Arduino radionica - osnove

Krešimir Topolovec i Nikola Jerković

13. studenoga 2022.

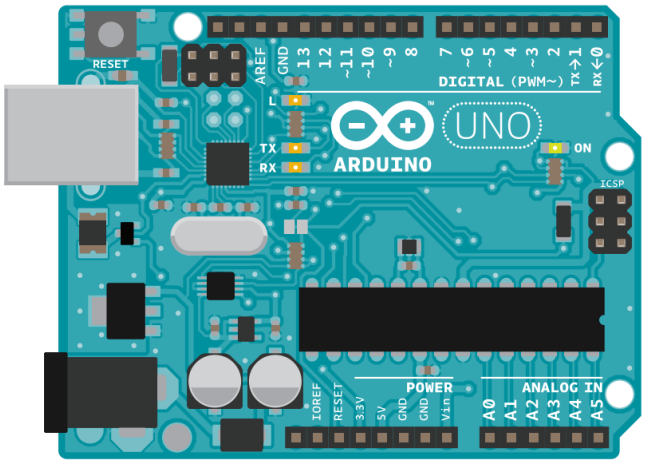
1 Uvod

Kroz ovu vježbu, upoznat ćemo se s Arduino Uno pločicom, Arduino razvojnim okruženjem te osnovnim pojmovima potrebnim za ostvarivanje logike u programiranju i njezinoga manifestiranja u stvarnom svijetu.

1.1 Što ćemo koristiti

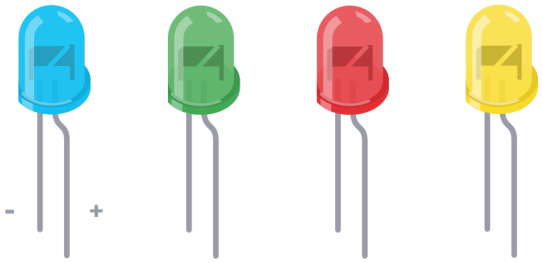
U nastavku ćemo upoznati osnovne elektroničke komponente koje ćemo koristiti kroz primjere odnosno zadatke koji slijede.

- **Arduino UNO pločica**
  Glavna komponenta oko koje ćemo graditi sve primjere i zadatke koji slijede. Srce ove razvojne pločice je mikrokontroler ATmega328 koji je zapravo malo 8-bitno računalo. Na ovoj pločici možemo lako prepoznati označene digitalne ulaze/izlaze te analogne ulaze. Osim toga, na pločici se nalazi i izvodi sa naponima 5V te 5V.



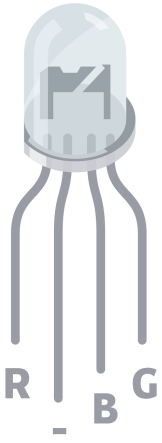
Slika 1: Arduino UNO razvojna pločica

- **Svjetleće diode**
  Vrsta poluvodičke komponente koja emitira svijetlost prolaskom struje kroz nju. Pri korištenju ovih komponenti bitan je polaritet što znači da je za ispravan rad bitno kako ćemo ih smjestiti u elektroničkom krugu.

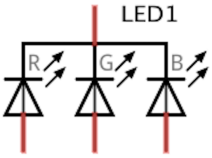


(a) Svjetleće diode

(b) LED simbol

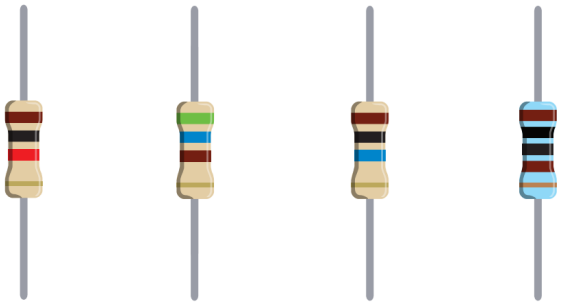


(a) RGB svjetleća dioda

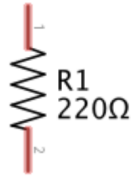


(b) RGB dioda simbol

- **Otpornici**
  Pasivne elektroničke komponente koje pružaju otpor prolasku struje čija se vrijednost se iskazuje u mjernoj jedinici Ω. Koristiti ćemo ih u kombinaciji sa svjetlećim diodama.



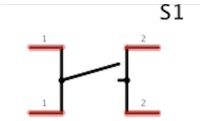
(a) Otpornici



(b) Elektronički simbol otpornika

- **Tipkala**
  Elektromehaničke komponente koje zatvaraju strujni krug pritiskom na gumb. Vrlo su pogodni za korištenje na eksperimentalnoj pločici.

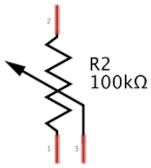
(a) Tipkalo



(b) Simbol tipkala

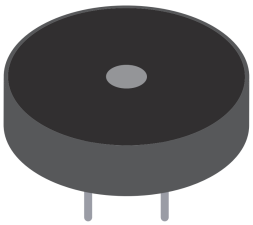
- **Potenciometar**
  Elektromehanička komponenta koja mijenja otpor okretanjem dugmeta. Ima tri nožice. Otpor između krajnje dvije nožice je fiksan dok se okretanjem dugmeta pomiče klizač te se time otpor na srednjoj nožici mijenja. Ovom komponentom možemo kontrolirati pad napona na središnjoj nožici ili pak kontrolirati tok struje kroz neki dio kruga.

(a) Potenciometar

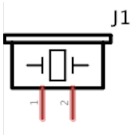


(b) Simbol potenciometra

- **Piezo zvučnik**
  Komponenta koja proizvodi zvuk prolaskom promjenjive struje. Također se može koristiti za detektiranje vibracija.

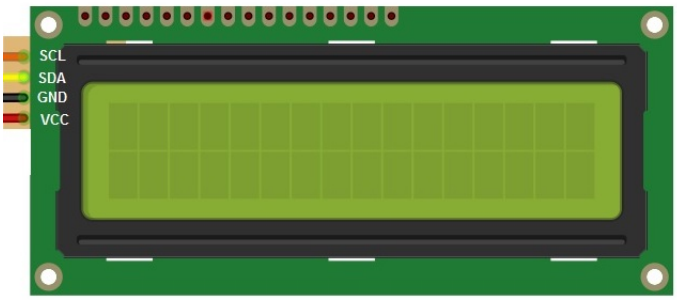


(a) Piezo zvučnik



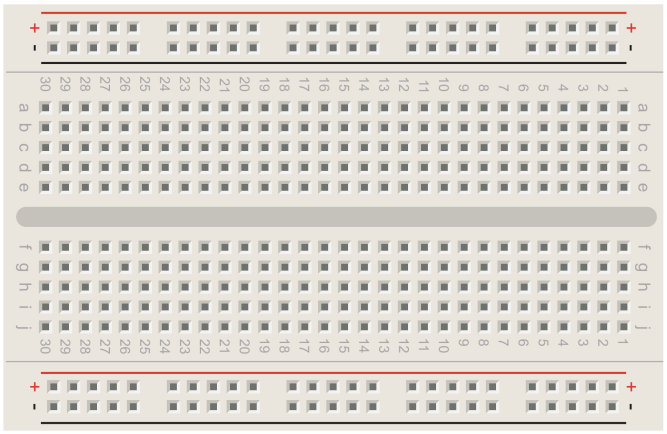
(b) Simbol piezo zvučnika

- **LCD ekran**
  Koristiti ćemo ekran koji može prikazati 2 retka sa po 16 alfanumeričkih znakova



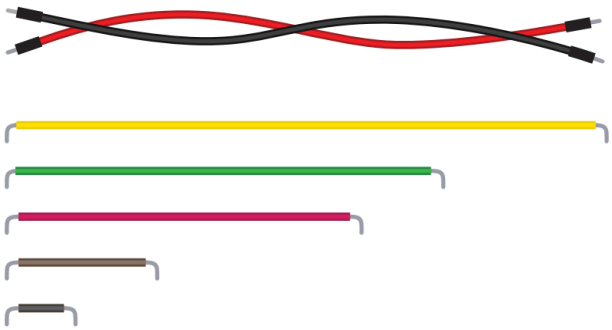
Slika 8: 1602 LCD ekran

- **Eksperimentalna pločica (engl. breadboard)**
  Pločica na kojoj možemo graditi eksperimentalne projekte spajanjem komponenti pomoću žica bez potrebe za lemljenjem.



Slika 9: Eksperimentalna pločica

- **Žice - kratkospojnici**
  Korištene za spajanje komponenti na eksperimentalnoj pločici te na razvojnu pločicu.



Slika 10: Spojne žice

1.2 Arduino IDE

Arduino razvojno okruženje sastoji se od tekstualnog editora, prevoditelja C koda (engl. compiler), popratnim Arduino sadržajima, sučelja za komunikaciju s Arduino mikrokontrolerom te standardnih alata za rad s datotekama.
Kako bi isprobali neke osnovne funkcionalnosti, otvoriti ćemo ugrađeni primjer koji dolazi uz editor.

Potrebno je pokrenuti Arduino IDE te kliknuti na **File -> Examples -> 01. Basics -> Blink**. To otvara prozor s već napisanim programom koji u intervalima pali i gasi svjetleću diodu ugrađenu na Arduino pločici označenu oznakom L.
Da bi se program izvršio, potrebno ga je prvo prevesti u strojni jezik te prebaciti u memoriju mikrokontrolera. Za to je prvo potrebno odrediti tip kontrolera i priključak na kojem se nalazi. To je moguće uraditi na **Tools -> Board: -> Arduino Uno** za odabir kontrolera te **Tools -> Port -> COM3** za odabir priključka računala na koji je Arduino spojen (broj porta može varirati).
Sada kod možemo verificirati pritiskom na ikonu kvačice. Nakon što verifikacija završi, kod možemo prenijeti na Arduino pritiskom na ikonu desne strelice te bi ugrađena LED-ica trebala treperiti.

2 Primjer: Digitalni ulazi i izlazi

U primjeru **Blink** prikazano je pet funkcija bitnih za korištenje Arduina.

setup() je osnovna funkcija u Arduino okruženju koja se izvršava **jednom pri pokretanju** programa. U tijelo *setup* funkcije stavljamo sve naredbe koje želimo izvršiti samo jednom, poput, konfiguracijskih naredbi (*pinMode()*) ili inicijalizacija globalnih varijabli za, primjerice, brojanje ili praćenje nekog stanja.

loop() je druga osnovna funkcija Arduino okruženja, ali se naredbe koje se nalaze u njenom tijelu kontinuirano ponavljaju dok god je Arduino upaljen. Većina naredbi će se nalaziti upravo u ovoj funkciji.
Bitno je napomenuti da u C jeziku postoje programski blokovi koji se otvaraju sa znakom **{** i zatvaraju s **}**.
Svaki programski blok vidi samo varijable koje su deklarirane u njemu ili u njegovim roditeljskim blokovima, a ne vidi varijable deklarirane u blokovima unutar njega ili koji mu nisu izravno roditeljski.

```
1   //globalno stanje
2   int global_state;
3
4   void loop(){ //korijenski blok, vidi samo global_state i root
5       char root = 'a';
6
7       global_state = 5;
8
9       if ( root == 'a' ) {
10          //blok 1, vidi global_state, root i blok1_number
11          float blok1_number = global_state / 2;
12      }
13      //blok1_number nije vidljiv van bloka 1 pa ovaj kod nije
            valjan
14      int second_root = blok1_number;
15
16      while (global_state > 0) {
17          //blok 2, vidi global_state, root, second_root i blok2
18          //ali blok1_number nije vidljiv van bloka 1 pa ni ovaj
                kod nije valjan
19          int blok2 = blok1_number - 1;
20      }
21  }
```

pinMode(pin_number, mode) je konfiguracijska naredba koja Arduinu govori kako se pin (priključak) pod brojem *pin_number* treba ponašati.

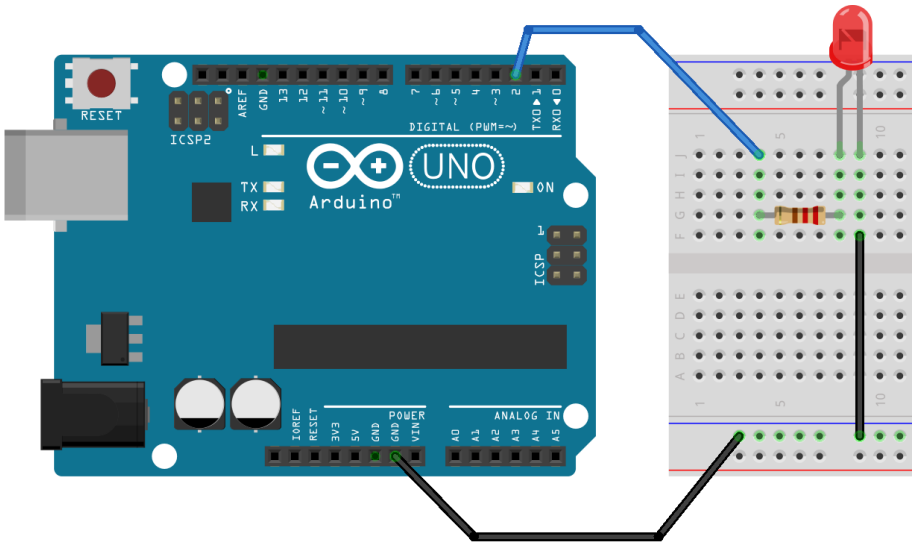
digitalWrite(pin_number, value) na pin broj *pin_number* postavlja vrijednost *value* koja može biti **HIGH** ili **LOW**, to jest **1** ili **0**. U fizičkim vrijednostima, to se na Arduinu prevodi u napone od **5V** ili **0V**.
Primjerice, za paljenje desetog pina bi to bilo *digitalWrite(10,HIGH);*
Da bi naredba *digitalWrite()* funkcionirala za određeni pin, prvo mu se u funkciji *setup()* način rada mora postaviti u **OUTPUT** koristeći *pinMode()*

digitalRead(pin) sa pina očitava digitalnu vrijednost (opet HIGH/LOW tj 1/0) te ju pohranjuje u varijablu.
Za očitavanje vrijednosti tipkala bi prvo van *setup()* i *loop()* funkcija deklarirali varijablu što ju čini globalnom i omogućava da njena vrijednost perzistira između izvođenja funkcije.

```
1   #define BUTTON_PIN 2
2   int button_state;
3   void setup() {
4       pinMode(BUTTON_PIN, INPUT_PULLUP);
5   }
6   void loop() {
7       button_state = digitalRead(BUTTON_PIN);
8   }
```

2.1 Blink

Umjesto ugrađene LED-ice, sada je potrebno spojiti i kontrolirati vanjsku LED-icu po shemi prikazanoj na slici 11



Slika 11: Spajanje LED-ice

Nakon spajanja po shemi, prilagoditi ćemo postojeći **Blink** primjer kod kako bi mijenjali stanje na pinu 2.

2.2 Zadatak 1: Trčeće svjetlo

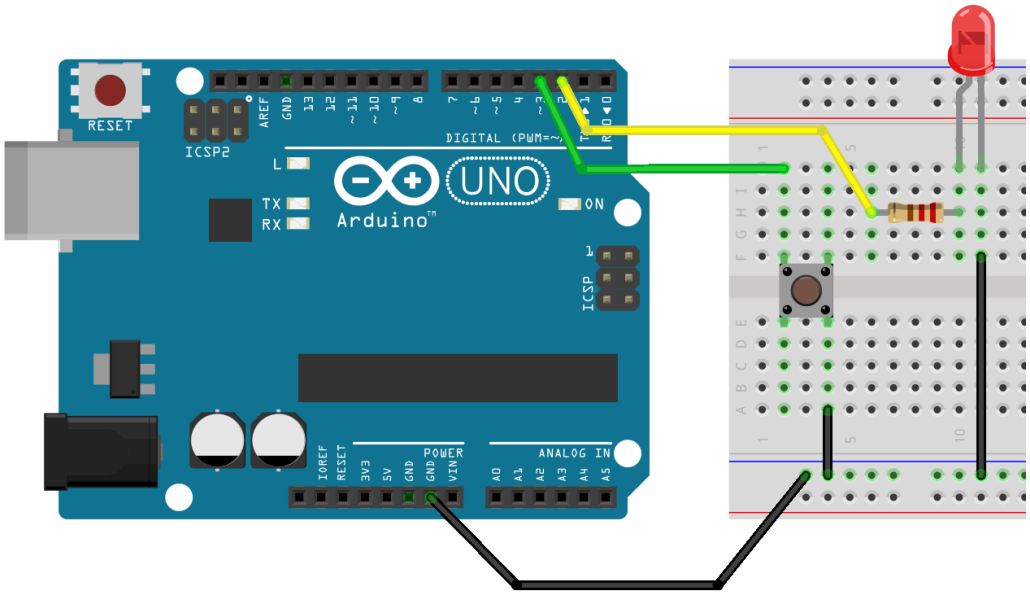
Sad kada znamo spojiti i upravljati jednom LED-icom, možemo ostvariti i nešto kompleksniji sklop. Potrebno je spojiti tri LED-ice na tri zasebna pina te ih upogoniti u tzv. "trčeće svjetlo".

2.3 Tipkalo i digitalni ulaz

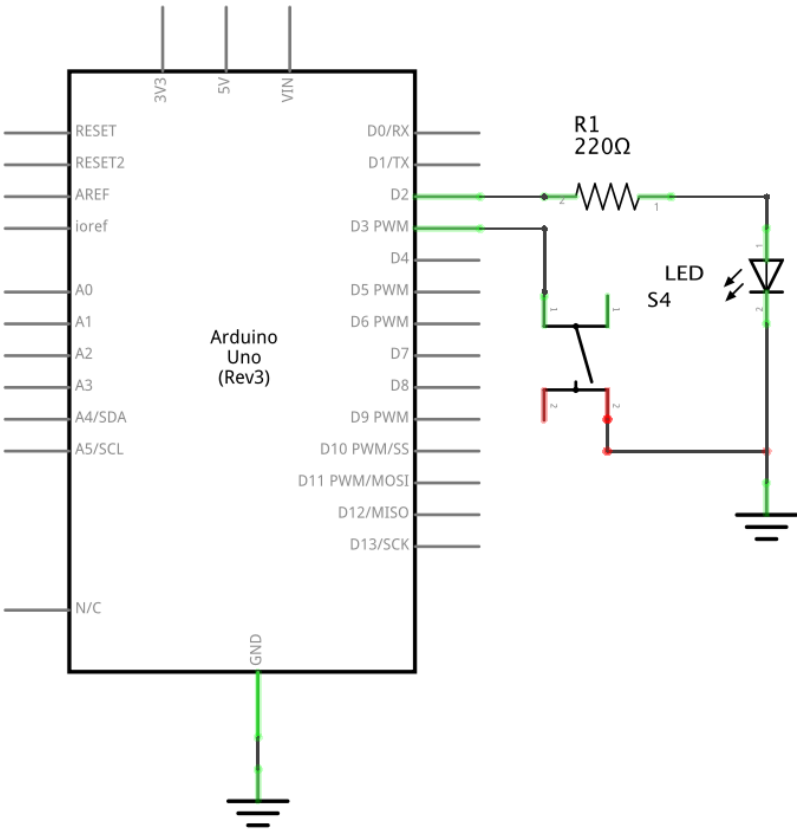
Svaki Arduino pin koji može funkcionirati kao izlaz, može se koristiti i kao digitalni ulaz. Vrijednost koja se očita funkcijom *digitalRead()* se može odmah predati drugoj funkciji na korištenje ili se može pohraniti u varijablu pa koristiti na više lokacija.

Bitan pojam pri čitanju digitalnih ulaza su pull-up i pull-down otpornici. Budući da je prije pritiskanja tipkala strujni krug otvoren, napon nije nužno blizu 0 ili 5V te se njegova vrijednost ne mora nužno ispravno interpretirati kao 0 ili 1 tj. LOW/HIGH. Stoga se digitalni ulazi obično preko otpornika (kako bi se ograničila struja) spajaju na 0 ili 5V. U svrhu praktičnosti, Arduino sadrži ugrađene pull-up otpornike koji se mogu aktivirati naredbom *pinMode(pin_number, INPUT_PULLUP);* umjesto *pinMode(pin_number, INPUT);* koji bi zahtijevao vanjsko spajanje pull-up otpornika.

S obzirom da u ovom primjeru koristimo ugrađeni pull-up otpornik, spojit ćemo shemu prema slici 12 i programirati dolje navedeni kod.



Slika 12: Spajanje tipkala s LED-icom



Slika 13: Tipkalo i LED - shema

```
1 const int button = 3;
2 const int led = 2;
3
4 int button_state = 0;
```

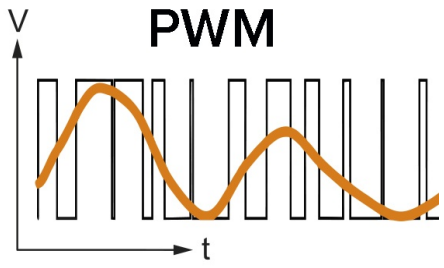
```
5
6 void setup() {
7   pinMode(led, OUTPUT);
8   pinMode(button, INPUT_PULLUP);
9 }
10
11 void loop(){
12   button_state = digitalRead(button);
13   digitalWrite(led, button_state);
14
15   //takoder moguce gornje dvije naredbe
16   //zamijeniti s digitalWrite(led, digitalRead(button));
17 }
```

Ponaša li se LED-ica kako ste očekivali? Kako bi izmijenili logiku uključivanja LED-ice ?

3 Analogna sučelja

Rad s digitalnim vrijednostima 0 i 1 tj. uključeno i isključeno u interakciji sa stvarnim svijetom ipak nije dovoljan jer su u mnogo primjena potrebne različite naponske razine. Senzori, potenciometri, zvučni uređaji i mnogi drugi rade s analognim vrijednostima koje se nalaze u spektru između minimalnih i maksimalnih naponskih vrijednosti. Primjerice, senzor temperature neće uvijek vratit 0°ili 100°nego decimalnu vrijednost na tom rasponu.

Za očitanje analognih vrijednosti, Arduino Uno je opremljen s šest analognih ulaza grupiranih u donjem desnom kutu pločice i označenih u rasponu *A0-A5*. Analogni izlazi pinove dijele s određenim digitalnim pinovima (*3, 5, 6, 9, 10, 11*) jer zapravo koriste izrazito brze digitalne impulse (tzv. PWM modulacija) kako bi postigli analognu vrijednost na izlazu kao što je vidljivo na slici 14.



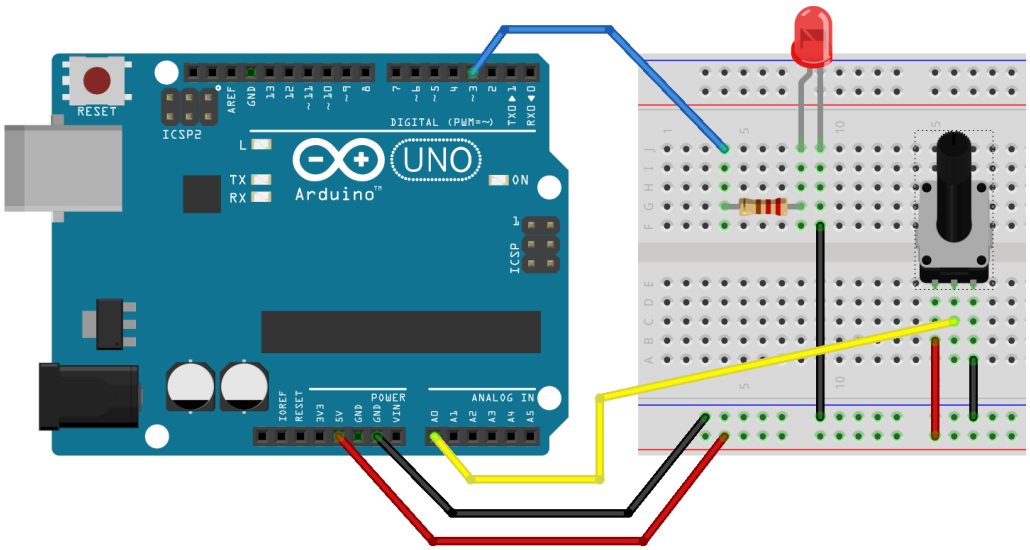
Slika 14: Postizanje analogne vrijednosti digitalnim impulsima

3.1 Analogni ulaz

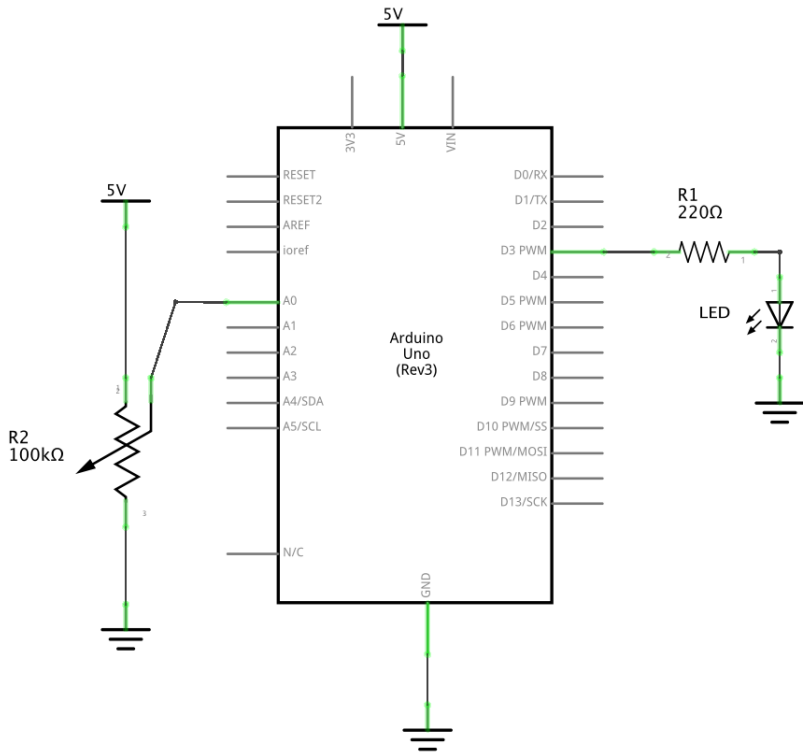
Za čitanje analognih vrijednosti se koristi funkcija *analogRead(analog_pin)* koja na ulazu *analog_pin* mjeri naponsku razinu u rasponu 0 do 5V te ju preslikava u 10 bitova, tj. u cjelobrojni raspon **0-1023**. U slučaju da nam taj povratni raspon ne odgovara, možemo koristiti funkciju *map(value, fromMin, fromMax, toMin, toMax)* koja prima očitanu vrijednost, raspon u kojem se ona nalazi i raspon u koji ju želimo preslikati te vraća preslikanu vrijednost. Primjerice, ako bi htjeli izlaz funkcije *analogRead()* preslikati u postotke, pozvali bi

```
1 int analog = analogRead(A0);
2 int percentage = map(analog, 0, 1023, 0, 100);
```

Koristeći gore navedene funkcije, LED-ici iz prethodnih zadataka sada možemo promijeniti frekvenciju treptanja. Za njenu regulaciju koristiti ćemo potenciometar spojen na analogni ulaz prema shemi sa slike 15



Slika 15: Spajanje potenciometra na analogni ulaz



Slika 16: Potenciometar i analogni ulaz - shema

Nakon toga možemo učitati kod:

```
1 const int led = 3;
2 const int pot_pin = A0;
3
4 void setup() {
5   pinMode(led, OUTPUT);
6 }
7
8 void loop(){
9   int value = analogRead(pot_pin);
10
```

```
11    digitalWrite(led, LOW);
12    delay(value);
13    digitalWrite(led, HIGH);
14    delay(value);
15  }
```

3.1.1 Zadatak 2: *map()*

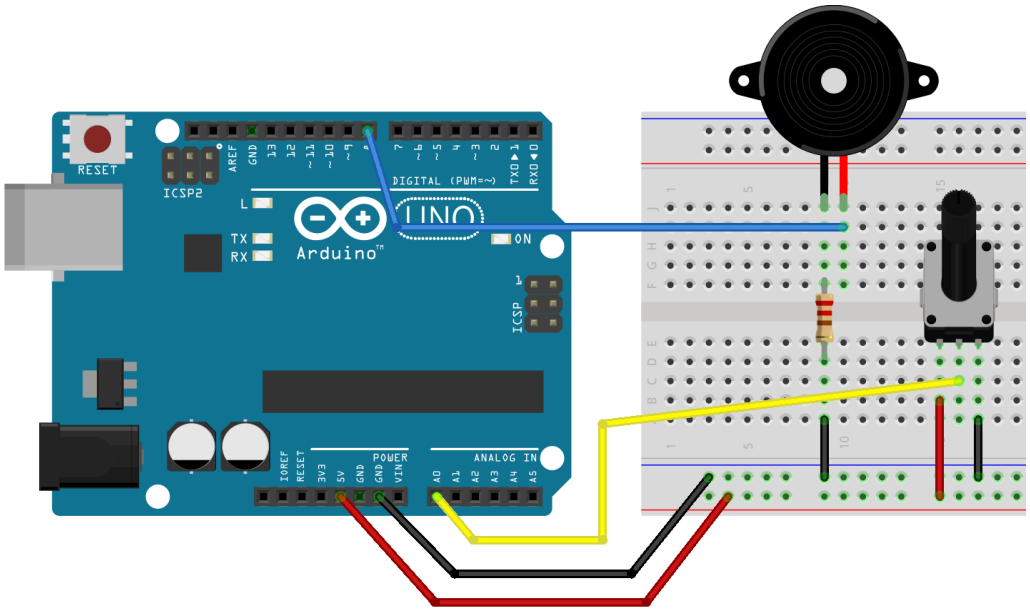
Izmijenite gore napisani kod uz pomoć funkcije *map()* kako bi povećali frekvencijski raspon treperenja LED-ice.

3.2 Analogni izlaz

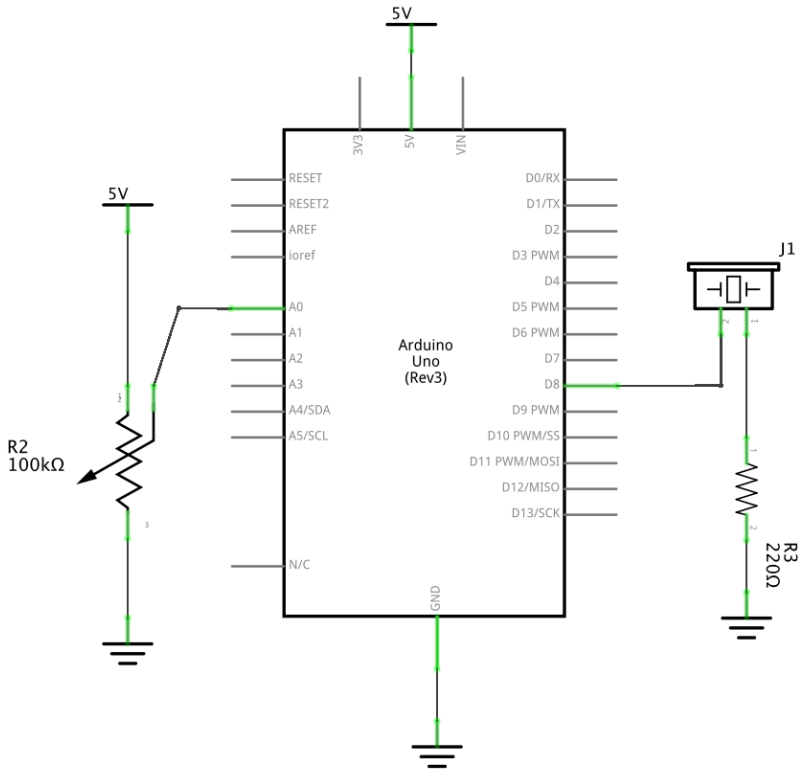
Kao što je ranije navedeno, određeni digitalni izlazi na Arduinu imaju mogućnost generiranja analognih vrijednosti što je korisno za uređaje upravljane analognim vrijednostima poput zvučnika, motora, ventila i njima sličnih.

3.2.1 Zvučnik

Za upravljanje piezoelektričnim zvučnikom postoji posebna funkcija *tone(pin, frequency, duration)* koja prima *pin*, frekvenciju i trajanje zvučnog signala što uvelike olakšava generiranje zvučnih signala. Isprobajmo primjer prikazan na shemi 17



Slika 17: Spajanje piezoelektričnog zvučnika



Slika 18: Piezoelektrični zvučnik - shema

```
1 const int speaker_pin = 8;
2 const int pot_pin = A0;
3
4 void setup() {
5   pinMode(speaker_pin, OUTPUT);
6 }
7
8 void loop() {
9   int pot_value = analogRead(A0);
10
11   int pitch = map(pot_value, 0, 1023, 200, 5000);
12   tone(speaker_pin,pitch,20);
13
14   delay(10);
15 }
```

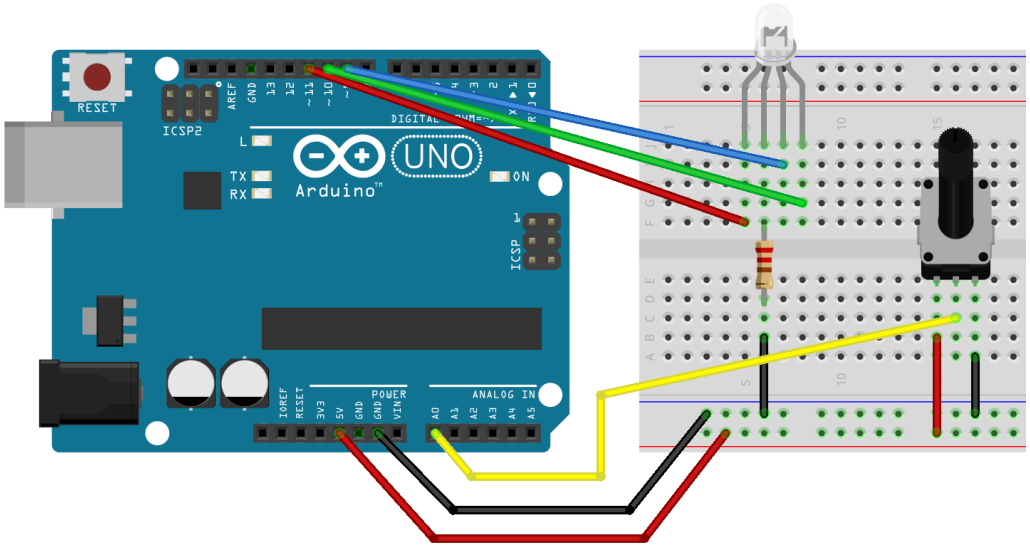
3.2.2 Zadatak 3: Zvučnik i tipkalo

Na gore navedenu shemu dodaje tipkalo spojeno na zaseban digitalni ulaz te izmijenite kod tako da se na pritisak tipkala mijenja frekvencija tona.

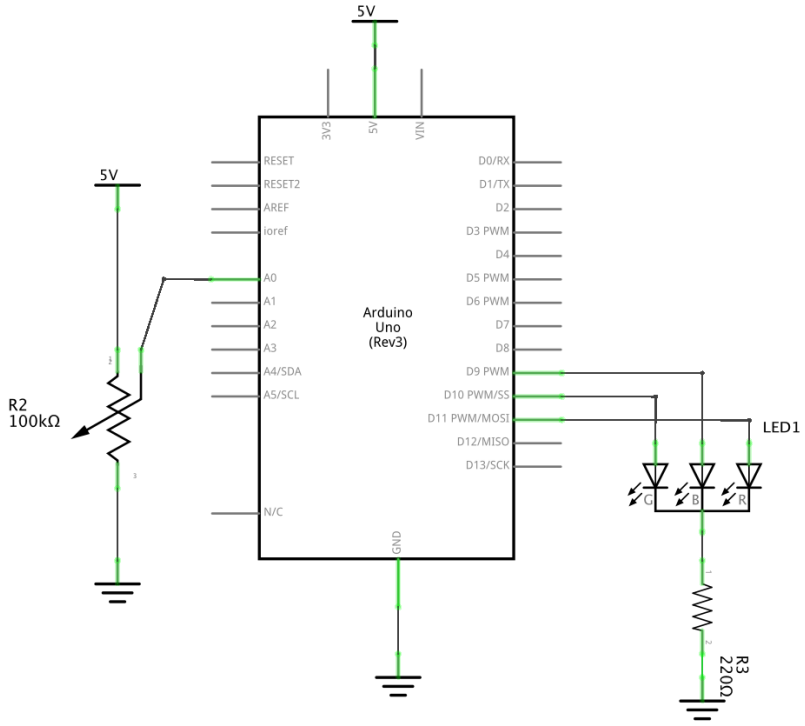
3.2.3 RGB svjetleća dioda

Osim za izmjenu tona, analogni izlaz se može koristiti i za upravljanje intenzitetom svjetlosti. Ovo je korisno za upravljanje RGB LED-icom koju možemo promatrati kao tri diode različite boje unutar jednog kućišta. Izmjenom intenziteta crvene, zelene i plave komponente lako se ostvaruju različite boje. S obzirom da dijele kućište, sve tri LED-ice imaju zajedničku katodu (negativni pol) i odvojene anode (pozitivni pol).

U niže prikazanom primjeru ćemo očitati analognu vrijednost s potenciometra te ju mapirati na širok spektar različitih boja.



Slika 19: Spajanje potenciometra s RGB LED-icom



Slika 20: Potenciometar i RGB LED

```
const int rgb_red_pin = 11;
const int rgb_green_pin = 10;
const int rgb_blue_pin = 9;
const int pot_pin = A0;

void setup()
{
  pinMode(rgb_red_pin, OUTPUT);
  pinMode(rgb_green_pin, OUTPUT);
  pinMode(rgb_blue_pin, OUTPUT);
}
void loop()
```

```
{
  int potentiometerValue = analogRead(pot_pin);
  int rgbValue = map(potentiometerValue, 0, 1023, 0, 1535);

  int red;
  int blue;
  int green;

  if (rgbValue < 256) {
    red = 255;
    blue = rgbValue;
    green = 0;
  }
  else if (rgbValue < 512) {
    red = 511 - rgbValue;
    blue = 255;
    green = 0;
  }
  else if (rgbValue < 768) {
    red = 0;
    blue = 255;
    green = rgbValue - 512;
  }
  else if (rgbValue < 1024) {
    red = 0;
    blue = 1023 - rgbValue;
    green = 255;
  }
  else if (rgbValue < 1280) {
    red = rgbValue - 1024;
    blue = 0;
    green = 255;
  }
  else {
    red = 255;
    blue = 0;
    green = 1535 - rgbValue;
  }

  analogWrite(rgb_red_pin, red);
  analogWrite(rgb_blue_pin, blue);
  analogWrite(rgb_green_pin, green);
}
```

U navedenom kodu je korišten uvjetni blok *if-else* koji ovisno o predanim logičkim izrazima, izvršava različite programske blokove. Dakle, koristeći if-else uvjete možemo kontrolirati izvođenje programa.

```
if(state < 8) {
// ukoliko je uvjet u zagradama istinit
// izvrsava se kod u ovom bloku
} else if (state < 3) {
// ukoliko prvi uvjet nije bio istinit, a ovaj jest
// izvrsava se kod u ovom bloku
} else {
// ukoliko nijedan od uvjeta do sad nije bio istinit
// izvrsava se kod u ovom bloku
}
```

else if () i **else** blokovi su opcionalni te je moguće koristiti **if ()** samostalno. Također, osim logičkih izraza koji rezultiraju istinom ili laži, za uvjet je moguće koristiti i brojke po principu **0 = laž, sve druge brojke su istina**.

4 Serijska komunikacija

Kada se mikrokontroleri koriste u većim sustavima poput proizvodnih pogona, jedan mikrokontroler često nije dovoljan pa se koristi veći broj povezan u neki oblik mreže. Zbog smanjenja broja kablova, najčešće se koristi neki oblik serijske komunikacije. Među Arduino periferijom se najčešće koriste UART, I2C i SPI protokoli, s tim da SPI ovaj put nećemo obraditi.

4.1 UART

UART je serijski protokol koji preko dvije žice (plus uzemljenje) omogućava duplex komunikaciju, tj. istovremeno slanje i primanje podataka. Ograničen je na komunikaciju između samo dva uređaja te s obzirom da nema vod za takt, oba uređaja moraju imati konfiguriranu istu frekvenciju

Na Arduino Uno postoji samo jedan UART port koji se koristi za komunikaciju između Arduina i računala kojim ga programiramo. Koristan je za dijagnostiku koda i otkrivanje pogrešaka. Primjerice, ukoliko u kodu iz poglavlja 3.1 želimo provjeriti na koju vrijednost se postavlja period možemo dodati serijsku komunikaciju s ove dvije linije:

```
1  const int led = 3;
2  const int pot_pin = A0;
3
4  void setup() {
5    pinMode(led, OUTPUT);
6    Serial.begin(9600); //inicijalizira komunikcaiju na 9600 baud
7  }
8
9  void loop(){
10   int value = analogRead(pot_pin);
11   Serial.println(value); //salje vrijednost preko serijskog porta na
         konzolu
12
13   digitalWrite(led, LOW);
14   delay(value);
15   digitalWrite(led, HIGH);
16   delay(value);
17 }
```

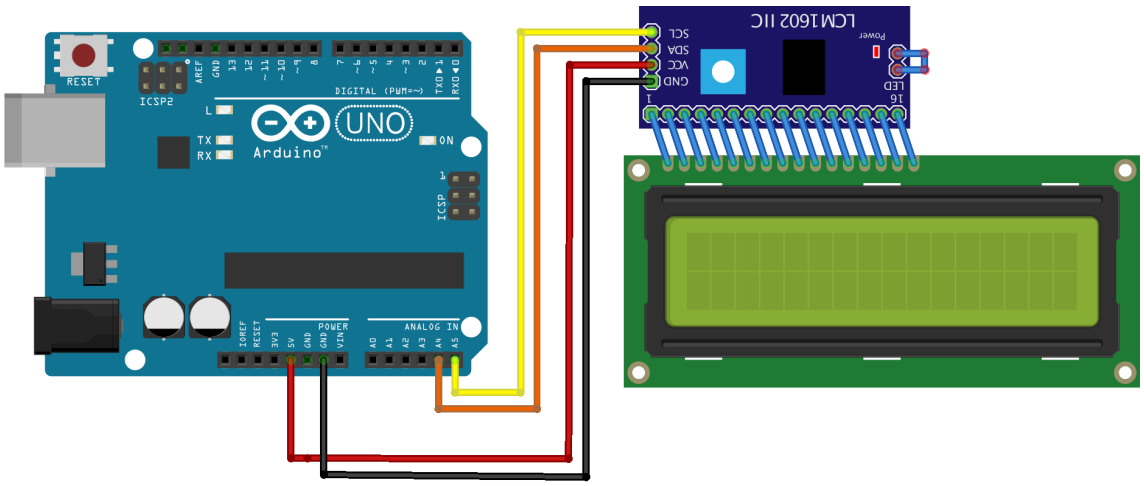
Nije potrebno spajati dodatne kablove jer se u ovom slučaju podaci šalju preko USB kabela. Da bi mogli vidjeti ispis, potrebno je otvoriti serijsku konzolu. To možemo u Arduino IDE klikom na **Tools -> Serial Monitor** čime se otvara prozor u kojem se ispisuju podaci koje šalje Arduino, ali je moguće i poslati podatke nazad preko polja za unos.

4.2 I2C

Protokol I2C također koristi dvije žice (plus uzemljenje), ali umjesto žice za slanje i žice za primanje ima žicu za takt i žicu za podatke. To znači da samo jedan uređaj može slati podatke u određenom trenutku, ali i da međusobno može komunicirati mnogo više uređaja ako im se raspodjele adrese. Jedna I2C sabirnica može podnijeti 127 različitih uređaja/adresa, ali ćemo ovaj put koristiti samo jedan.

4.3 LCD ispis

Hitachi-ev LCD sa 16 znakova u dva reda se desetljećima primjenjuje u svemu od fax uređaja do caffe aparata te uz pomoćni sklop (vidljiv na pozadini) podržava I2C komunikaciju. Uz pomoćnu knjižnicu *LiquidCrystal_I2C* možemo jednostavnim funkcijama ispisivati znakove na ekranu



Slika 21: Spajanje LCD-a

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2);

void setup()
{
lcd.init();
  lcd.backlight();

  lcd.setCursor(1,0);
  lcd.print("Pozdrav");
  lcd.setCursor(1,1);
  lcd.print("Jura");
}


void loop()
{
  // ne treba nista u loop
}
```

Probajte u loop funkciji dodati kod takav da se svakih 5 sekundi ispiše Vaše ime umjesto Jurinog.

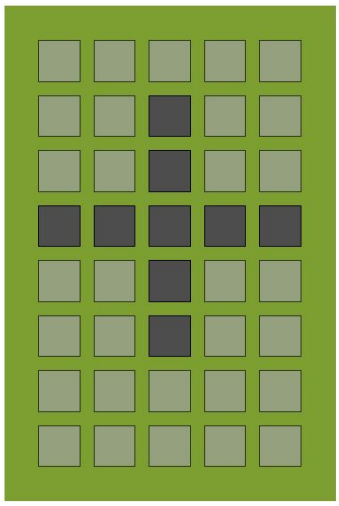
4.3.1 Novi znakovi

U Hitachi-evom LCD kontroleru je moguće definirati osam personaliziranih znakova. Znak se definira funkcijom *lcd.createChar(char_num, char_array)* gdje je *char_num* broj od 0 do 7 i predstavlja jedan od 8 personaliziranih znakova, a *char_array* polje vrijednosti koje definiraju piksele u novom znaku.

Polje u C programskom jeziku označava niz vrijednosti. Jedan znak na LCD-u se sastoji od osam redova po pet piksela pa je potrebno polje od 8 brojeva da bi se definirao novi znak jer svaki broj definira jedan red piksela. Tako bi, na primjer, polje bez upaljenih piksela izgledalo ovako:

```
byte char_array[8] = { 0, 0, 0, 0, 0, 0, 0, 0};
```

a polje koje definira znak plusa bi izgledao ovako:



Slika 22: Pikseli znaka plus

gdje **0x** označava da je broj zapisan u heksadecimalni bazi.
Tako bi kod za ispis srca izgledao ovako:

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

byte customChar[8] = {
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000,
  0b00000
};

void setup()
{
  lcd.init();
  lcd.backlight();

  lcd.createChar(0, customChar);

  lcd.setCursor(2, 0);
  lcd.write((byte)0);
}

void loop()
{
}
```

4.3.2 Zadatak 4: Izrada novih znakova

Ukoliko Vaše ime i prezime sadrži dijakritičke znakove, dodajte znakove i ispišite Vaše puno ime i prezime. Ukoliko to nije slučaj, dodajte potrebne znakove i ispišite ime *Šimširpašić Sabahudin*.

Ukoliko Vam je potrebna pomoć s dizajnom, možete se poslužiti online alatom za dodavanje znakova

5 Izvori

- Službeno Arduino web sjedište: https://www.arduino.cc
- Download Arduino IDE razvojnog okruženja: https://www.arduino.cc/en/software
- Objašnjenja korištenih funkcija i varijabli: https://www.arduino.cc/reference/en/
- Simulator projekata za korištenje u web pregledniku: https://wokwi.com/

6 Preporučena literatura

- Arduino knjiga projekata (Arduino Projects Book - by Scott Fitzgerald, Michael Shiloh)