7.
SENZOR ZA DETEKCIJU PLAMENA

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

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

Potrebne komponente:

- Arduino UNO pločica i USB
- Matador pločica
- 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
}
```

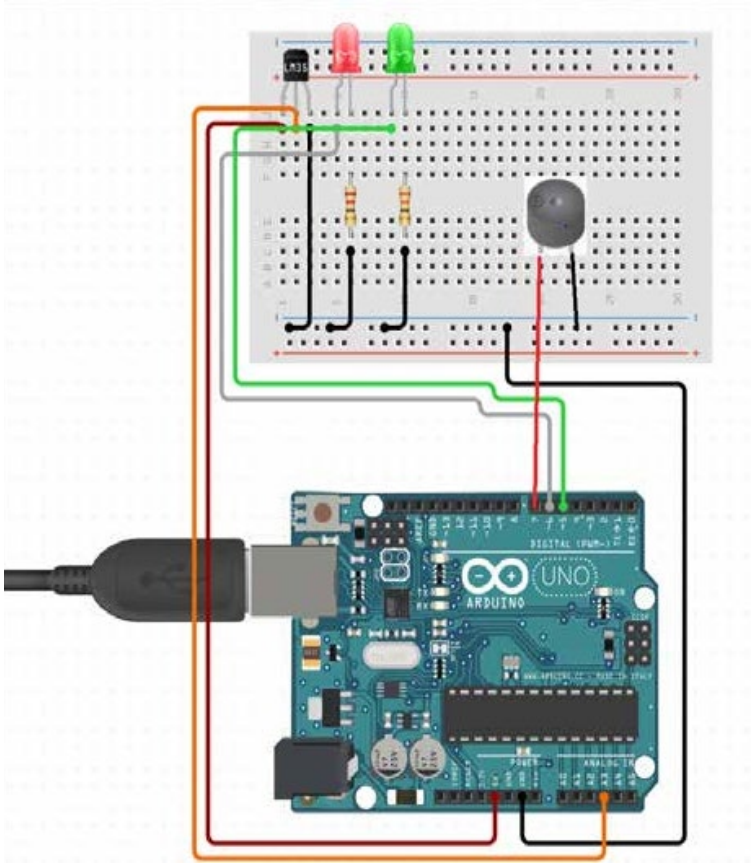
Vježba 8.
MJERENJE TEMPERATURE

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

Potrebne komponente:

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

Spojiti komponente kao na slici:



Slika 37. Šema 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);
  }
}
```

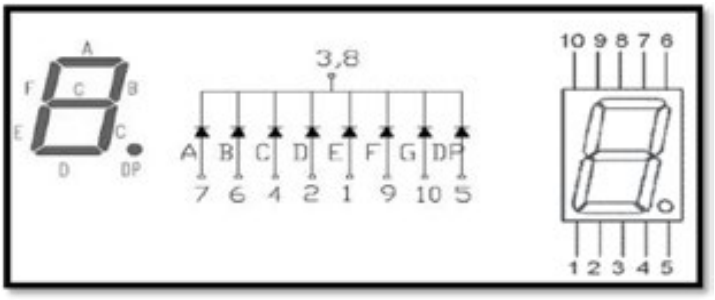
ISHODI UČENJA

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

Sedam-segmentni displej

Sedam-segmentni displej predstavlja izlazni uređaj na kojem 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 je prikazana konfiguracija sa zajedničkom katodom.

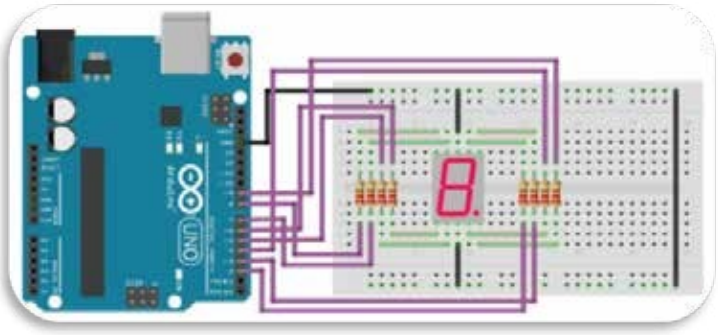
Dakle, anode LE dioda se spajaju na pinove mikrokontrolera preko predotpora, a zajednički pin na *masu-gnd* na Arduino razvojnoj platformi. HIGH signal na pinu mikrokontrolera aktivira jedan segment na displeju. Na taj način kombinacijom aktivnih pinova dobijemo prikaz brojeva na sedam-segmentnom displeju.



Slika 38. Pinout za sedam-segmentni displej

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom
2. sedam-segmentni displej 1 kom
3. Otpornik 330 Ω 7 kom



Slika 39. Šema spoja

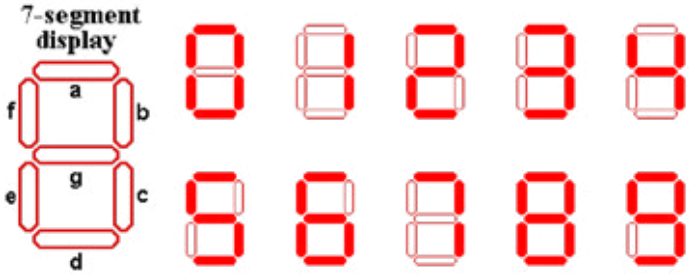
Koraci za realizaciju vježbe:

1. Kreirajte šemu spoja koristeći *Fritzing* skicu.

2. Kreirajte *Sketch* kojim ćete upravljati radom sedam-segmentnog displeja.

Više je načina da se taj problem riješi. Jedan od njih je da ponovno bez upotrebe drugih programskih struktura riješimo problem. Ali prvo trebamo definisati 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 displeju

Prema gornjoj slici, kod za prikaz broja „1" na našem displeju bi bio 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 displeju bi bila:

```
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 bi bio da se za svaki broj kreira custom funkcija za prikaz broja „1” te da se ista pozove u loop petlju. To je moguće uraditi 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 kompletan kod za prikaz broja „1” bi izgledao 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” displej; funkcija za gašenje displeja bi izgledala ovako:

```
void gasi() //kreiranje funkcije za "isključenje" displeja
{
      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 displeju. 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 od načina je upotreba ***if*** testa..

```
if (uslov ispunjen)
        {
   //uradi 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 bi glasio

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 brojac. Globalne varijable se deklarišu izvan tijela funkcije i vidljive su svim funkcijama u programu. Lokalne varijable se deklarišu unutar neke funkcije i samo ta funkcija vidi tu varijablu.

```
int brojac=0;   // globalno deklarisana varijabla svi je "vide"
void setup ()
{
    int i=0;    // lokalno deklarisana 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, ukoliko poslije if testa imamo samo
jedan
                           //programski iskaz, nisu potrebne vitičaste
zagrade

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

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

Dodajte linije koda da stanje na sedam-segmentnom displeju odgovara stanju globalnog 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, ukoliko poslije if testa imamo samo
jedan
                           //programski iskaz, nisu potrebne vitičaste
zagrade

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

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

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

```
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 našem 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 pomoću funkcije

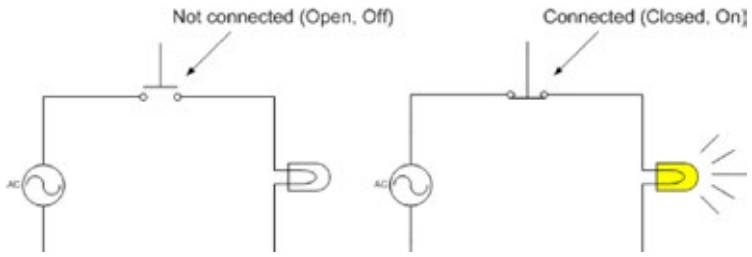
digitalRead(pin)

Funkcija ima samo jedan argument „pin” s kojeg 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 softwaresku stranu problema, idemo sada da vidimo kako ćemo riješiti hardwer za digitalni ulaz.

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

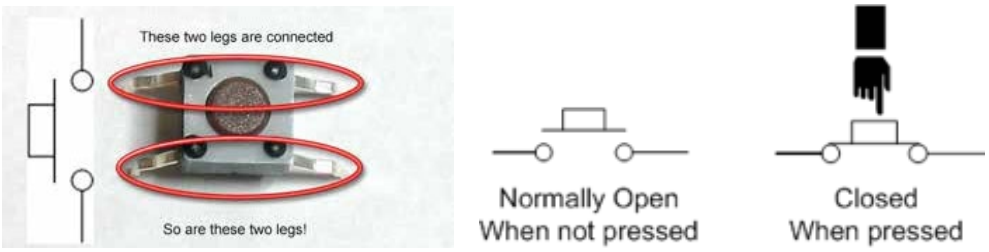


Slika 41. Taster aktivan i taster neaktivan

Naravno, prekidači koje koristimo u našim domovima su super pouzdani, ali jednostavno su gabaritima veliki za primjenu u mikrokontrolerskim sistemima. Mi ćemo za naše eksperimente koristiti mikro taster, 6 mm.

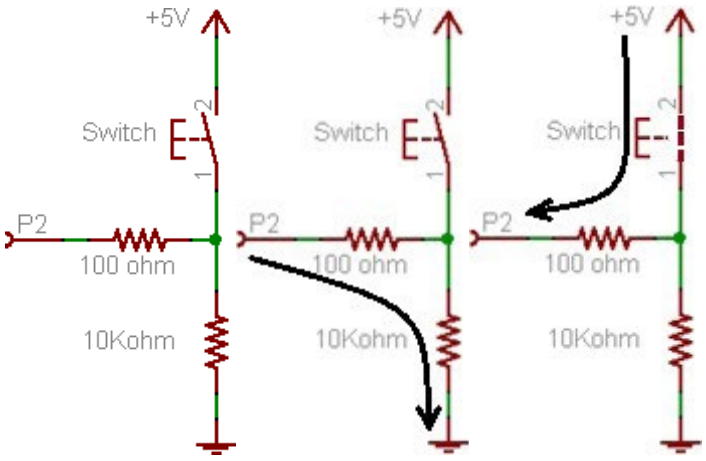
Slika 42. Mikro taster, 6 mm

Ovi mali tasteri su jeftini, praktični za upotrebu 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. Mikro taster, 6 mm, unutrašnja struktura

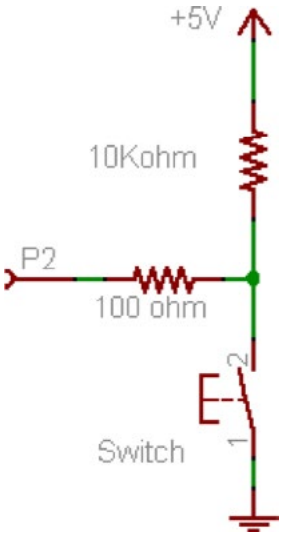
Dobra praksa je nožice ispraviti jer na taj način se lakše pozicioniraju na matador ploči. Ali kako formirati električno kolo na ispravan način? Naime, dvije su kombinacije kako spojiti taster na ulaz mikrokontrolera. To su *pull-down* i *pull-up* konfiguracija. *Pull-down* konfiguracija prikazana je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



Slika 44. Pull-down konfiguracija

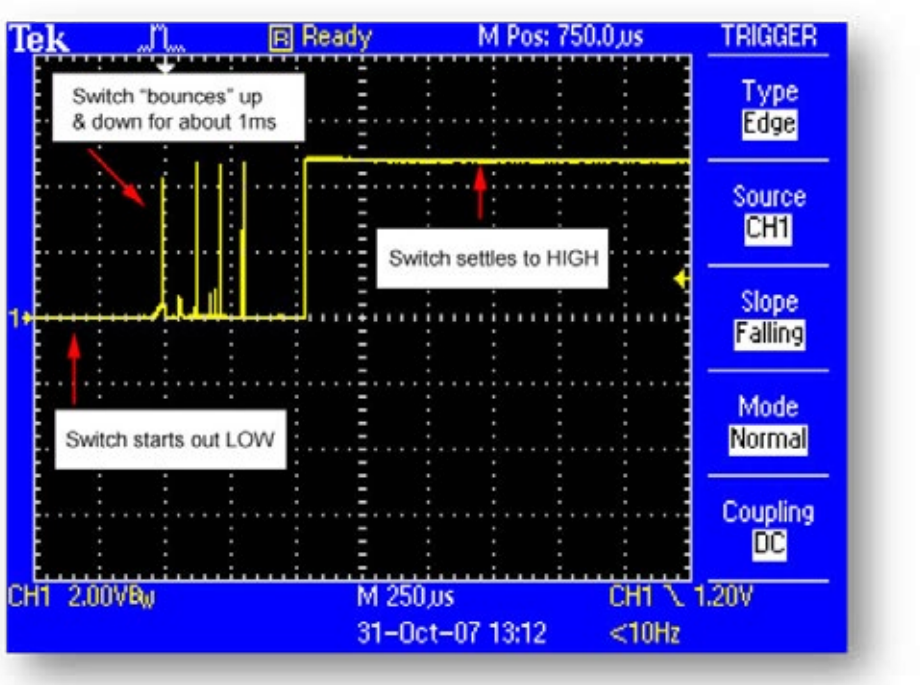
Objasnit ćemo prethodnu sliku. Kada taster nije pritisnut, pin 2 mikrokontrolera je preko 100-omskog i 10-kiloomskog otpora spojen na masu. Stoga kontroler to vidi kao logičku 0 (nulu). S druge strane, kada se taster pritisne, imamo situaciju da na ulaz pina 2 mikrokontrolera preko 100-omskog otpora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

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



Slika 45. Pull-up konfiguracija

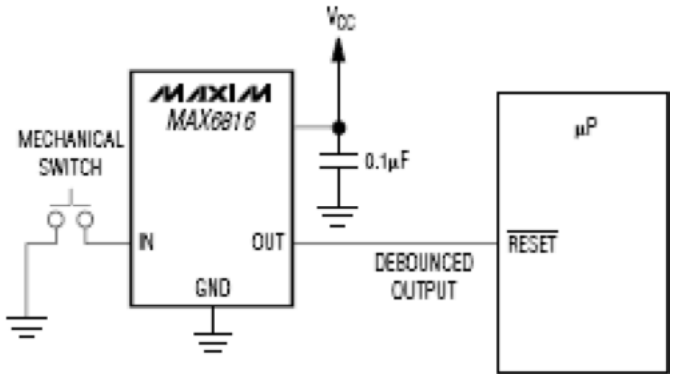
Drugi problem kod digitalnih ulaza s mehaničkim prekidačima je tzv. ***Debouncing*** ili odskakanje kontakata, koje se javlja prilikom pritiska tastera. Efekat odskakanja kontakata se najbolje vidi snimanjem signala osciloskopom.



Slika 46. Debouncing – odskakanje kontakata

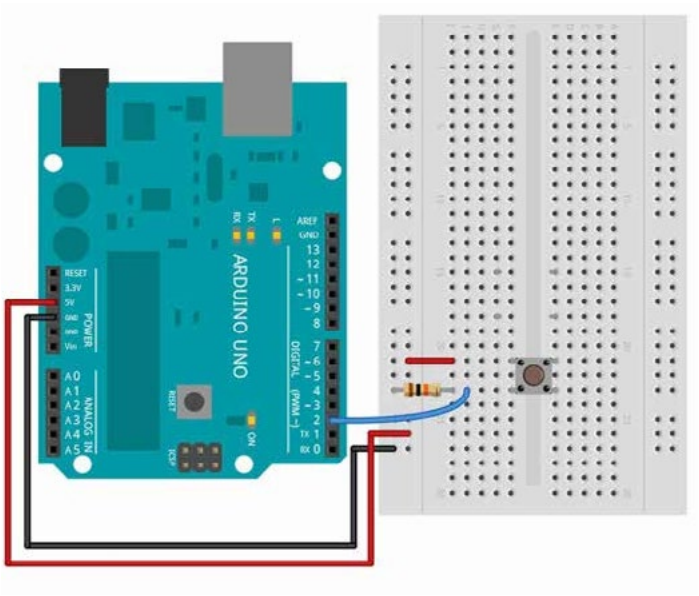
Sada je jasno da će mikrokontroler umjesto da jednom očita stanje logičke jedinice to u slučaju sa slike uraditi 5 puta, tako da ukoliko bi imali aplikaciju koja broji pritiske tastera korisnika, svaki put bi imali pogrešno brojanje.

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



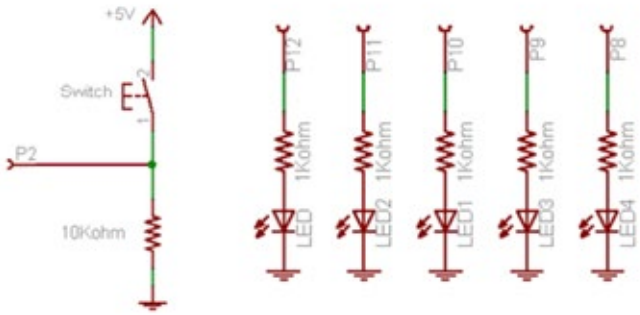
Slika 47. Debouncing integralno kolo

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 šema spajanja tastera na pin 2

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



Slika 49. Šema spajanja za kontrolu LE

```
    /*
    Praćenje stanja na digitalnom ulazu te promjena stanja na LED na
pinovima od 12 do 8. Brojanje koliko puta je 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 displej te prikažite vrijednost varijable counter na sedam-segmentnom displeju.

```
/*
*/
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:

Upotreba *digitalRead*() funkcije.

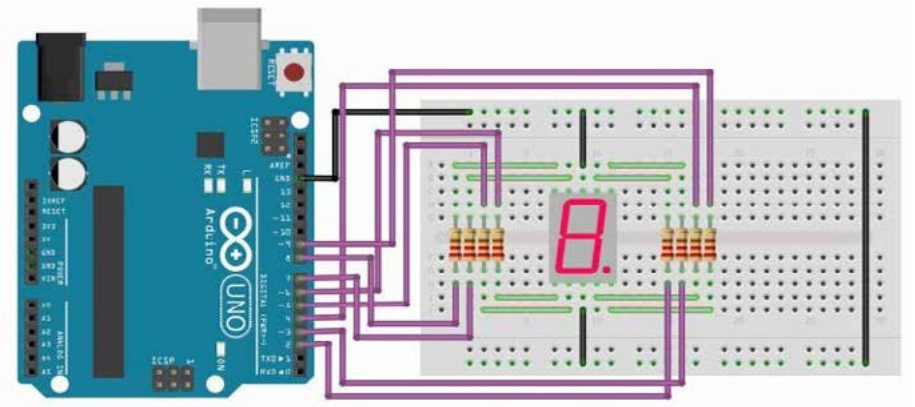
Zadatak vježbe:

Kreirati *Sketch* za kontrolu rada sedam-segmentnog displeja -> ***If uslovi!***

Potrebne komponente za vježbu:

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

Šema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte sklop koristeći *Fritzing* skicu.
2. Kreirajte *Sketch* kojim ćete upravljati radom sedam-segmentnog displeja na taj način da se svakim pritiskom tastera mijenja prikaz na istom 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:

Upotreba *digitalRead()* funkcije.

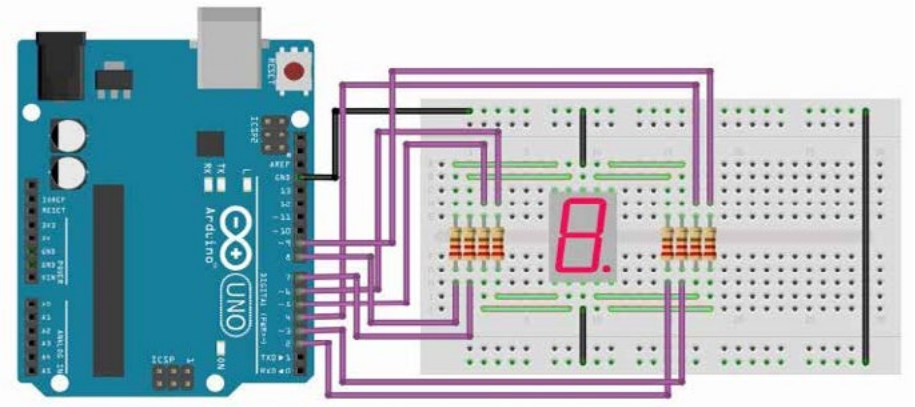
Zadatak vježbe:

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

Potrebne komponente za vježbu:

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

Šema spoja:



Koraci za realizaciju vježbe

1. Kreirajte sklop koristeći *Fritzing* skicu.
2. Kreirajte *Sketch* kojim ćete upravljati radom sedam-segmentnog displeja na taj način da se svakim pritiskom tastera mijenja prikaz na istom 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:

Upotreba *digitalRead*() funkcije..

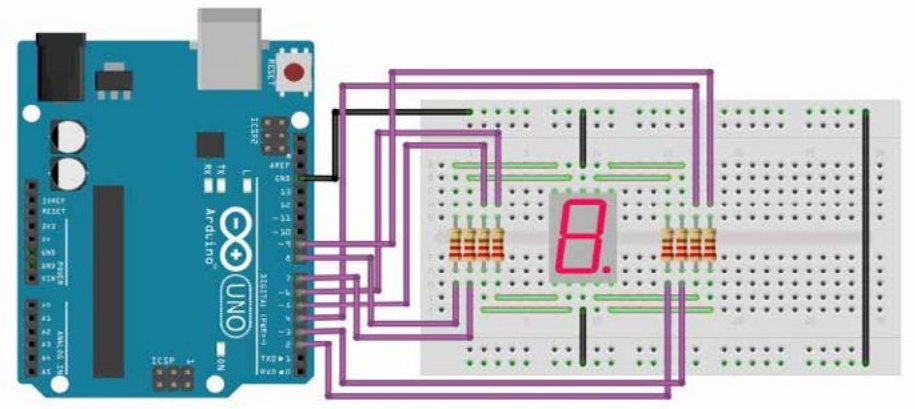
Zadatak vježbe:

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

Potrebne komponente za vježbu:

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

Šema spoja:



Koraci za realizaciju vježbe

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

Napisati kod:

Serial biblioteka – software debugging

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

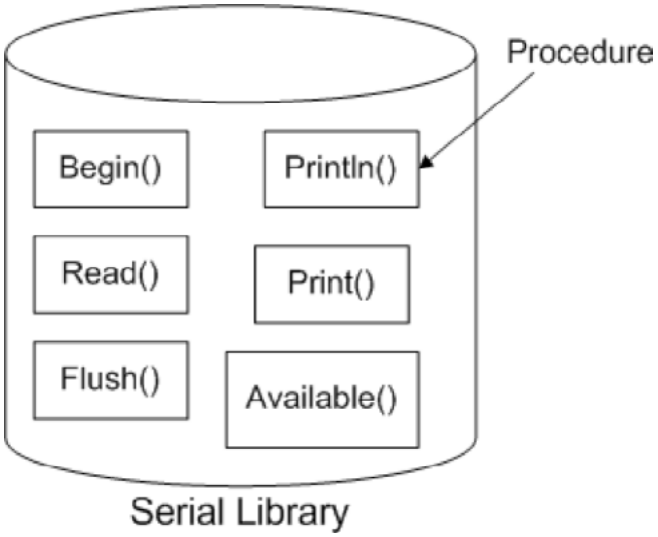
Taj skup procedura se zove biblioteka, u našem konkretnom slučaju riječ je o Serial Library koja omogućava da PC aplikacija komunicira s Arduino kroz USB port.

Prije nego što nastavimo, reći ćemo nekoliko riječi o bibliotekama. Recimo kada budete imali zahtjev da napravite aplikaciju za kontrolu stepper ili servo motora, vjerovatno ćete trebati *servo.h* biblioteku ili *stepper.h* biblioteku. Pozivanje biblioteke se radi na sljedeći način:

#include <servo.h>

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

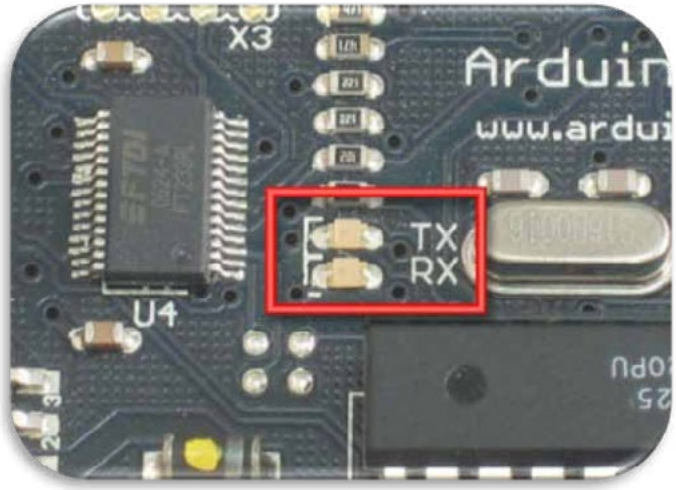
Biblioteka koju prvu trebamo naučiti koristiti je *Serial biblioteka*. Na slici ispod su prikazane osnovne procedure koje ćemo koristiti u narednim primjerima.



Slika 51. Procedure iz biblioteke 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 Arudino 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.

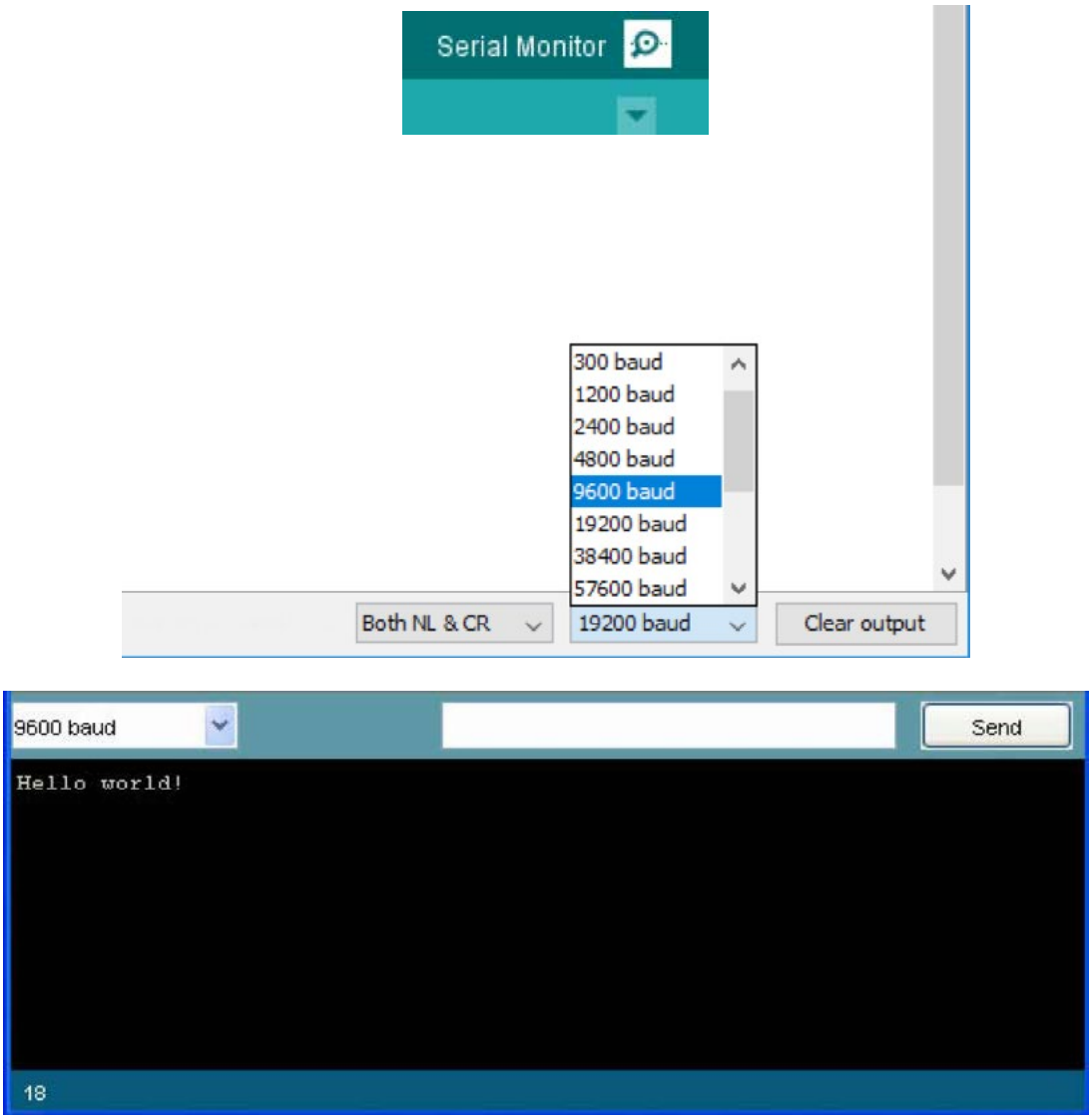


Slika 52. RX i TX LED

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

```
    /*
    „Zdravo, svijete!" aplikacija u kojoj ćemo vidjeti kako Arduino
šalje informacije na PC. Nećemo morati uraditi #include <Serial.h> jer
je ova biblioteka buil-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 resetujte kontroler na tasteru *reset* na razvojnoj platformi.

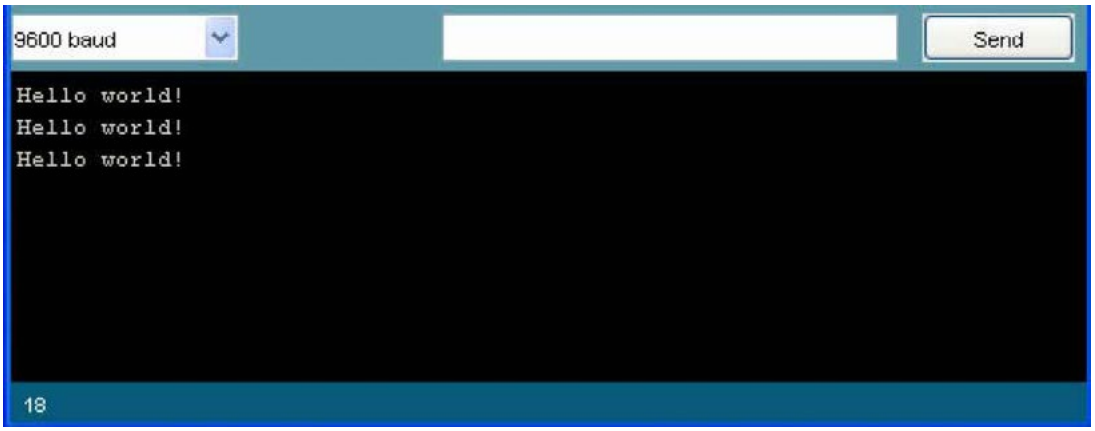


Slika 53. „Hello, World!" Aplikacija za komunikaciju

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

```
     /*
     „Zdravo, svijete!” aplikacija 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
     }
```

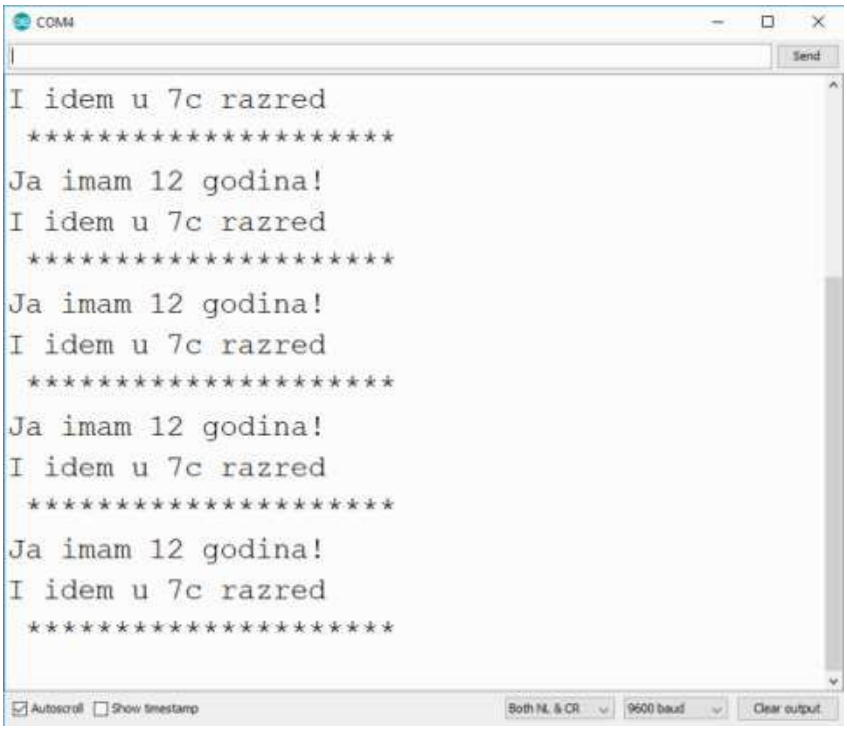
Rezultat ovog koda je prikazan na slici ispod, dakle svake sekunde mikrokontroler na PC pošalje poruku „Hello, World!”.



Slika 54. „Hello, World!”

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

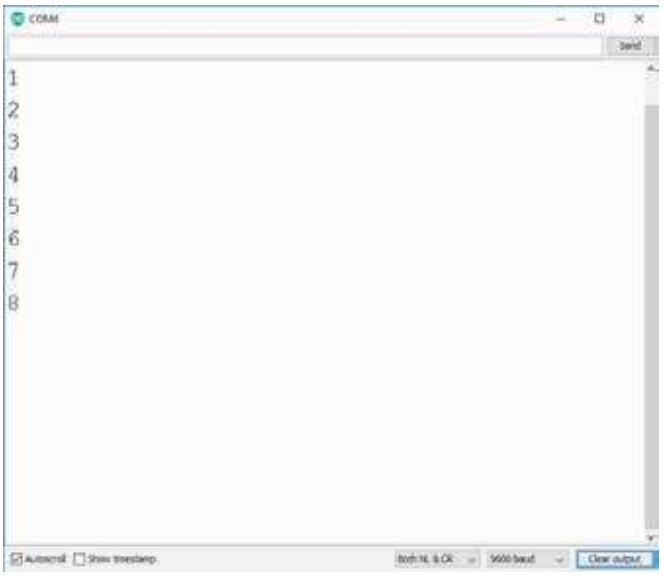
```
     /*
     „Zdravo, svijete!” aplikacija u kojoj ćemo vidjeti kako Arduino
šalje informacije na PC
     */

     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 55. Možete napisati i pismenu zadaću u Arduinu

Dakle, iz gornjeg primjera se vidi da je moguće kreirati i malo kompleksnije informacije koje se mogu slati na PC. Prije nego 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.

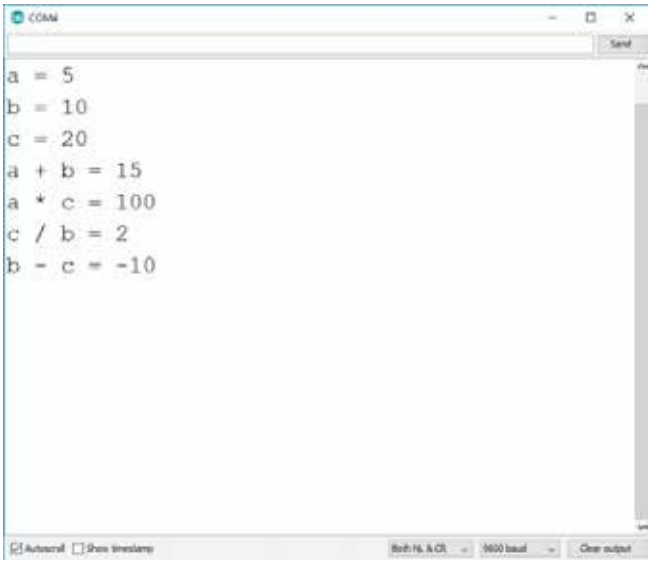
```
      /*
      „Zdravo, svijete!” aplikacija u kojoj ćemo vidjeti kako Arduino
šalje informacije na PC
      */
      int i=0;

      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 56. Prikaz stanja bojača u Serial Monitoru

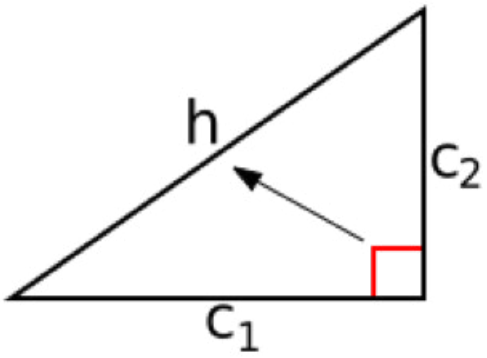
Idemo probati raditi matematiku koristeći Arduino.

```
/*
 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 57. Rješenje u Serial Monitoru

DIY – serial



Slika 58. Pitagorina teorema

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

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

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

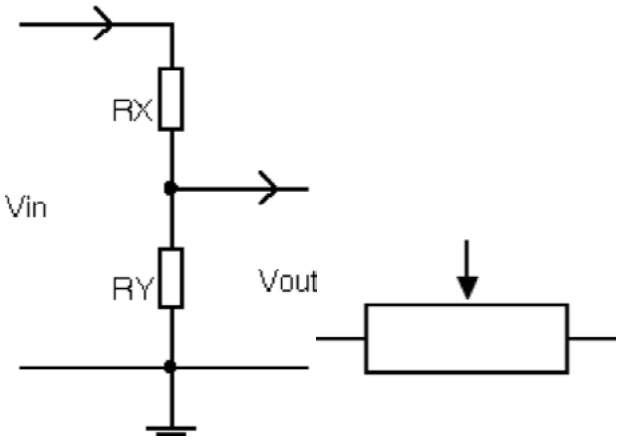
Prepisati rješenje:

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



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

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

  Serial.println("Proracun 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 Arduino platformi imamo 6 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*** će biti duplo manji od Vin ako je ***Rx=Ry.***



Slika 59. Djelitelj napona i simbol potenciometra

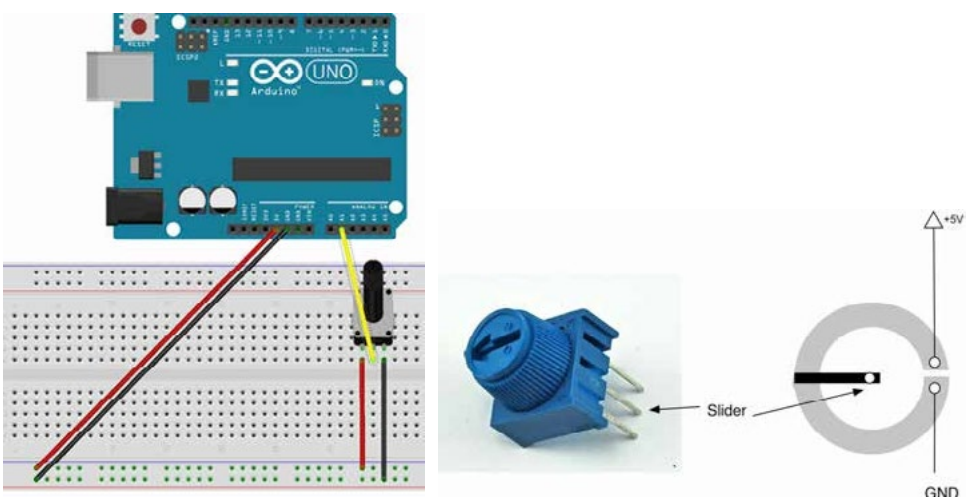
Tako ako je napon na njegovim krajevima 5 V, na kliznom kontaktu možemo imati vrijednosti od 0 do 5 V, te ovaj element kao takav predstavlja idealan elektronski element za testiranje analognog ulaza mikrokontrolera.

Naravno, mikrokontroler ne vidi napon na svom ulazu, već preko svog internog ADC (*analog to digital* konvertera) 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 *Serial* biblioteku za softwareski *debugging*. Spojite šemu na matadoru prema slici ispod.



Slika 60. Potenciometar na matador ploči 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 monitor svakih 250 milisekundi u sljedećem formatu:

Raw data: 512 Voltage: 2.5 (V)

DiY – analogRead

Cilj vježbe:

Upotreba *analogRead(pin)* funkcije.

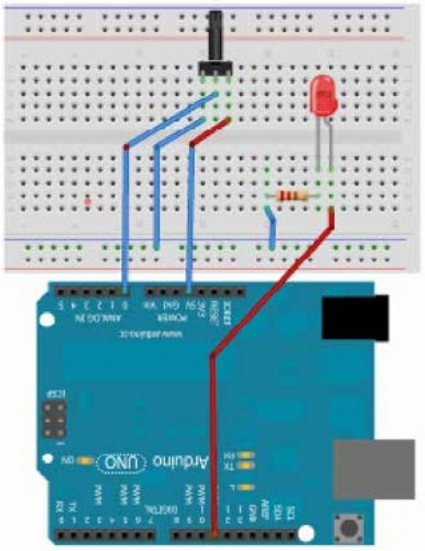
Zadatak vježbe:

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 61. Šema spoja

Koraci za realizaciju vježbe:

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

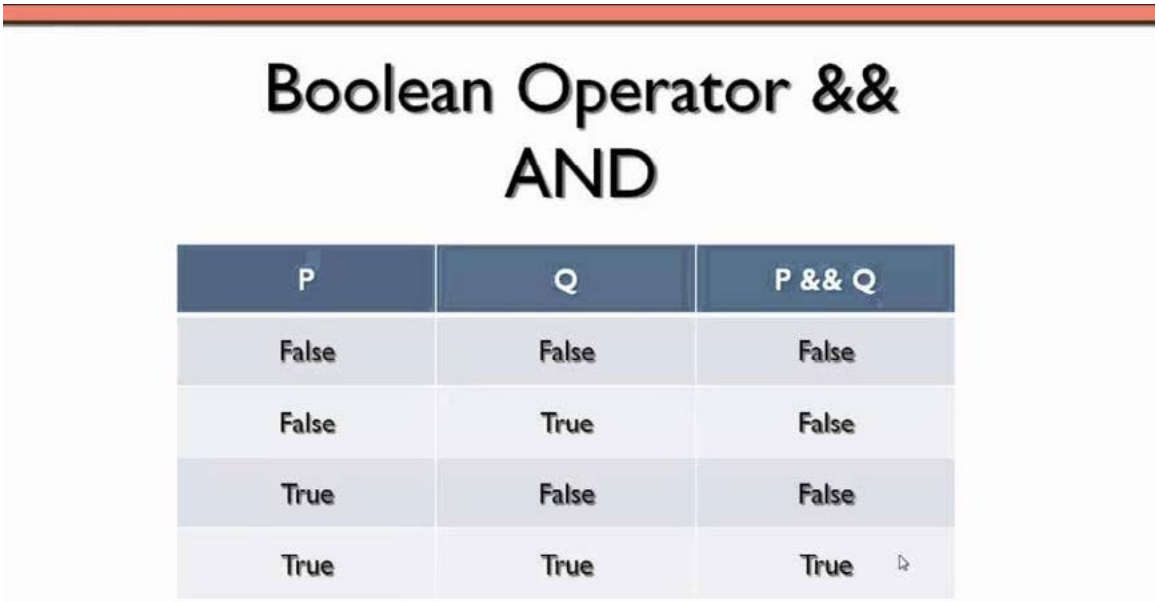
2. Kreirajte *Sketch* kojim ćete čitati analognu vrijednost s analognog pina 0, te vrijednost prikazati u *Serial Monitoru* i spram zadatka vježbe upaliti odgovarajuće LE diode.

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)

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

&& (logical and)

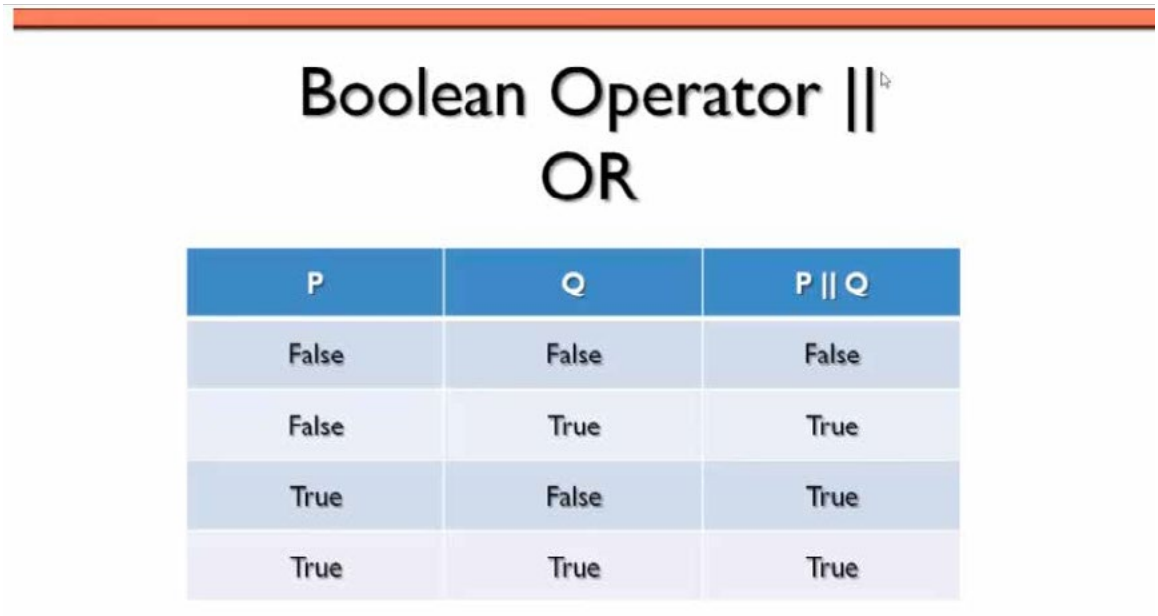


Logički AND ili logički „i" operator zahtijeva da oba iskaza budu istinita da bi rezultat upita bio istinit. Npr.:

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

Dakle, uslov je ispunjen samo ako je vrijednost varijable val1 = 100 i varijable val2 = 200. Za sve ostale uslove nećemo izvršiti ono što je unutar *if* uslova.

|| (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, uslov je ispunjen ako je vrijednost varijable val1 = 100 ili varijable val2 = 200 ili ako su obje varijable tačne.

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:

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

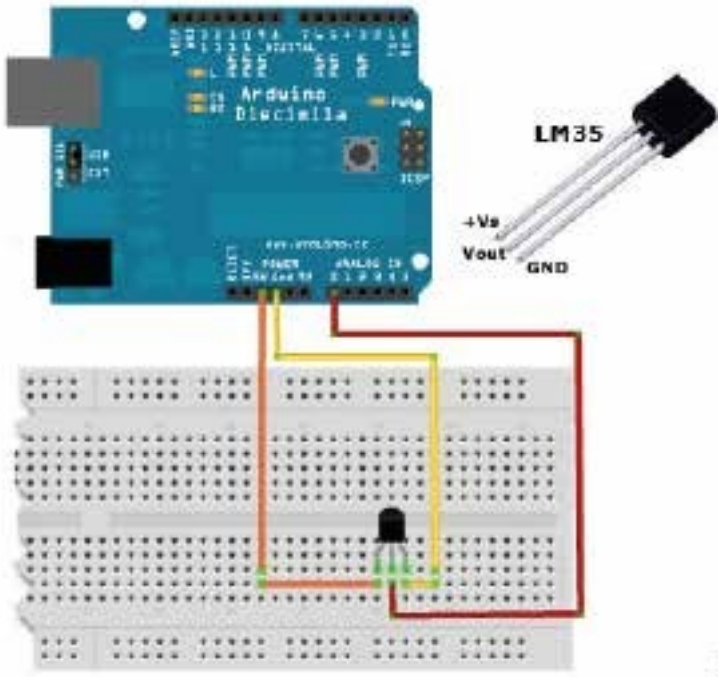
Zadatak vježbe:

Kreirati *Sketch* za proračun temperature.

Potrebne komponente za vježbu:

1. Arduino Uno 1 kom
2. LM35 1 kom

Šema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sistem koristeći skicu iznad.

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

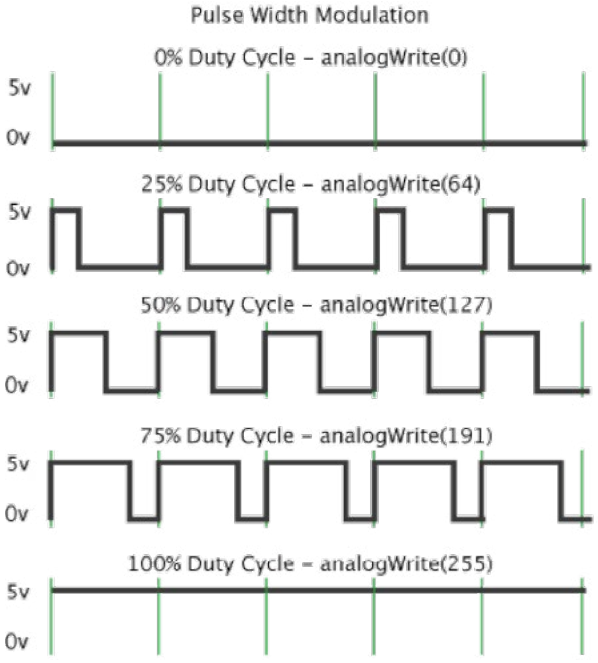
PWM – analogWrite

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

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

Vrijeme „On time" signala se zove 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 grafiku ispod je prikazan PWM signal. Period (T) je označen zelenom bojom i traje 2 milisekunde, pa je frekvencija f = 1/T, 500 Hz.



Slika 62. PWM signal

Za nas je dovoljno da znamo da se PWM signal može generisati 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 generišemo PWM signal i *value*, koja može biti u intervalu od 0 do 255. Naš PWM signal ima 8-bitnu rezoluciju pa je $2^8 = 256$.

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:

Upotreba *analogWrite()* funkcije.

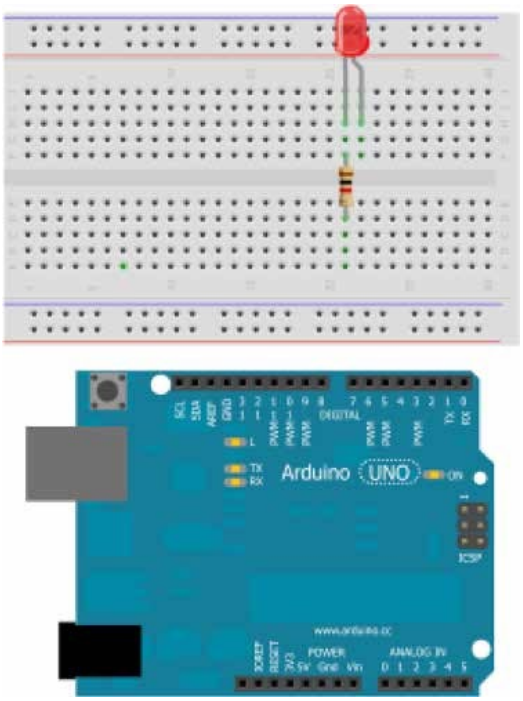
Zadatak vježbe:

Kreirati *Sketch* za upravljane radom LE diode sa *fade in* i *fade out* efektom.

Potrebne komponente za vježbu:

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

Šema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte šemu spoja koristeći *Fritzing* skicu.

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 vice versa do (0), sve s inkrementom i dekrementom od 1.

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

```
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:

Upotreba *analogWrite()* funkcije.

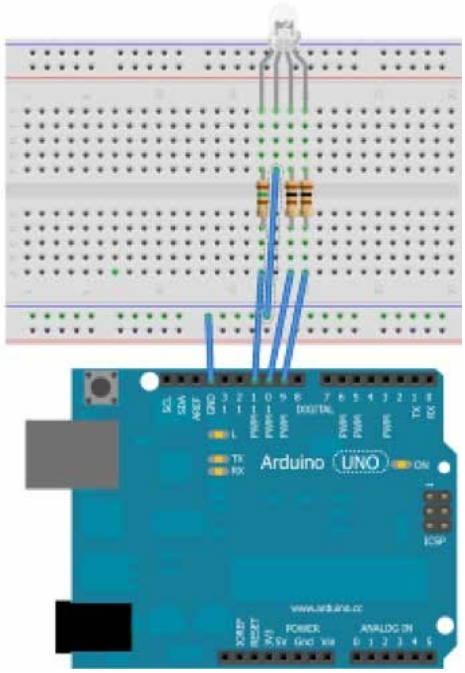
Zadatak vježbe:

Kreirati *Sketch* za upravljane radom RGB LED sa *fade in* i *fade out* efektom..

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

Šema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sistem koristeći *Fritzing* šemu spoja.

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 *vice versa* 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 sisteme, a možda čak i razumjeti kako oni funkcionišu.

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

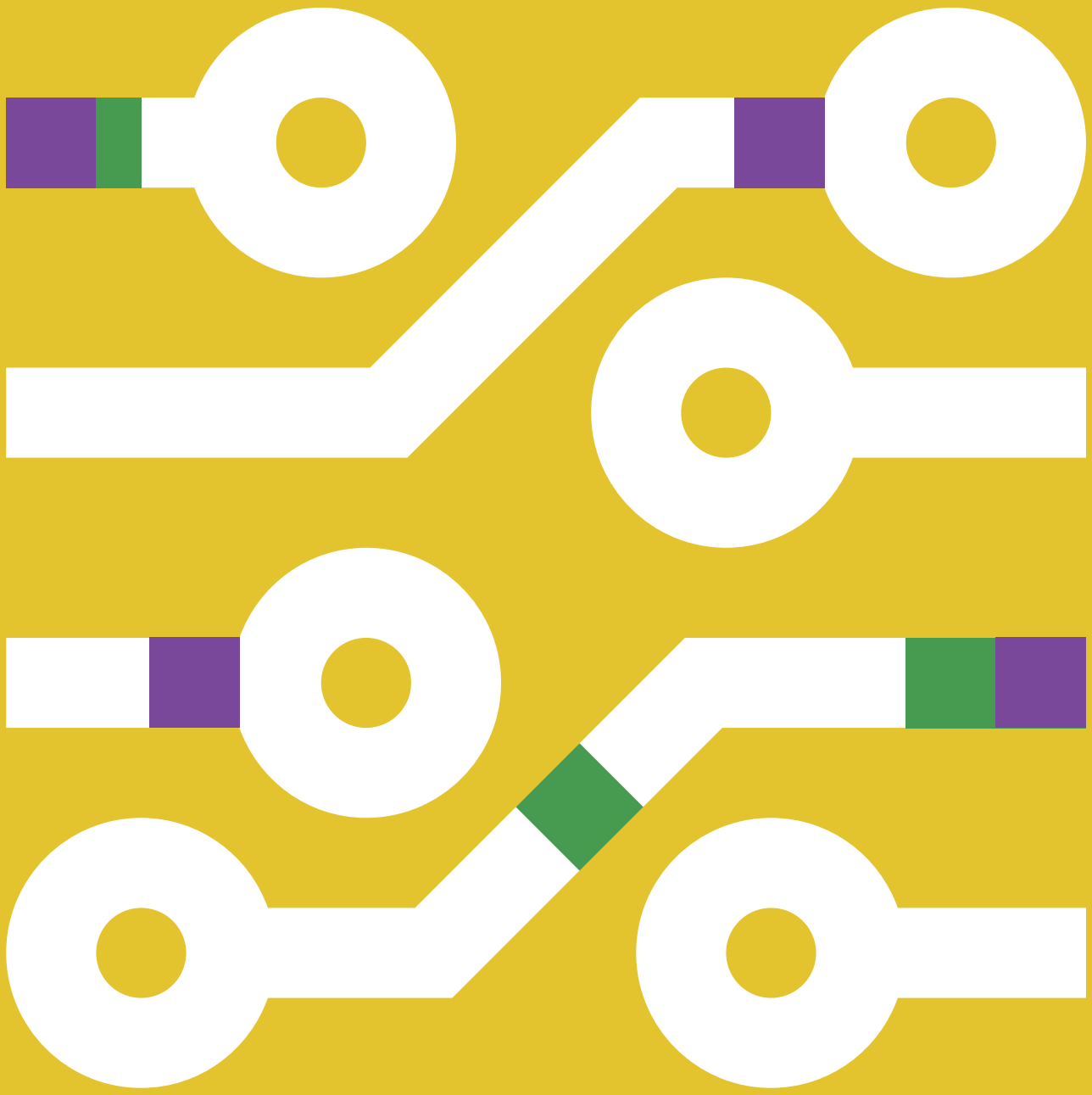
Pokušajte uvidjeti s kojim problemom se srećete svakodnevno. Razmišljajte da li je moguće za rješenje Vaših problema osmisliti neki mikrokontrolerski sistem.

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

Mr. sci. 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: .....

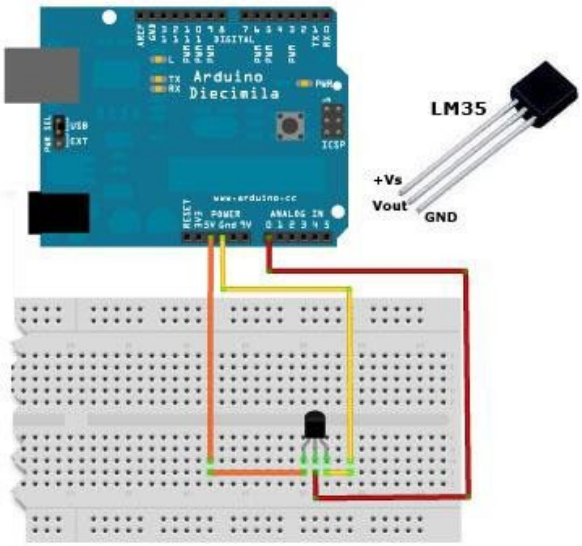
Cilj vježbe:
Upotreba analogRead(), analogWrite() , serialBegin() i serialPrintln() funkcije.

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

Potrebni elementi za vježbu:

1. Arduino Uno 1kom
2. LM35 1kom
3. Matador ploča 1kom

Šema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sistem koristeći Fritzing skicu.

2. Kreirajte Sketch kojim ćete čitati analognu vrijednost sa analognog pina 0 te vrijednost prikazati u Serial Monitoru. Koristeći jednačinu za proračun temperature izračunati temperature u ^{o}C, K i F te iste prikazati u Serial Monitoru.

Osnovne torijske postavke:

Temperatura temepraturnog senzora LM35 se može izračunati prema sledećem izrazu:

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

analogRead()

Opis
Čita vrijednost sa definsanog analognog pina (10-bitni A/D konverter). Ulazni napon od 0 do 5 V se mapira 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 vjezbe je upotreba analogRead() funkcije
 * te upotreba tvz. arduino softverskog debugera
 * za ocitanje i prikaz tmeperature prostorije
 */

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

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

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

 T=(5*val*100)/1023;  //formula za proracun temeprature za LM35
senzor

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

Zaključak:

LABORATORIJSKA VJEŽBA BR.2

ARDUINO UNO – 2x16 LCD display

UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....

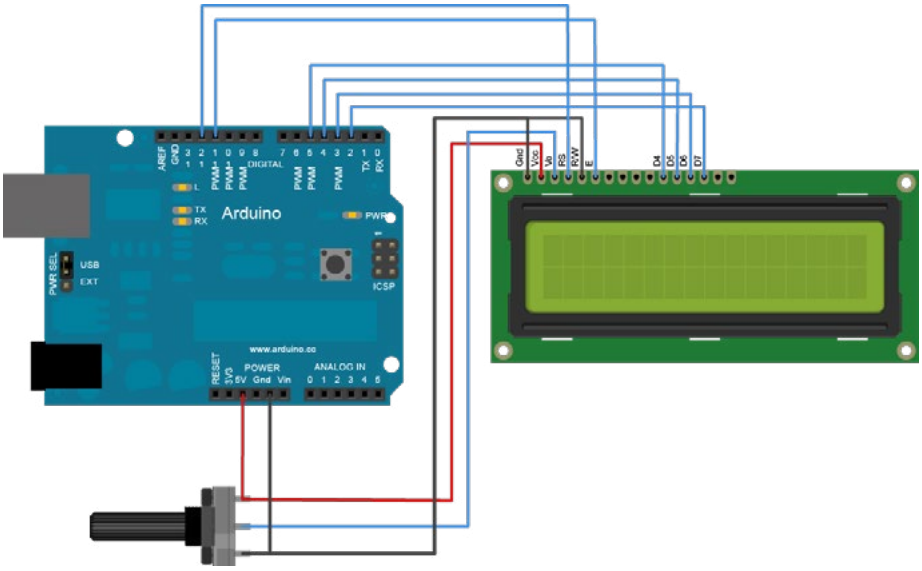
Cilj vježbe:
Upotreba 2x16 LCD displeja, način spajanja.

Zadatak vježbe:
Kreirati Sketch za prikaz teksta .

Potrebni elementi za vježbu:

1.	Arduino Uno	1kom
2.	2x16 LCD display	1kom
3.	Potenciometar 10KΩ	1kom
4.	Matador ploča	1kom

Šema spoja:



1. Kreirajte testni sistem koristeći *Fritzing* skicu.

2. Kreirajte *Sketch* u kojem cete pozvati biblioteku LiquidCrystal.h naredbom include.h i koristeci HELP upoznati se sa funkcijama potrebnim za ispis teksta na lcd display.

3. U prvom redu napisati svoje ime a u ime škole ili razred i odjeljenje.

Rješenje:

```
/*
 * Primjer koristenja 2x16 lcd displeja
 * za prikaz informacija korisniku
 */

//poziv biblioteke za LCD

#include <LiquidCrystal.h>

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

void setup()
{
  // nas LCD ima 16 kolona 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 display i LM35

UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....

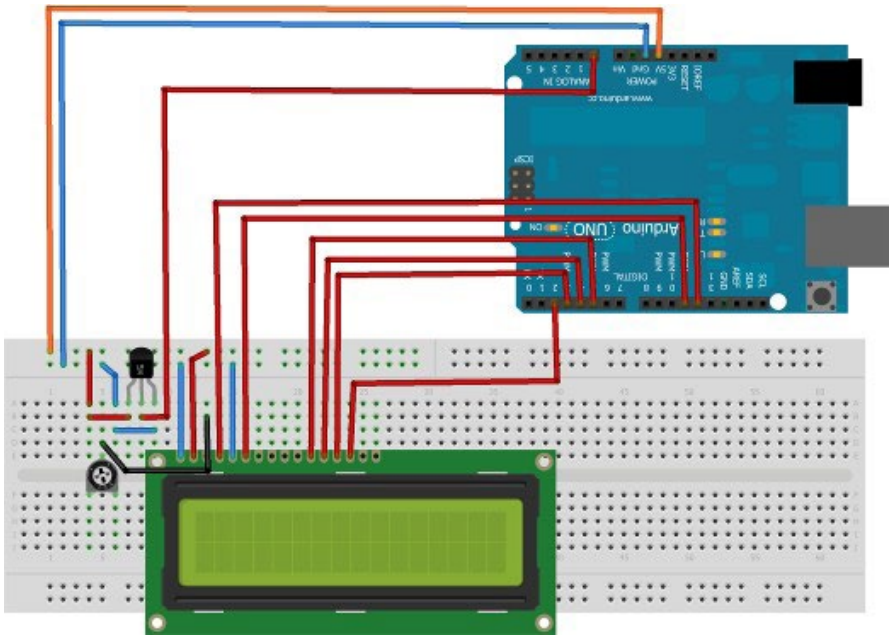
Cilj vježbe:
Upotreba 2x16 LCD displeja, način spajanja.

Zadatak vježbe:
Kreirati Sketch za prikaz temperature.

Potrebni elementi za vježbu:

1.	Arduino Uno	1kom
2.	2x16 LCD display	1kom
3.	Potenciometar 10KΩ	1kom
4.	LM35	1kom
5.	Matador ploča	1kom

Šema spoja:



Koraci za realizaciju vježbe:

1. Kreirajte testni sistem koristeći *Fritzing* skicu.
2. Kreirajte *Sketch* u kojem ćete pozvati biblioteku LiquidCrystal.h naredbom include.h
3. Kreirati app za prikazati temperature u ^{0}C u prvom redu, a u drugom vrijednost napona na analognom izlazu A0. Kreirati tabelu vrijednosti napona i odgovarajuće temperature.

IT Girls - Arduino za sve

Rješenje:

```
                                        /*
 * Primjer koristenja 2x16 lcd displeja
 * za prikaz informacija korisniku
 */

//poziv biblioteke za LCD

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

void setup()
{
  // nas LCD ima 16 kolona 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 se aktivira kada detektira jačinju zvuka veću od one podešene pomoću potenciometra na njemu.

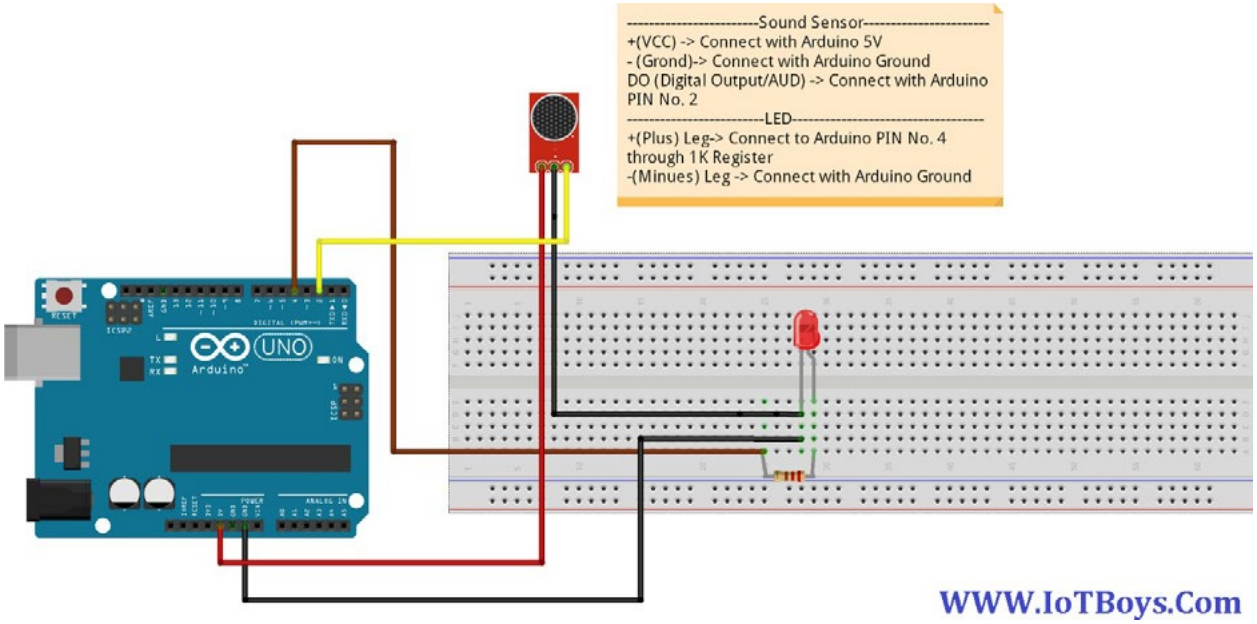
Zadatak:

Napraviti sklop za uključivanje i isključivanje svjetleće 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:

- Arduino UNO pločica i USB
- Matador pločica
- Senzor zvuka
- 1x crvena LED
- 1x otpornik 220 Ω

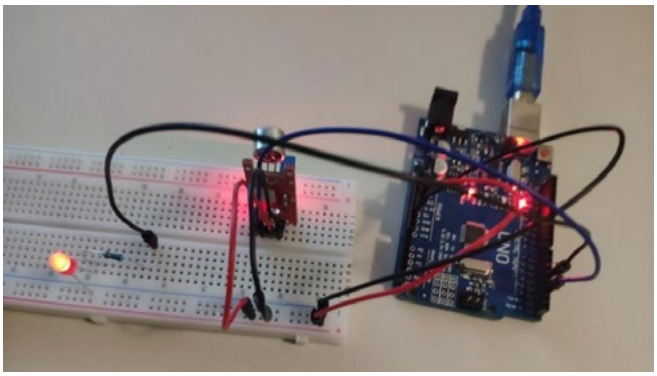
Prikaz spajanja:



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)                                //ukoliko 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

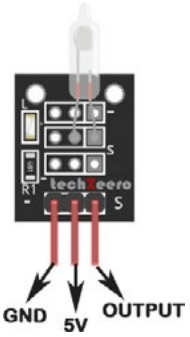
UČENIK/CA: .....

RAZRED: .....

DATUM: .....

VJEŽBU PREGLEDAO/LA: .....

DATUM: .....



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

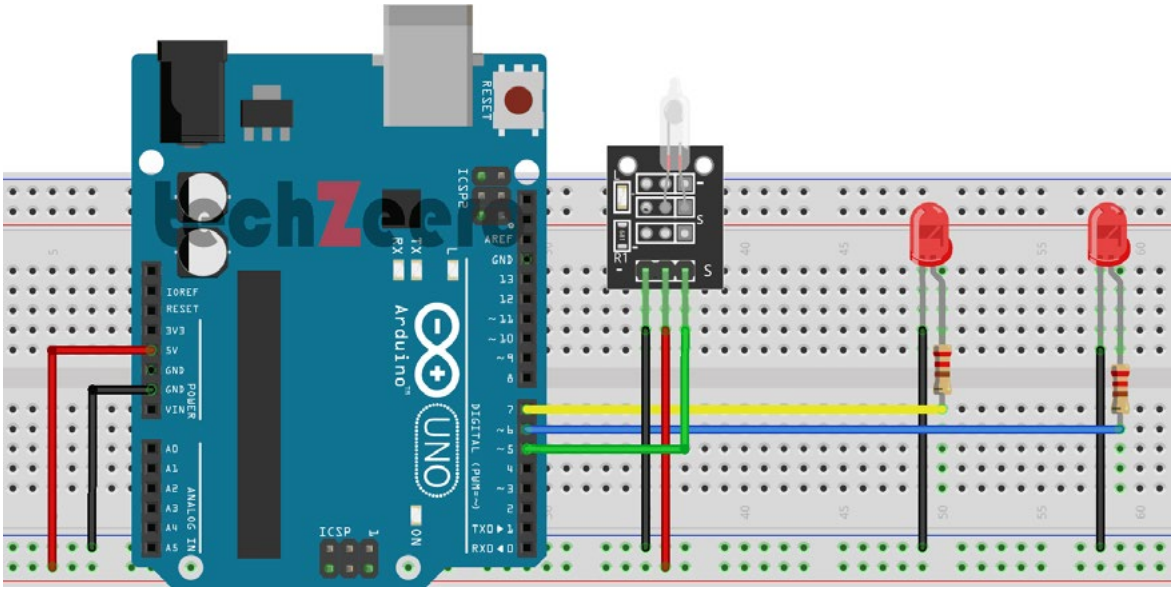
Zadatak:

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 senzorau-spravno uključuje se jedno svjetlo, a kada je horizontalno uključuje drugo svjetlo

Potrebne komponente:

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

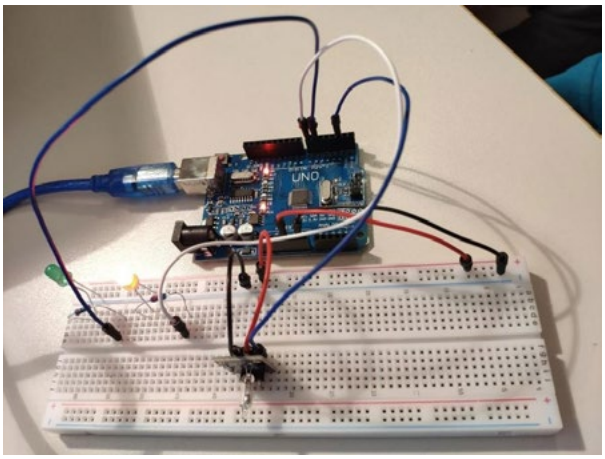
Prikaz spajanja:

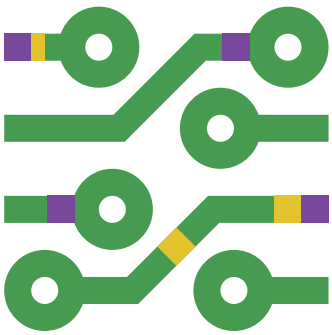


Rješenje:

```
  int tiltSensorPin = 2;    //Pin za sensor 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.