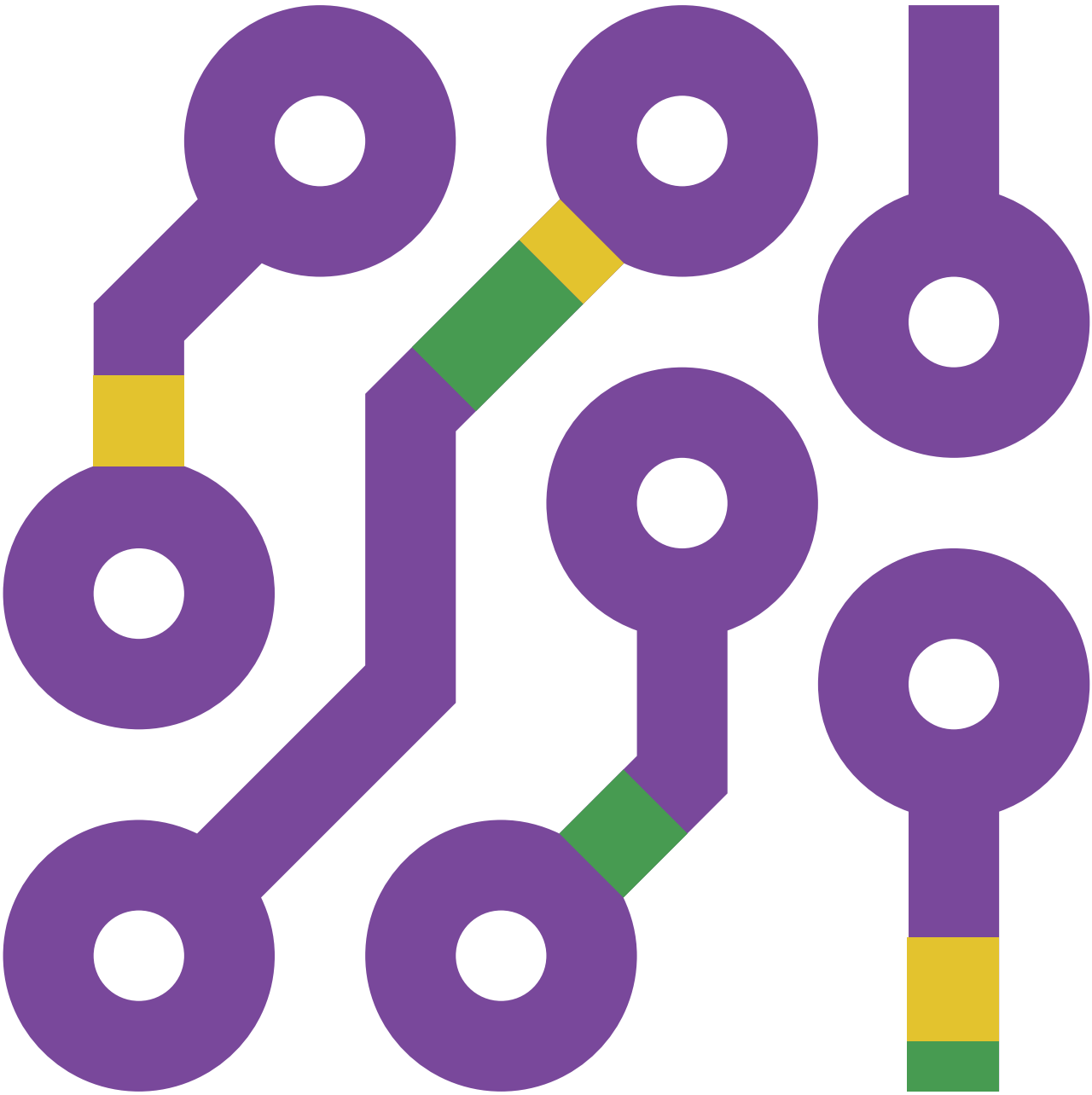


ARDUINO ZA SVE

PRIRUČNIK ZA NASTAVNIKE
Srednja škola

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

Mr. sc. Ajla Halilović,
dipl. matematičarka i informatičarka

< IT Girls >

Podržano od:
UJEDINJENE NACIJE
BOSNA I HERCEGOVINA

ARDUINO ZA SVE

PRIRUČNIK ZA NASTAVNIKE
Srednja škola

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

Mr. sc. Ajla Halilović, dipl. matematičarka i informatičarka

2021.

Sadržaj

Uvod

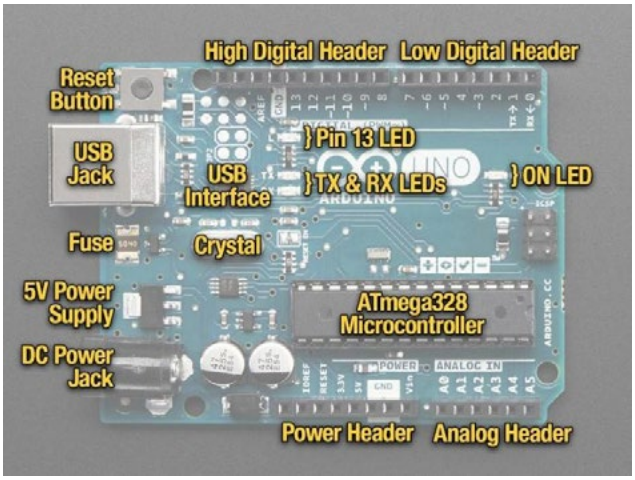
Naši se životi vrte oko megabajta, megaherca, čipova, procesora, AI, IoT, robota, ma jednostavno čitava zbrka pojmova i, priznajete, nekada se pretvaramo da ih razumijemo, a nekada pomislimo kako bi bilo baš dobro koristiti i stvarati, jednostavno biti u tom svijetu, neki bi rekli, shvatiti matricu (op.a. MATRIX). Većina pojmova koji će se ovdje spominjati i nisu tako nepoznati, ali je neophodno napraviti uvod u svijet kroz koji ćemo zajedno kročiti s Arduinom.

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

Broj prodanih originalnih platformi Arduino je milijunski, a broj onih kloniranih vjerojatno nikada nećemo ni saznati.

Platforma Arduino

Upoznajmo prvo platformu Arduino:



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

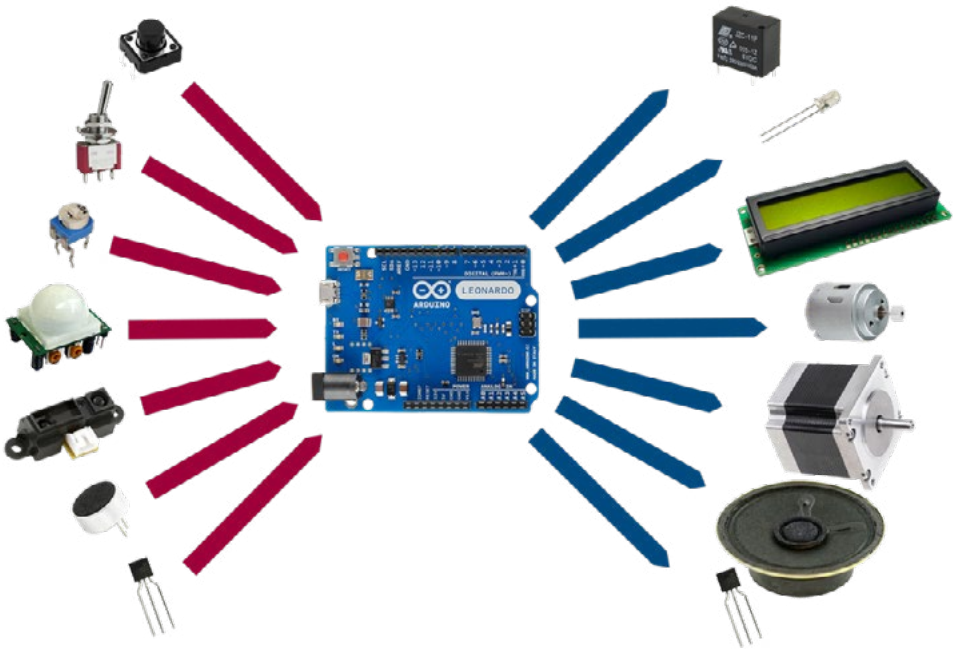
Za rad s ovom platformom nije neophodno veliko predznanje iz elektrotehnike, mada se na prvu tako i ne čini.

S ukupno 14 digitalnih ulazno-izlaznih pinova i 6 analognih 10-bitnih ulaza, Arduino Uno ili neka druga ploča Arduino sasvim je dovoljna da kreiramo vrlo korisne projekte.

Senzori i aktuatori

Komponente koje vezujemo na platformu možemo podijeliti u dvije skupine: senzore i aktuatore.

Mi svakodnevno imamo interakciju sa senzorima i aktuatorima, a možda je jednostavnije kada kažemo ulaznim i izlaznim uređajima.



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

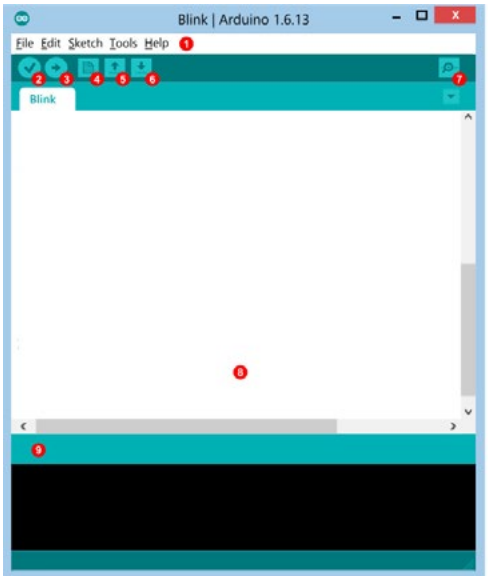
IDE Software Arduino

Upoznali smo se s platformom i komponentama, te nam preostaje da instaliramo IDE Software Arduino. Riječ je o besplatnoj aplikaciji koju koristimo da bismo kreirali ili razmjenjivali informacije s mikrokontrolerom Arduino. Aplikacija se može preuzeti sa sljedeće mrežne lokacije:

https://www.arduino.cc/en/software

Softver Arduino stalno se revidira. U trenutku pisanja ovog priručnika aktualna je inačica 1.8.15, ali je vrlo moguće da dok ovaj priručnik dođe do krajnjih korisnika, novija inačica softvera bude dostupna za instalaciju. Potrebno je odabrati instalaciju prema ponuđenim opcijama i proći kroz krake instalacije.

Prije nego što krenemo u programiranje, pogledajmo Arduino IDE okruženje.



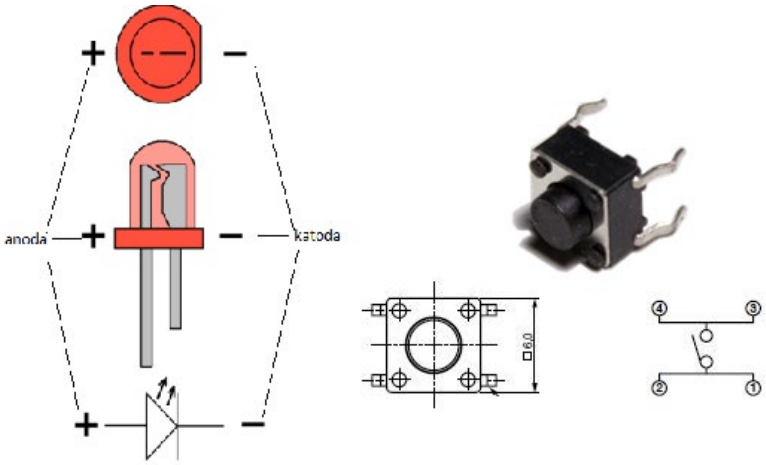
1. **Menu:** Meni softverskih opcija.
2. **Verify:** Kompajliranje i provjera koda.
3. **Upload:** Učitavanje "koda" u Arduino
4. **New:** Nova Arduino skica
5. **Open:** Otvori postojeću Arduino skicu.
6. **Save:** Sačuvaj trenutno aktivnu skicu.
7. **Monitor:** Otvori aplikaciju za slanje i primanje informacija.
8. **Editor:** Prostor za pisanje koda.
9. **Message:** IDE prostor za izvještaje o greškama i info.

Koristeći USB kabel Arduino spajamo na slobodni USB port. Dobra je rutina u Windows Device Manageru provjeriti kojem je portu COM (komunikacijskom sučelju) pridružen Arduino. (Halilović, 2019)

Za početak smo pripremili jednostavnije vježbe koje ćete moći dalje razvijati i primjenjivati kroz različite projekte.

Vježba 1. Toplo i hladno!

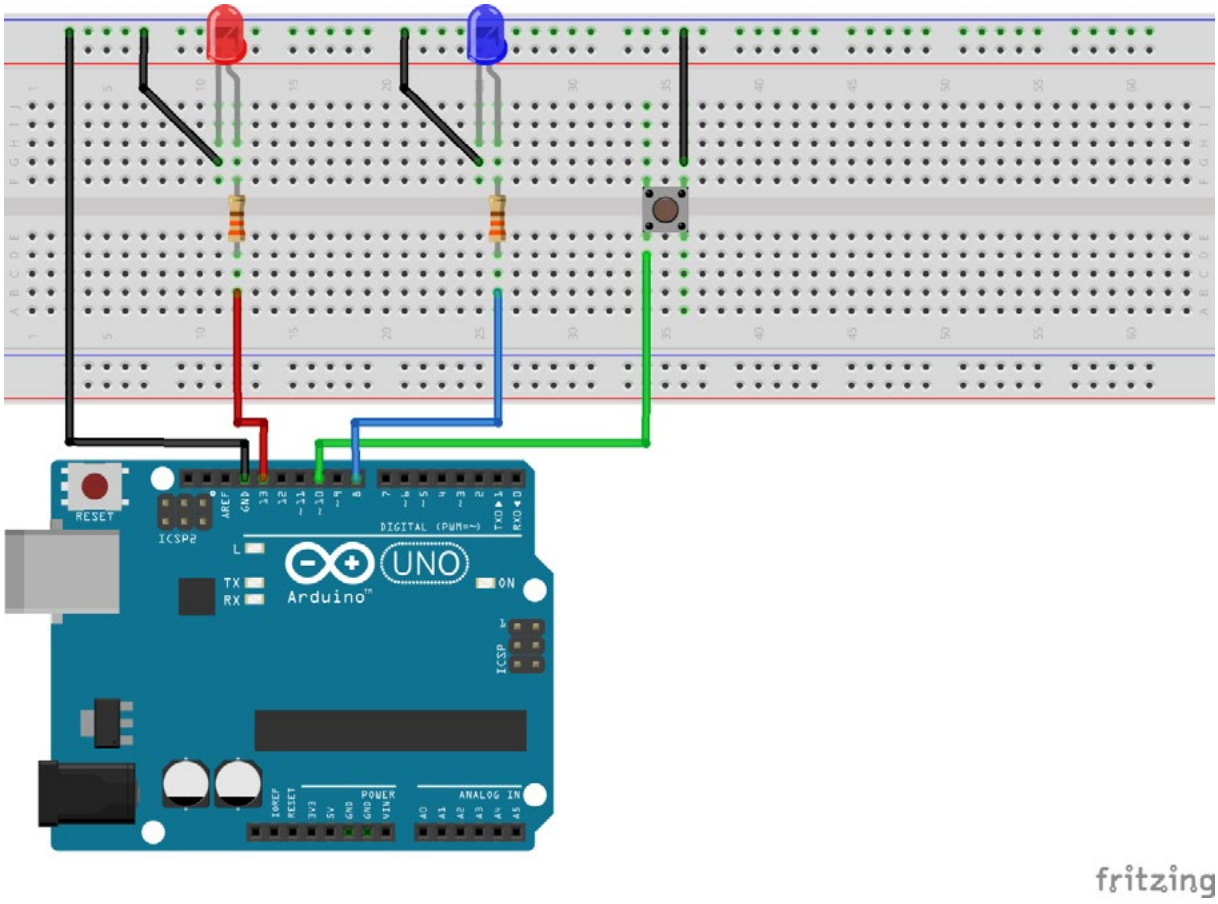
Za one koji su se odvažili i prvi put kreću u programiranje Arduina ova je aplikacija idealna kao prva vježba: na Arduino treba spojiti dvije LE diode i jedan taster. Kada se taster pritisne, treba svijetliti crvena, a inače je uključena plava LE dioda.



Slika vj. 1.1. LE dioda i taster – izgled i simbol

Potrebni elementi za realizaciju vježbe:

1. UNO pločica Arduino i USB 1 kom.
2. Pločica Matador 1 kom.
3. LE diode 2 kom.
4. Otpornik 220 Ω 2 kom.
5. Taster 1 kom.
6. Kablići



Slika vj. 1.2. Shema spoja

Kod Arduino:

Sve Arduino skice sadrže dvije osnovne funkcije: setup i loop.

Setup je funkcija u kojoj se slijedno izvršavaju programske instrukcije, ali samo jednom tijekom izvršavanja aplikacije. Dakle prilikom svakog pokretanja aplikacije kod u funkciji setup izvrši se samo jednom. Samo ime funkcije kaže da se unutar nje radi podešenje mikrokontrolera, definira se koji će pinovi biti ulazni, a koji izlazni, podešavaju se ili aktiviraju neke specijalne funkcije samog mikrokontrolera.

U funkciji loop() nalaze se sve naredbe koje Arduino treba izvršiti. Za razliku od naredbe setup() naredba loop() izvršava se beskonačno mnogo puta, dokle god je Arduino uključen. Loop se primjenjuje nakon definiranja početnih vrijednosti u funkciji setup. (Vuković, 2017)

U narednom primjeru upoznat ćemo se s nekoliko osnovnih funkcija u Arduino skicama.

Unutar dijela funkcije ***setup*** imamo pozvanu funkciju:

```
pinMode(LedCrvena, OUTPUT);
```

Funkcija ima dva argumenta: prvi podrazumijeva broj pina kojem želimo dati specifičnu funkciju, a s drugim argumentom definiramo hoće li neki pin biti INPUT(na pin spojeni taster, digitalni senzor ili prekidač), OUTPUT(na pin spojena LED, relej ili tranzistor za uključenje većih potrošača tipa motora, grijača itd.).

Unutar dijela funkcije ***loop*** imamo pozvane dvije funkcije:

digitalRead(Taster);

digitalWrite(LedCrvena, HIGH);

Prva funkcija prati promjenu stanja na pinu koji je proglašen INPUTOM i ako je "taster" aktiviran, funkcija vraća "1", u suprotnom funkcija vraća vrijednost "0". Funkcija nam omogućava detekciju promjene stanja na pinu koji je proglašen digitalnim ulazom.

Druga funkcija ima dva argumenta, prvim definiramo pin kojem želimo mijenjati naponsko stanje, a drugim vrijednost na tom pinu. Kako je riječ o digitalnim izlazima koji mogu imati dva stanja – uključeno ili isključeno, to argument HIGH znači "uključi" pin, a LOW "isključi" pin. Ako govorimo o naponu HIGH, znači 5V, a LOW 0V.

```
int LedCrvena = 13;
int LedPlava = 8;
int Taster = 10;

int StanjeTastera;

void setup() {
  pinMode(LedCrvena, OUTPUT);
  pinMode(LedPlava, OUTPUT);
  pinMode(Taster, INPUT_PULLUP);
}

void loop() {
  StanjeTastera = digitalRead(Taster);
  if (StanjeTastera == LOW) {
    digitalWrite(LedCrvena, HIGH);
    digitalWrite(LedPlava, LOW);
  } else {
    digitalWrite(LedCrvena, LOW);
    digitalWrite(LedPlava, HIGH);
  }
}
```

Ova se vježba može i modificirati, nadopuniti i proširiti uz primjenu nekih drugih senzora i aktuatora kako bismo napravili složeniji projekt. Na primjer, ako koristimo senzor zvuka i ultrasonični senzor, možemo napraviti alarmni sustav.

Digitalni i analogni senzori

U setu Arduino imamo različite senzore koje možemo razvrstati u dvije skupine prema signalu koji daju na izlazu.

Digitalni senzori na izlazu daju signal koji može poprimiti sam određene vrijednosti amplitude i dostupan je u samo određenim vremenskim intervalima. Digitalni senzori imaju slovo D, koje se nalazi ispisano na komponentama. Neki od digitalnih senzora su:

- Digital Push Button – taster
- Touch Sensor – senzor dodira
- Tilt Sensor – senzor nagiba
- Vibration Sensor – senzor vibracija
- Magnetic Sensor – magnetni senzor

U prethodnoj vježbi koristili smo digitalni senzor taster. Na isti način možete izabrati neki drugi digitalni senzor i testirati rad aplikacije.

Analogni senzori na izlazu daju veličinu koja je kontinuirana u vremenu i po amplitudi. Analogni senzori imaju slovo A, koje se nalazi ispisano na komponentama. U analogne senzore spadaju:

- Ultrasonic Sensor – ultrazvučni senzor
- Rotation Sensor – potenciometar
- Moisture Sensor – senzor vlažnosti
- Steam Sensor – senzor prisustva pare
- Sound Sensor – senzor zvuka
- Ambient Light Sensor – senzor ambijentalnog osvjetljenja
- Grayscale Sensor – svjetlosni senzor
- Piezo Disk Vibration Sensor – senzor vibracije

Vježba 2. Vlažnost tla

Kreirati aplikaciju koju ćemo koristiti za mjerenje razine vlažnosti tla. U ovoj vježbi koristit ćemo jednu LE diodu koja će se uključiti ukoliko je neophodno povećati vlažnost tla.

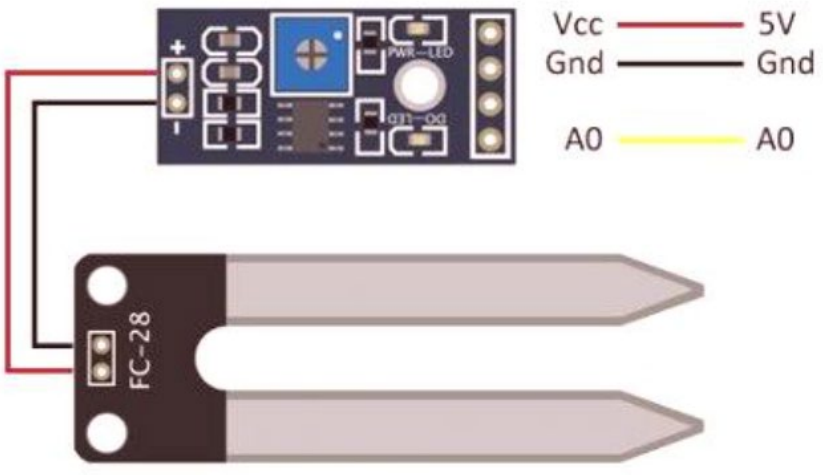
Ako imamo sobno bilje, onda ovu aplikaciju možemo primijeniti u razdoblju kada smo odsutni. Neće nam trebati usluga susjeda za zalijevanje cvijeća. S pomoću senzora za vlažnost možemo napraviti automatski sustav navodnjavanja. A uz ovaj senzor može se napraviti mnogo različitih i korisnih projekata.

Senzor vlažnosti ima dvije sonde koje provode električnu struju kroz zemlju. Ako je tlo vlažno, tada je otpor između sondi manji. U suhom je zemljištu otpor veći. Arduino prihvaća ove vrijednosti, uspoređuje ih, ako je potrebno, uključuje, naprimjer, pumpu. Senzor se može koristiti na način da se spoji na analogni ulaz.

Za povezivanje preko analognog ulaza koristimo ulaz A0. Senzor vlage u tlu Arduino uzima vrijednosti od 0 do 1023. U vježbi smo postavili uvjet (`StanjeSenzora <400)` za uključivanje crvene LE diode, a to može biti promjenljivo u zavisnosti od toga koja je vlažnost zemljišta odgovarajuća za određenu biljku.

Senzor se spaja na sljedeći način:

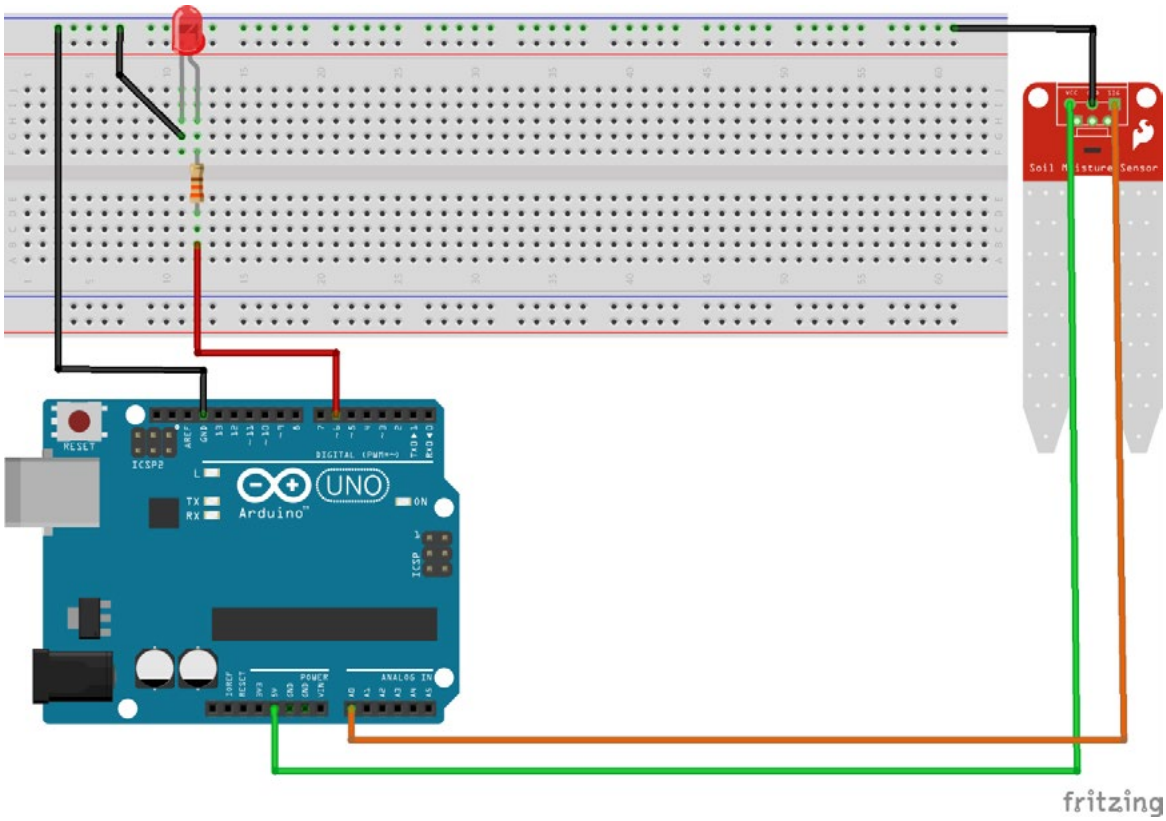
- VCC se povezuje s 5V na Arduino.
- GND na senzor na GND na ploči Arduino.
- A0 se povezuje s A0 na Arduinu.



Slika vj. 2.1. Moisture Sensor – senzor vlažnosti

Potrebni elementi za realizaciju vježbe:

1. Arduino UNO pločica i USB 1 kom.
2. Matador pločica 1 kom.
3. LE diode 1 kom.
4. Otpornik 220 Ω 1 kom.
5. Senzor za vlažnost 1 kom.
6. Kablići



Slika vj. 2.2. Shema spoja

Kod Arduino:

Unutar dijela funkcije ***setup*** imamo pozvanu funkciju:

```
Serial.begin(9600);
```

Serial.begin(9600) koristimo za podešavanje mikrokontrolera da aktivira serijsku komunikaciju na brzini od 9600 bits/sec. Serijska komunikacija omogućava nam da kroz samo dva voda označena sa RX (vod za primanje podataka) i TX (vod za slanje podataka) možemo poslati ili primiti kompleksne informacije tima stringa. Osim toga aktiviranjem i podešenjem serijske komunikacije dobivamo opciju tzv. softverskog *debugginga*. Naprimjer:

```
Serial.println ("sada sam u dijelu koda za odlučivanje!");
```

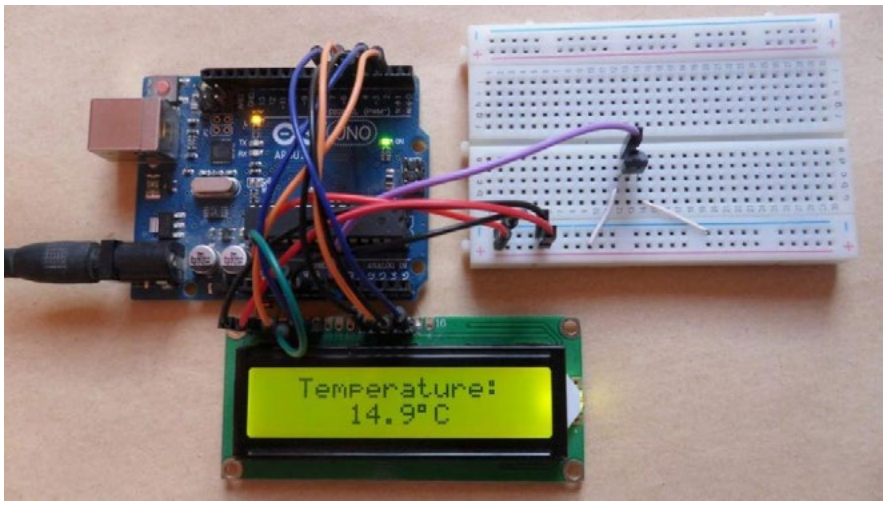
```
int StanjeSenzora;
int LedCrvena = 6;

void setup() {
  Serial.begin(9600);
  pinMode(LedCrvena, OUTPUT);
}
void loop() {
  StanjeSenzora = analogRead(A0);
  if (StanjeSenzora <400) {
    digitalWrite(LedCrvena, HIGH);
  } else {
    digitalWrite(LedCrvena, LOW);
  }
  Serial.println(StanjeSenzora);
  delay(100);
}
```

ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u vježbama	- definira osnovnu ulogu digitalnih i analognih senzora - opisuje primjenu LE diode u aplikacijama - koristi analogne i digitalne senzore u aplikacijama - opisuje ulogu senzora za vlažnost tla - koristi tastere u aplikacijama
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje elektronske komponente na Arduino prema shemi spoja - prepoznaje pinove na korištenim komponentama kako bi ih ispravno povezao
- primjenjuje stečeno znanje iz programiranja za izradu projekta	- kreira program za dati zadatak uz adekvatne smjernice nastavnika/-ice - po potrebi modificira kreirane kodove
- verificira i izvršava program	- analizira programski kod - izvršava otklanjanje pogreške po potrebi i prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - analizira projektni zadatak - razvija vlastite ideje za primjenu analognih i digitalnih senzora koji su korišteni u vježbama

S obzirom na to da smo se upoznali s osnovama Arduina, time smo spremni za kreiranje i složenijih projekata.

Projekt 1.
LM35

Kao prvi ozbiljniji projekt koji nam može poslužiti u mnogo korisnih aplikacija je mjerenje temperature s pomoću analognog senzora temperature LM35. Analogni ulazi na razvojnoj platformi Arduino su 10-bitne rezolucije. To nam govori da su vrijednosti od 0-1023 (2^10 = 1024), koje može dati bilo koji senzor kada je spojen na analogni ulaz. Taj podatak zovemo „sirovi" podatak i mi ljudi ga ne razumijemo. Da bismo dobili razumljiv podatak tipa vrijednost temperature, pritiska ili udaljenosti, potrebno je koristiti matematičke formule koje su specifične za svaki senzor i dobiju se iz opisa proizvoda proizvođača (op.a. *datasheet*). U ovom konkretnom projektu potrebno je realizirati jednostavan digitalni toplomjer sa senzorom LM35 i prikazom temperature na LCD zaslonu.



Slika pr. 1.1. *Prikaz temperature na LCD zaslonu*

Možda vam trenutačno pada na pamet da mjerite temperaturu u prostoriji u kojoj boravite, ali evo nekoliko ideja koje možete iskoristiti za neku nadogradnju ovoga početnoga projekta:

1. Mjerenje temperature kućišta PC-a i PCB grafičke kartice.
2. Mjerenje vanjske i unutarnje temperature plastenika.
3. Mjerenje temperature vode.

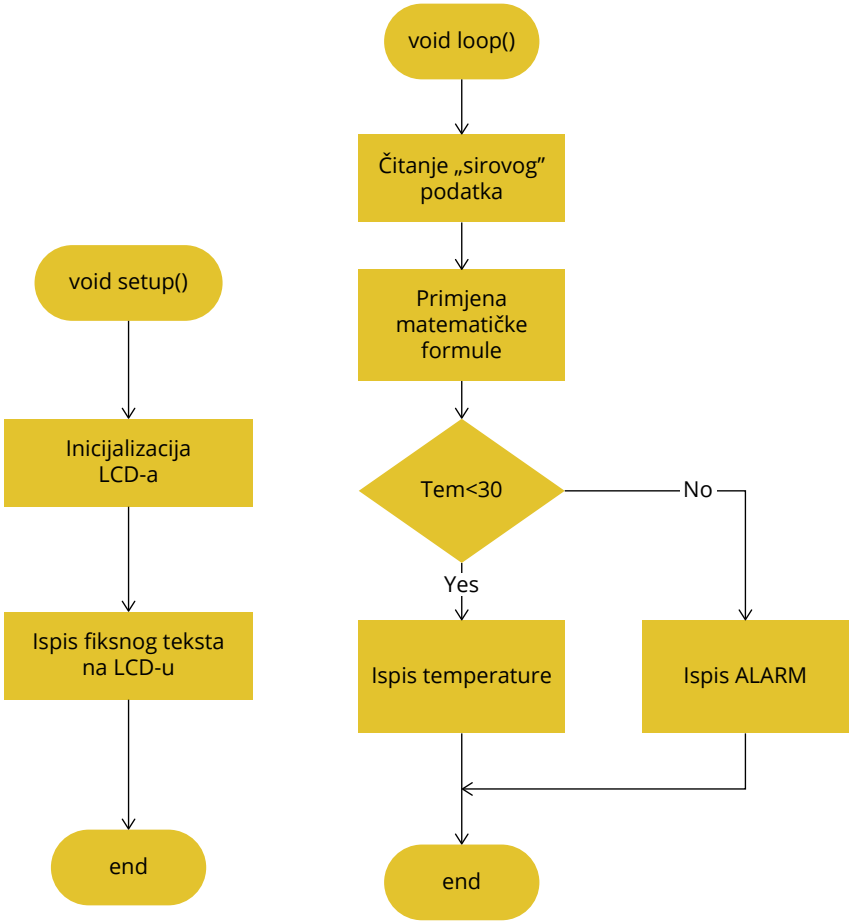
Svaki od spomenutih prijedloga možete nadograditi uvodeći dodatne komponente. Naprimjer, osim mjerenja temperature plastenika za uzgoj biljaka neophodno je i mjerenje vlažnosti tla.

Kada proširite svoje znanje, onda neće biti teško na platformu Arduino spojiti Bluetooth modul ili, što da ne, WiFi modul, te prikupljene podatke slati na neki mobilni uređaj ili mrežu. Ali više o ovome u nekome od narednih projekata.

Potrebni elementi za realizaciju projekta navedeni su ispod:

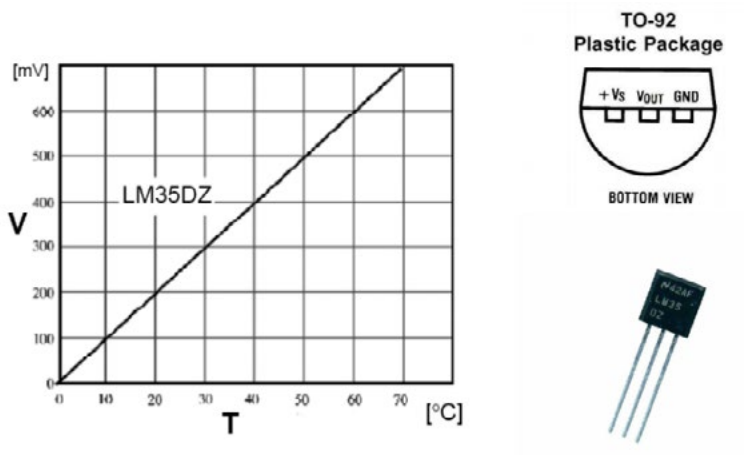
1. ARDUINO UNO
2. Temperaturni senzor LM35
3. 16×2 LCD zaslon
4. 10KΩ potenciometar
5. Ploča Matador
6. Kablići
7. Otpornik 330Ω

Bilo bi sjajno kada biste pokušali za svaki projekt koji zamislite prvo nacrtati grubu algoritamsku strukturu projekta. U ovom projektu mi ćemo to uraditi za vas, ali nadamo se da ćete steći naviku da prije nego počnete pisati kod, prvo taj kod probate vizualizirati i „nacrtati“ ga.



Slika pr. 1.2. *Pojednostavljena algoritamska struktura aplikacije*

Prema kataloškim podatcima senzor posjeduje osjetljivost od 10 mV/°C, a njegova prijenosna funkcija prilično je linearna i odstupanje se nalazi u granicama od ±0,1 °C.



Slika pr. 1.3. *Svojstvo i pinout LM35*

Kao što se vidi sa slike iznad, senzor temperature LM35 ima tri pina (**V_S, V_{OUT} i GND**). Ako bismo na krajeve senzora doveli napon u granicama od 3.3 do 30 V istosmjerno, a između krajeva V_{OUT} i GND spojili voltmetar, za izmjerene vrijednosti napona mogli bismo spram svojstva senzora zaključiti da senzor trenutačno mjeri sljedeće temperature.

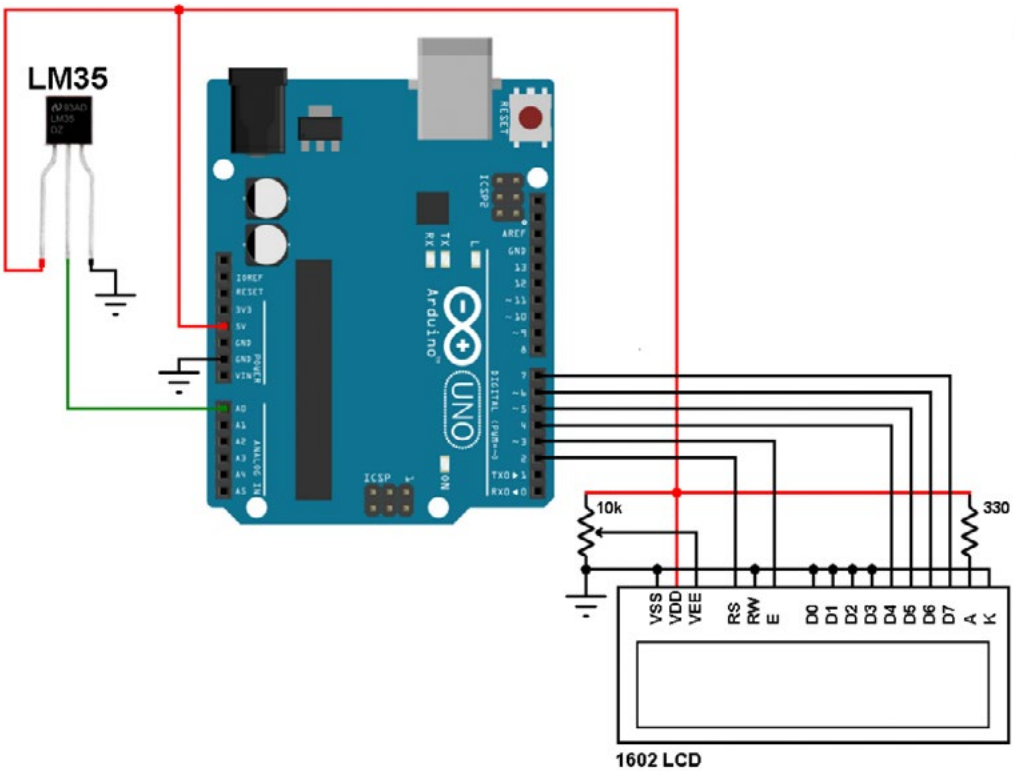
1. Za izlazni napon: 10 mV —> temperatura = 1 °C
2. Za izlazni napon: 100 mV —> temperatura = 10 °C
3. Za izlazni napon: 200 mV —> temperatura = 20 °C
4. Za izlazni napon: 370 mV —> temperatura = 37 °C

Ono što nam je kao programerima/-kama najvažnije jeste matematička formula s pomoću koje možemo dobiti nama razumljiv podatak:

*T = (Vcc * ADC * 100.0) / 1024;*

Celsius = Kelvin - 273.15

Celsius = 5/9 x (Fahrenheit -32)



Slika pr. 1.4. *Shema spoja*

Kod Arduino:

Knjižnicu LyquidCrystal.h pozivamo u svoj projekt kod programiranja aplikacija s LCD zaslonom. LCD je baziran na Hitachi HD44780 chipsetu. Knjižnica sadrži skup funkcija čije korištenje olakšava podešavanje LCD-a te slanje i ispis varijabli na njemu.

```
#include <LiquideCrystal.h>
```

Ukratko ćemo pojasniti neke od funkcija iz knjižnice LiqideCrystal.

```
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;

lcd(2, 3, 4, 5, 6, 7);
```

Argumenti funkcije *lcd* definiraju kako će pinovi LCD-a biti spojeni na razvojnu ploču Arduino. LCD radi u tzv. 4-bitnom modu i pinovi s indeksom "d" služe za slanje komandi i podataka ka zaslonu. Pinovi *RS* i *EN* služe za kontrolu zaslona. Postoji još jedan pin s oznakom *R/W* koji se spaja na GND i u tom modu LCD može samo primati podatke. Odnosno LCD je u tzv. "Read" modu.

```
lcd.begin(16,2);
```

Funkcija *lcd.begin* služi za inicijalizaciju LCD-a, a proslijeđeni argumenti definiraju dimenzije zaslona koji se koristi.

```
lcd.setCursor(3,1);
```

Funkcija *lcd.setCursor* postavlja kursor na poziciju proslijeđenih argumenata. Na primjeru iznad kursor na LCD-u bit će podešen na 3. red i 1. stupac.

lcd.print ("Zdravo svijete");

Funkcija *lcd.print* prosljeđuje podatke koji bi se trebali ispisati na LCD-u. Ako podatci dolaze u formatu s dvostrukim apostrofom "ja sam string", riječ je o konstantnom stringu. Moguće je poslati na ispis i varijable kao u primjeru za mjerenje temperature navedenom u nastavku.

```
// Primjer aplikacije
// include LCD biblioteke
#include <LiquidCrystal.h>
// LM35 out pin je spojen na A0 pin Arduina
#define  LM35_pin  0
// deklaracija globalne varijable
float temp;

// kreiranje LCD objekta
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
    // Inicijalizacija LCD i podešavanje kursora
    lcd.begin(16, 2);
    lcd.setCursor(2, 0);
    // Print fixnog texta
    lcd.print("Temperatura:");
    analogReference(INTERNAL);//za referentni napon koristimo interni ref.
napon na ADC-u
}

void loop() {
   // Analognu vrijednost napona pretvara u °C ( 9.3 = 1023/(1.1*100))
   temp = analogRead(LM35_pin) / 9.3;
   lcd.setCursor(3, 1);
   lcd.print(temp);
   delay(1000);  //1s za ponovno učitavanje
}
```

Ovaj dio koda ne trebate promatrati kao završen projekt, nego pokušati ga nadograditi već predloženim idejama ili realizirati neke svoje.

ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- definira svojstva senzora temperature LM35 - opisuje ulogu 16×2 LCD zaslona
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje 16×2 LCD zaslon na Arduino prema shemi spoja - povezuje senzor temperature LM35 na Arduino prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za digitalni toplomjer sa senzorom LM35 i prikazom temperature na LCD zaslonu	- primjenjuje znanje o upotrebi funkcija iz programiranja za realizaciju zadataka - koristiti matematičke formule koje su specifične za senzor - kreira program za dati zadatak uz adekvatne smjernice nastavnika/-ice - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka	- pronalazi odgovarajuće knjižnice za realizaciju datog projektnog zadatka - instalira odgovarajuće knjižnice - poziva adekvatnu knjižnicu unutar programskog koda
- verificira i izvršava program	- analizira programski kod - izvršava otklanjanje pogreške po potrebi i prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - mjeri temperaturu s pomoću analognog senzora LM35 - učitava temperature sa 16×2 LCD zaslona - analizira projektni zadatak - dopunjuje projektni zadatak svojim idejama

Prostor za vježbu:

```
 void setup() {
}
void loop() {
}
```

Projekt 2.
HC-SR04

Nadam se da svi znamo kako šišmiš (slijepi miš) leti u skučenome prostoru, a ne udara u prepreke ili kako podmornica može detektirati drugu podmornicu. Odgovor je sonar (prema engl. *Sound Navigation and Ranging*: navigacija i određivanje udaljenosti s pomoću zvuka). Pojednostavljeno u oba slučaja izvor emitira zvuk i mjeri vrijeme koje je prošlo dok se taj zvuk vrati. Naravno, i sustav detekcije kod šišmiša kao i kod podmornice su složeni, ali to nas neće spriječiti da pokušamo napraviti nešto u najmanju ruku slično.

VCC: Napon napajanja (+5V)
Trig: Trigger input pin
Echo: Echo output pin
GND: Ground (masa) (0V)

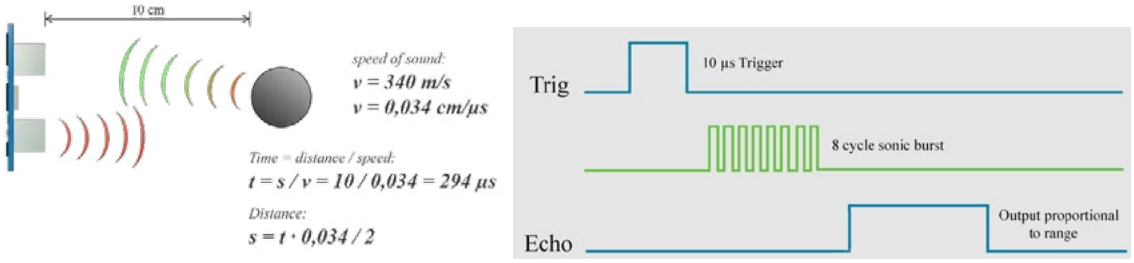
Slika pr. 2.3. *Izgled i pinout HC-SR04*

Modul **HC-SR04** Ultrasonic jednostavan je senzor koji emitira ultrazvučni val pri 40 000 Hz. Ovaj val putuje zrakom te ako na njegovome putu postoji neki objekt ili prepreka, odbit će se natrag do modula. Uzimajući u obzir vrijeme putovanja i brzinu zvuka možemo izračunati udaljenost.

HC-SR04 posjeduje 4 pina: GND, V_{CC}, Trig i Echo. Pinovi GND i V_{CC} trebaju se povezati na Arduino pinove GND i 5V Arduino ploče, respektivno. Pinovi Trig i Echo povezuju se na bilo koji Digitalni I/O pin ploče Arduino.

Kako bismo generirali ultrazvučni val i počeli s proračunom udaljenosti, potrebno je **Trig** pin podići u stanje logičke jedinice na razdoblje od 10 us u vidu impulsa. Ako se ispred senzora nalazi objekt, **Echo** pin će otići u stanje logičke jedinice i u zavisnosti od udaljenosti predmeta trajanje stanja logičke jedinice će varirati. Echo pin daje kao izlaz vrijeme putovanja zvuka u mikrosekundama (µs).

Naprimjer, ako je objekt 10 cm udaljen od senzora, znajući da je brzina zvuka 340 m/s (ili 0.034 cm/µs) zvučni val će prelaziti put oko 294 µs. Ovo je vrijeme za koje val prijeđe put od izvora (senzora) do prepreke i vrati se do izvora (pogledati sliku ispod).



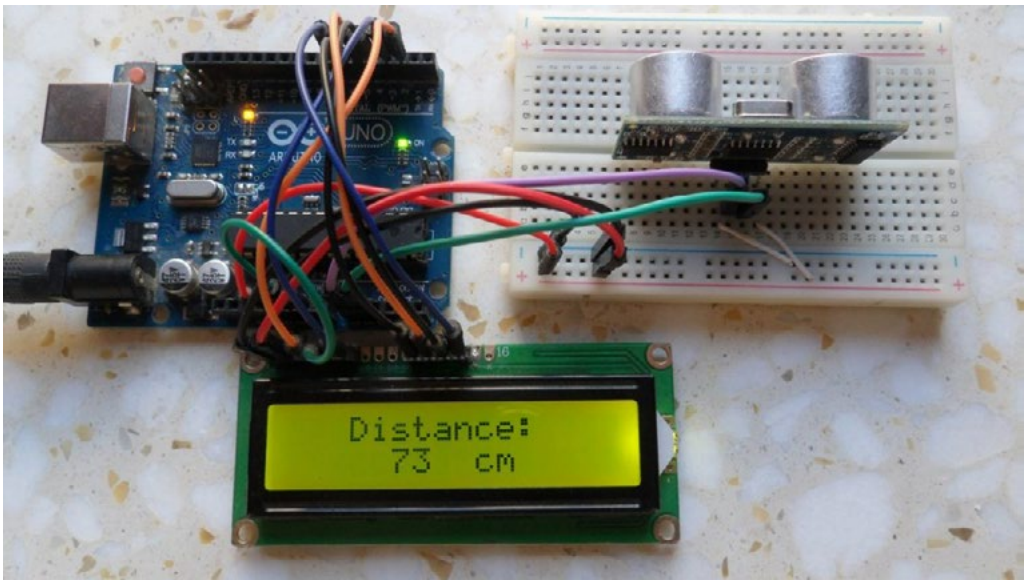
Izvor: https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

Tako da, ako želimo dobiti točnu udaljenost u centimetrima (cm), moramo pomnožiti izmjereno vrijeme na Echo pinu s brzinom (vrijednost 0.034) i podijeliti sve s 2. Odnosno, možemo to izraziti i kao: udaljenost = vrijeme / (2/0.034) = vrijeme / 58.8 ≈ vrijeme / 58.

Kao i u prethodnome projektu, i ovaj put možemo koristiti formulu za proračun koja je data od proizvođača senzora, a s pomoću koje se vrijeme trajanja impulsa pretvori u udaljenost:

Udaljenost u cm = (Trajanje HIGH stanja Echo pina)/58

Ako se ispred senzora ne nalazi prepreka, na Echo pinu će se generirati impuls trajanja 38 ms, u suprotnom imat ćemo impulse trajanja (125 µs – 25 ms) u zavisnosti od udaljenosti.



Slika pr. 2.1. *Prikaz udaljenosti na LCD zaslonu*

Možda vam trenutačno pada na pamet da mjerite udaljenost nekog predmeta na stolu na kojem se nalazi senzor, ali evo nekoliko ideja koje možete iskoristiti za unapređenje ovog početnog projekta:

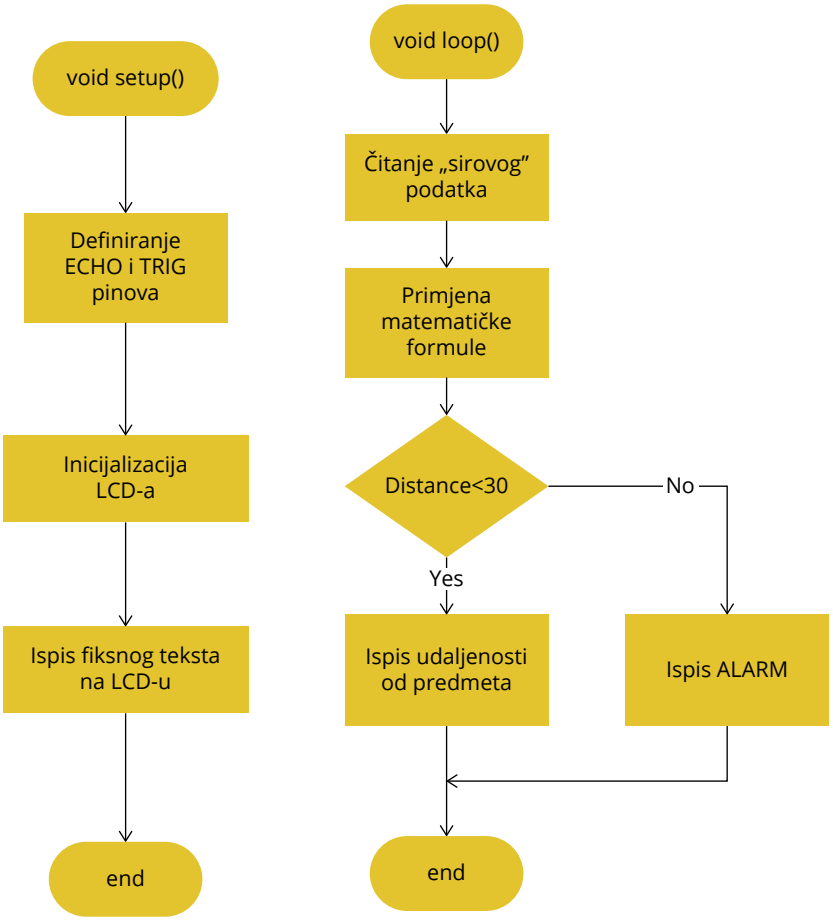
1. Mjerenje visine vaših školskih kolega/-ica
2. Koristiti ga za mjerenje skoka udalj na satu sporta
3. Senzor za parking za automobil vaših roditelja
4. Detekcija propisno razmaknutih predmeta u prostoru

Kada proširite svoje znanje, onda neće biti teško na platformu Arduino spojiti Bluetooth modul ili, što da ne, WiFi modul te prikupljene podatke slati na neki mobilni uređaj ili mrežu. Ali više o ovome u nekom od narednih projekata.

Potrebni elementi za realizaciju projekta navedeni su ispod:

1. ARDUINO UNO
2. HC-SR04 ultrasonic sensor module
3. 16×2 LCD zaslon
4. 10KΩ potenciometar
5. 330 ohm otpornik
6. Pločica Matador
7. Kablići

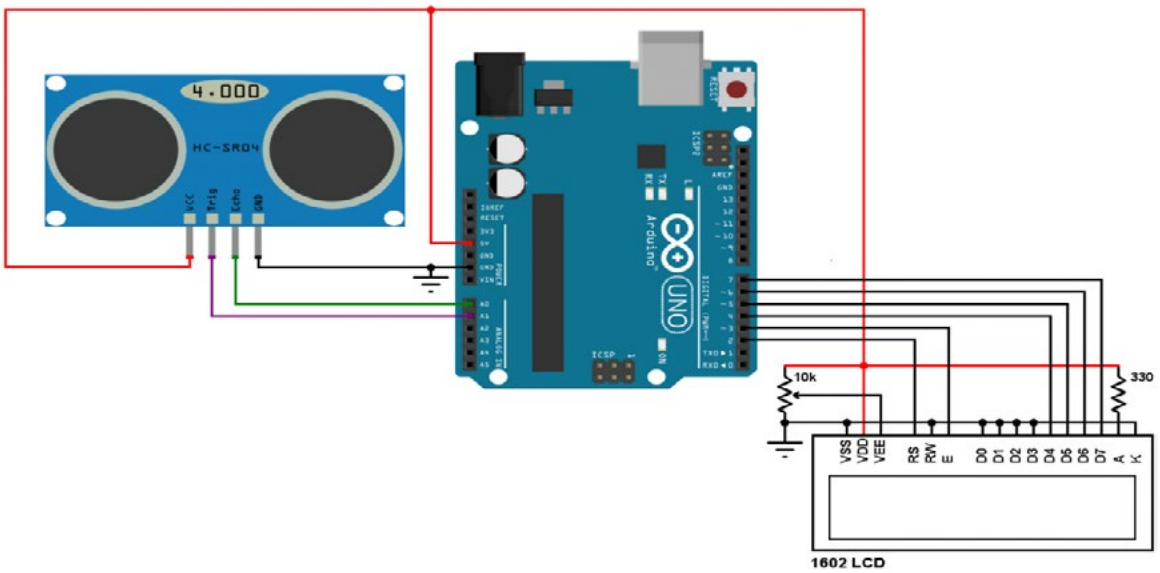
Bilo bi sjajno kada biste pokušali za svaki projekt koji zamislite prvo nacrtati grubu algoritamsku strukturu projekta, u ovom projektu ćemo mi to uraditi za vas, ali nadamo se da ćete steći naviku da prije nego počnete pisati kod, prvo taj kod probate vizualizirati i „nacrtati“ ga.



Slika pr. 2.2. Pojednostavljena algoritamska struktura aplikacije

Za dobivanje duljine trajanja impulsa koristiti funkciju ***pulsIn()***, to je funkcija koja ima dva argumenta: ***pin*** na kojem se u ovom konkretnom slučaju spaja E***cho*** pin senzora, a drugim argumentom definiramo očekujemo li signal HIGH ili LOW na pinu. Na primjeru ispod možemo uočiti da je pin 7 pin na kojem se očekuje promjena stanja. Konkretno nas zanima i vrijeme koje je potrebno da se pojavi HIGH signal.

```
int pin = 7;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseIn(pin, HIGH);
  Serial.println(duration);
}
```



Slika pr. 2.4. *Shema spoja*

Kod Arduino:

```
// Mjerenje udaljenosti s Arduino UNO i  HC-SR04
// ultrasoničnim senzorom

// uključujemo LCD biblioteku
#include <LiquidCrystal.h>
#define  trigger_pin  15   / HC-SR04 trigger pin je povezan na Arduino A1
#define  echo_pin     14   // HC-SR04 echo pin je povezan na Arduino A0

// deklaracija globalnih varijabli
unsigned long duration;
unsigned int  distance;

// LCD objekat
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  pinMode(trigger_pin, OUTPUT);
  pinMode(echo_pin, INPUT);
  digitalWrite(trigger_pin, LOW);
  // inicijalizacija LCD-a
  lcd.begin(16, 2);
  lcd.setCursor(3, 0);
  lcd.print(„Distance:“);
   }
void loop() {

    // Pošalji puls u trajanju od 10 us na HC-SR04 Trig pin
    digitalWrite(trigger_pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigger_pin, LOW);
```

```
    // Pročitaj puls koji stiže na Echo pin
     duration = pulseIn(echo_pin, HIGH);              // Pročitaj širinu
pulsa sa Echo pina
   if(duration == 0) {
// Prethodna funkcija vraća vrijednost nula u slučaju da je isteklo vri-
jeme
    lcd.setCursor(2, 1);
      lcd.print(„  Time Out   „);
    }
    else {
      distance = duration / 58;    // Izračunaj udaljenost u  cm
      if(distance > 400) {         // HC-SR04 modul mjeri udaljenosti do
400 cm
       lcd.setCursor(2, 1);
        lcd.print(„Out of Range“);
     }
      else {
        lcd.setCursor(2, 1);
        lcd.print(„       cm    „);
              lcd.setCursor(5, 1);
       lcd.print(distance);
     }
    }
    delay(100);                    //Sačekaj 100 ms između očitanja
 }
}
```

Ovaj dio koda ne trebate promatrati kao završen projekt, nego pokušati ga nadograditi već predloženim idejama ili realizirati neke svoje.

ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- objašnjava način rada senzora za udaljenost HC-SR04 - navodi ulogu pinova na ultrasoničnom senzoru
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje senzor HC-SR04 na Arduino prema shemi spoja - povezuje preostale komponente u projektu prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za ultrasonični senzor	- primjenjuje formulu s pomoću koje se vrijeme trajanja impulsa pretvori u udaljenost - kreira program za dati zadatak uz adekvatne smjernice nastavnika/-ice - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka	- po potrebi instalira odgovarajuće knjižnice - poziva unutar programskog koda adekvatnu knjižnicu
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice po potrebi
- testira projektni zadatak	- učitava kod na platformu Arduino - učitava udaljenost koja se prikazuje na zaslonu - analizira projektni zadatak - dopunjuje projektni zadatak datim prijedlozima ili nekim svojim idejama

Prostor za vježbu

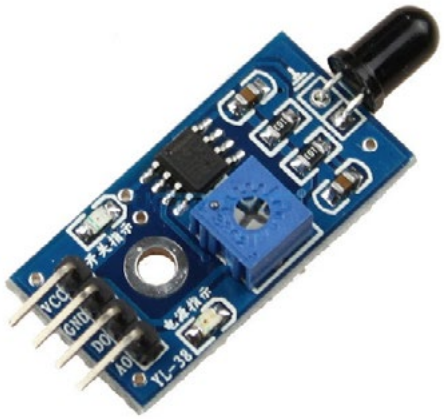
```
 void setup() {

}
void loop() {

}
```

:

Projekt 3.
Flame Sensor

Jedan od vrlo korisnih senzora u našem setu je senzor za detekciju plamena – *Flame Sensor*. Koristeći ovaj senzor u kombinaciji s drugim senzorima i komponentama možete kreirati dosta složene projekte. Svaka ustanova treba imati senzore za detekciju plamena, a mi to možemo kreirati koristeći Arduino.

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



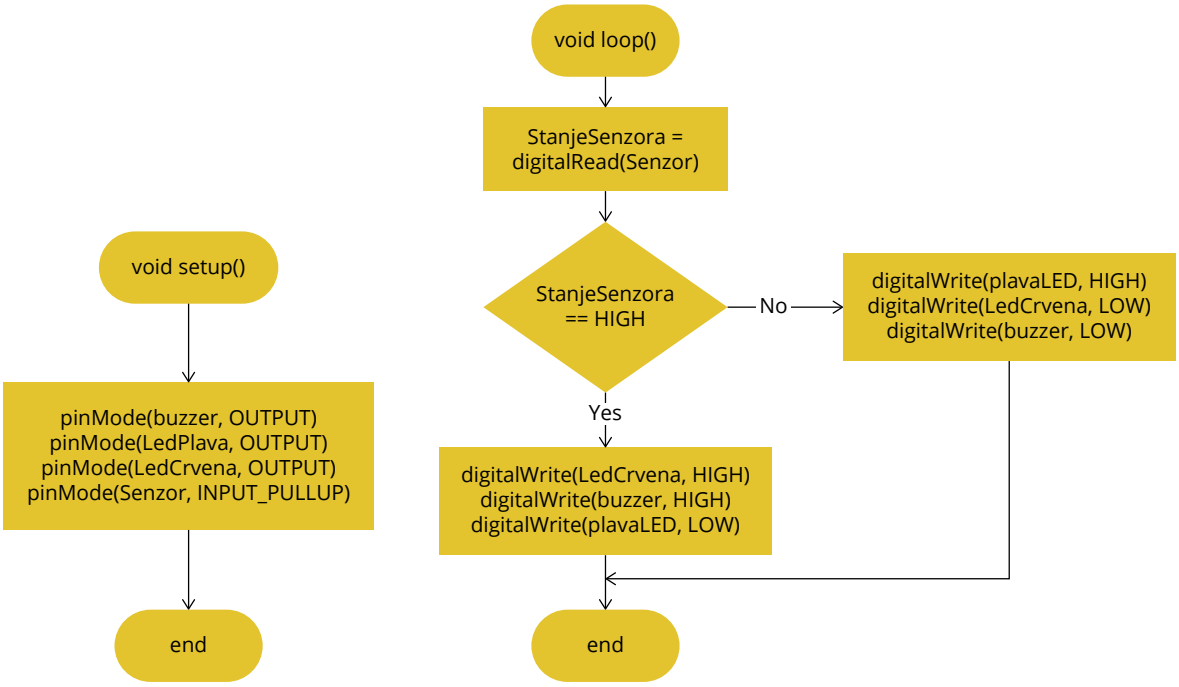
Slika pr. 3.1. Prikaz Flame Sensora

Ova vrsta senzora koristi se za detekciju požara kratkoga dometa i može se koristiti za praćenje projekata ili kao sigurnosna mjera za isključivanje/uključivanje uređaja. Senzor plamena vrlo je osjetljiv na infracrvenu (IR) valnu duljinu svjetlosti (760 nm do 1100 nm). I ovaj senzor ima analogni i digitalni izlaz. Ako u projektu koristimo analogni izlaz, onda V_{CC} vezujemo za pin od 5V, a ako koristimo digitalni izlaz, onda V_{CC} vezujemo za pin od 3.3V.

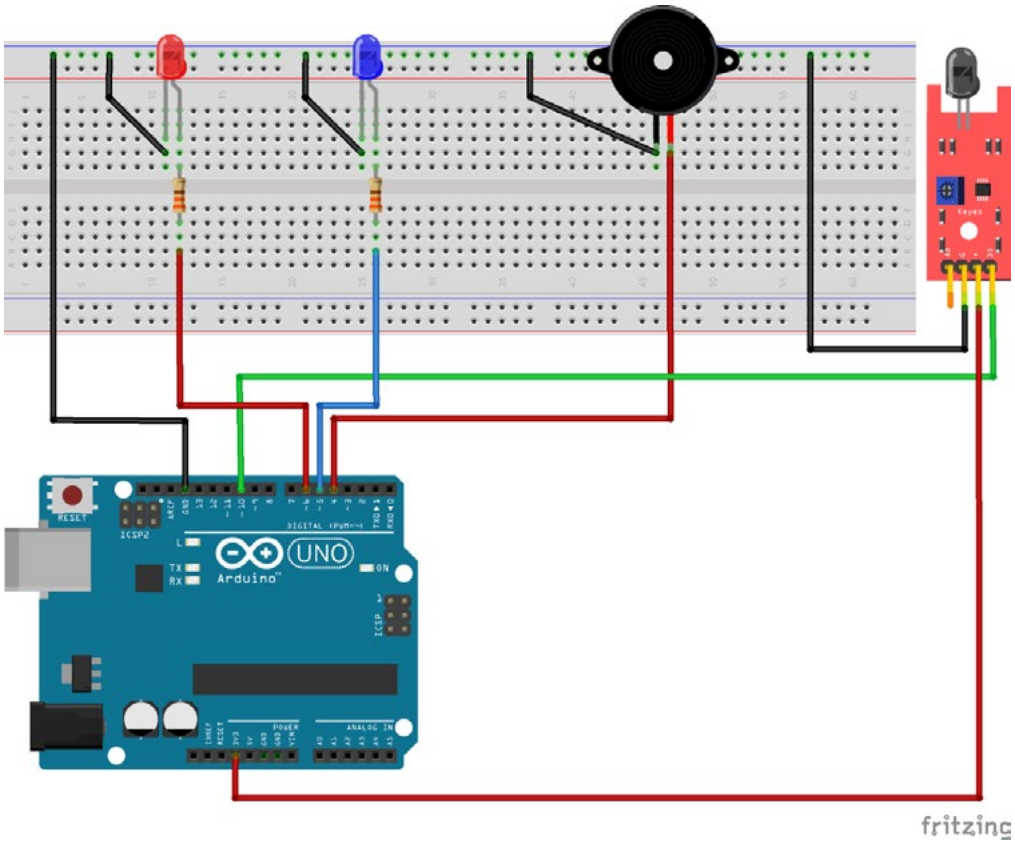
Potrebni elementi za realizaciju projekta su:

1. Pločica Arduino UNO i USB
2. Pločica Matador
3. Senzor za detekciju plamena
4. 1x zelena LED
5. 1x crvena LED
6. 2x otpornik 220 Ω

Kao i u prethodnim primjerima nacrtajmo grubu algoritamsku strukturu projekta.



Slika pr. 2.2. *Pojednostavljena algoritamska struktura aplikacije*



Slika pr. 2.4. *Shema spoja*

Kod Arduino:

```
int LedCrvena = 6;
int LedPlava=5;
int Senzor = 10;
int buzzer=4;


int StanjeSenzora;

void setup() {
  pinMode(buzzer, OUTPUT);
  pinMode(LedPlava,OUTPUT);
   pinMode(LedCrvena, OUTPUT);       //postavi izvod LedCrvena (6) kao
izlazni
  pinMode(Senzor, INPUT_PULLUP);  //postavi izvod Senzor (10) kao ulazni
}

void loop() {
  StanjeSenzora = digitalRead(Senzor); //očitaj stanje izvoda i pohrani
u StanjeSenzora
  if (StanjeSenzora == HIGH) {
    digitalWrite(LedCrvena, HIGH);
    digitalWrite(buzzer, HIGH);
    digitalWrite(LedPlava, LOW);
  } else {                            //inače (digitalna vrijednost je 0)
    digitalWrite(LedPlava, HIGH);
    digitalWrite(LedCrvena, LOW);
    digitalWrite(buzzer, LOW);
  }
}
```

Ovaj dio koda ne trebate promatrati kao završen projekt, nego ga pokušati nadograditi koristeći druge komponente kako biste razvili vlastite ideje.

NAPOMENA:

Budite oprezni prilikom testiranja ove aplikacije. Neophodno je osigurati prostor i ukloniti zapaljive materijale kako biste testirali senzor plamena.

ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- definira svojstva senzora za detekciju plamena - opisuje ulogu Bazera (engl. *buzzer*) - opisuje primjenu LE dioda u aplikaciji - objašnjava razliku između analognih i digitalnih senzora
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje senzor za detekciju plamena na Arduino prema shemi spoja - povezuje ostale komponente aplikacije na Arduino prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za Flame Sensor	- kreira program za dati zadatak primjenjujuću stečeno znanje iz razgranatih programskih struktura - rekonstruira programski kod prema vlastitim idejama
- verificira i izvršava program	- analizira programski kod - izvršava otklanjanje pogreške po potrebi uz smjernice nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - testira aplikaciju koristeći upaljač ili žigicu vodeći računa o sigurnosti - analizira projektni zadatak - dopunjuje projektni zadatak svojim idejama

‘

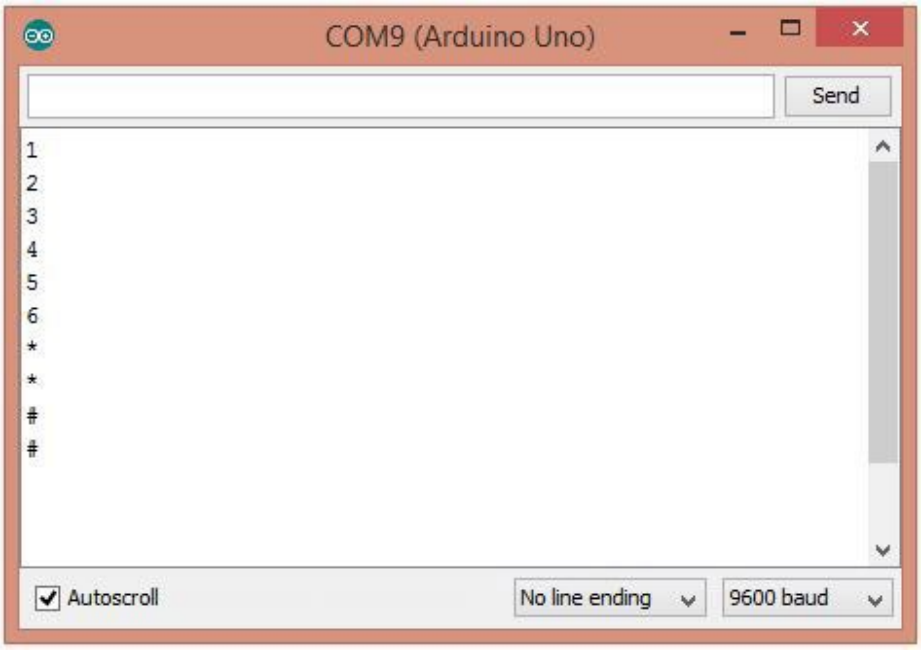
Prostor za vježbu:

```
.....

.....

.....

 void setup() {

.....

.....

.....

}

void loop() {

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

}
```

Projekt 4.
Arduino Keyboard

Ovo je još jedan zanimljiv *gadget* koji možemo koristiti u svojim projektima. Usudio bih se reći sigurnosni uređaj. Koristit ćemo membransku tipkovnicu.

Membranska tipkovnica diže svaki projekt Arduino na jednu novu razinu. Recimo, omogućava nam da u projekt ubacimo unos šifre, pina, ID korisnika aplikacije, ili, što da ne, unos broja telefona koji želimo pozvati. Membranska tipkovnica sastoji se od 12 ili u nekim izvedbama 16 tipki poredanih u stupce i redove, ona predstavlja ulazni uređaj u mikrokontroler i njegova unutarnja struktura predstavlja tastere koji su spojeni u formatu matrice npr. 4x4. Ovaj put nećemo koristiti LCD zaslon za prikaz unosa s tipkovnice, već ćemo ispis unosa raditi kroz ugrađenu aplikaciju (engl. *built-in)* Serial Monitor.



Slika pr. 4.1. *Prikaz udaljenosti na LCD zaslonu*

Vjerujem da i vi volite špijunske ili znanstvenofantastične filmove i vjerojatno vam neće nedostajati ideja za upotrebu ovog jednostavnog, ali efektnog uređaja u nekom od projekta, ali za svaki slučaj evo nekoliko primjera kako bi se on mogao iskoristiti:

1. PinLock brava – brava koja se zaključava/otključava unošenjem neke šifre (PIN-a)
2. Arduino GSM dialer – u kombinaciji s GSM modulom kreirajte telefon Arduino
3. Arduino keyboard klavijatura

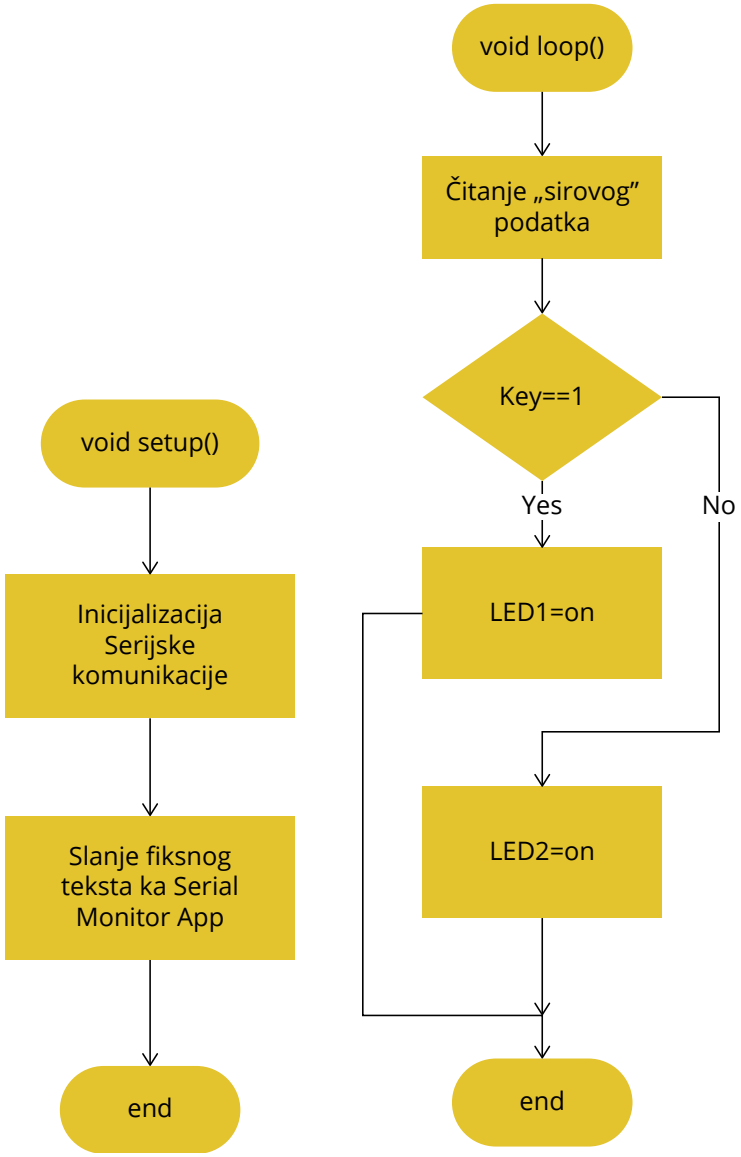
Moguće je kod projektiranja neke vaše aplikacije ostaviti prostora i slobodne pinove na kontroleru te kreirati plug-in konektor koji će se moći s napisanom aplikacijom koristiti za eventualno „plaćeno otključavanje“ opcija na aplikaciji.

Potrebni elementi za realizaciju projekta navedeni su ispod:

1. ARDUINO UNO
2. Tipkovnica

Nakon upoznavanja s hardverom i opcijom *Serial monitor* projektni zadatak je da kreirate aplikaciju koja će ako je pritisnut taster 1, aktivirati LED spojenu na PIN2, a ako je aktiviran neki drugi taster, aktivirati LED spojenu na PIN3.

Nacrtajmo prvo grubu algoritamsku strukturu projekta, kao što smo radili u prethodnim projektima. Ovoga puta ćemo iskoristiti tipkovnicu za malo ponavljanje izjave IF...THEN...ELSE.

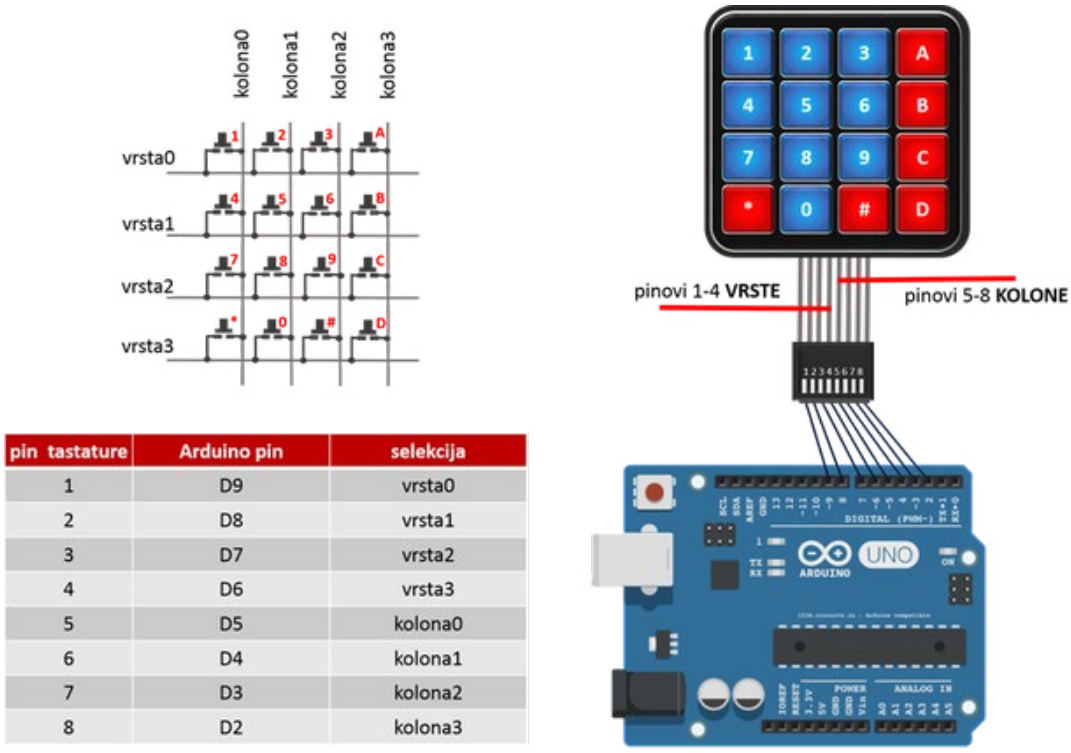


Slika pr. 4.2. *Pojednostavljena algoritamska struktura aplikacije*

Za ovaj konkretni uređaj nije nam potrebna matematička formula, ali se već u samom algoritmu aplikacije *sample* postavlja pitanje mogućnosti „pamćenja“ složenijih informacija. Moguće je kreirati niz (*array*) tip podatka:

```
int enteredPIN[4];

int fixedPIN[] = {1, 2, 3, 4};
 enteredPIN[0]== fixedPIN[0];
     enteredPIN[0]== 1;
     enteredPIN[1]== 2;
```

Elementi niza bi se potom punili tako da se provjerava unos s tipkovnice s elementima „fiksnog“ niza. Naravno, ovo je samo jedan od načina kako bismo mogli razviti malo složeniju aplikaciju.



pin tastature	Arduino pin	selekcija
1	D9	vrsta0
2	D8	vrsta1
3	D7	vrsta2
4	D6	vrsta3
5	D5	kolona0
6	D4	kolona1
7	D3	kolona2
8	D2	kolona3

Slika pr. 4.3. *Izgled i pinout keyboarda*

Kod Arduino:

```
#include "Keypad.h"
const byte ROWS = 4; // broj redova
const byte COLS = 3; // broj kolona
char keys[ROWS][COLS] = {
{'1','2','3'},
{'4','5','6'},
{'7','8','9'},
{'#','0','*'}
};
byte rowPins[ROWS] = {8, 7, 6, 5}; // izlazni pinovi tastature po redo-
vima R1 = D8, R2 = D7, R3 = D6, R4 = D5
byte colPins[COLS] = {4, 3, 2};    // izlazni pinovi tastature po kolo-
nama C1 = D4, C2 = D3, C3 = D2
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  char key = keypad.getKey();
  if (key != NO_KEY)
    Serial.println(key);
}
```

Ovaj dio koda ne trebate promatrati kao završen projekt, nego ga pokušati nadograditi već predloženim idejama ili realizirati neke svoje ideje.

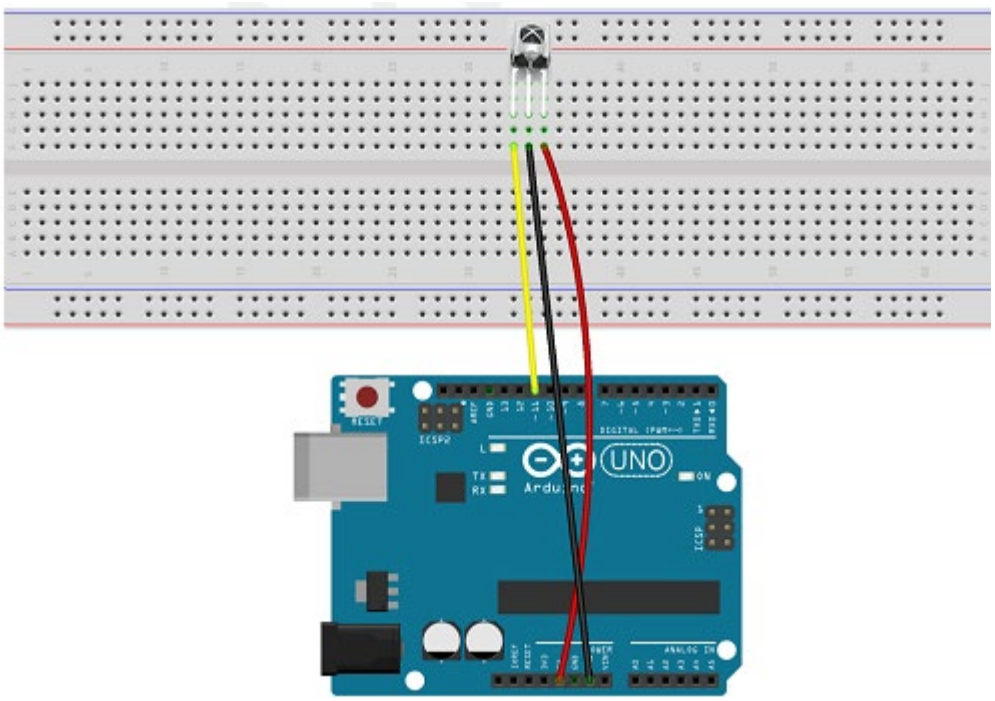
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- objašnjava ulogu membranske tipkovnice - Koristi se Serial monitorom
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje membransku tipkovnicu na Arduino prema shemi spoja - povezuje preostale komponente u Projektu prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za primjenu membranske tipkovnice	- primjenjuje stečeno znanje iz nizova i matrica - kreira programski kod za dati zadatak uz adekvatne smjernice nastavnika/-ice - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka	- po potrebi instalira odgovarajuće knjižnice - poziva adekvatnu knjižnicu unutar programskog koda
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - koristi se Serial monitorom za testiranje projektnog zadatka - analizira projektni zadatak - dopunjuje projektni zadatak datim prijedlozima ili nekim svojim idejama

Prostor za vježbu:

```
.....

.....

.....

 void setup() {

.....

.....

.....

}

void loop() {

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

}
```

Projekt 5.
IR senzor

Ovaj nam je projekt posebno zanimljiv, jer smo od malih nogu okruženi i fascinirani daljinskim upravljačima. Ovo će ujedno biti ispunjenje sna svake programerke i programera: konačno ulazimo u svijet daljinske kontrole. Realizacijom ovog projekta otvaramo bezbroj mogućnosti za kontrolu uređaja na udaljenosti od četiri do devet metara. Da, to i nije baš puno, ali bitno je napraviti prvi korak. Mora se naglasiti da udaljenost s koje možemo kontrolirati neku aplikaciju koristeći ovaj senzor uveliko zavisi od okruženja u kojem se koristi, kao i eventualnih prepreka prilikom njegova korištenja.



Slika pr. 5.1. *Način spajanja IR senzora*

Za početak prvo ćemo pokušati kontrolirati LE diode, ali evo nekoliko ideja koje možete iskoristiti za nadogradnju ovog početnog projekta.

1. Kontrola sobnog ventilatora
2. Aktivacija sobne rasvjete
3. Izrada RGB kontrolera

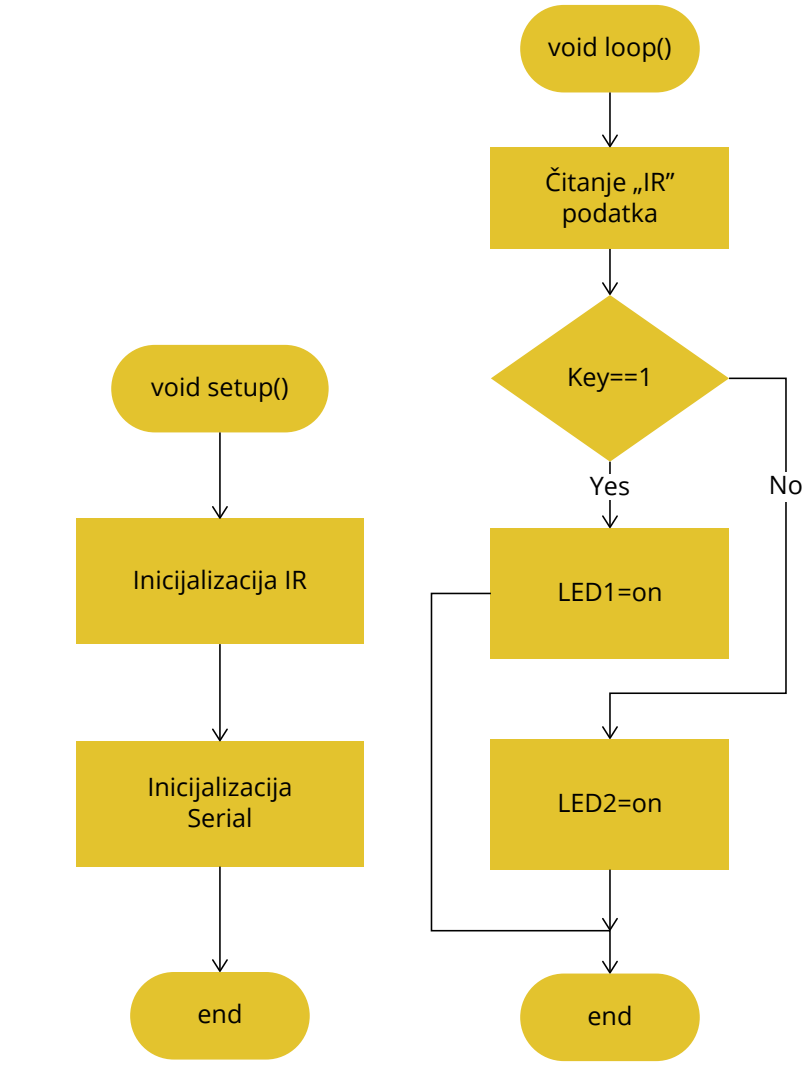
Svaki od spomenutih prijedloga može se nadograditi uvodeći u projekt nove komponente ili komponente koje su već korištene u ovom praktikumu. Jedna od ideja koja bi vam mogla pomoći je izrada IR debugera, za koji možete koristiti LCD zaslon. Naime, moguće je iskoristiti LCD zaslon za ispis primljenoga podatka s IR predajnika. Na taj način vaš LCD postaje *live debuger* i olakšat će vam pisanje aplikacije.

Kako budete proširivali svoje znanje, možda s pomoću ovog senzora napravite opciju za odabir aplikacije spram primljenoga podatka s daljinskog upravljača. Možete napisati tri različite aplikacije na vašem Arduinu. Odabir koja aplikacija će se koristiti može se uraditi na osnovi podatka dobivenog s IR senzora. Na ovaj način naš Arduino barem prividno postaje multifunkcionalni uređaj.

Potrebni elementi za vježbu:

1. ARDUINO UNO
2. IR Receiver HX1838
3. Matador
4. Daljinski upravljač
5. Kablići

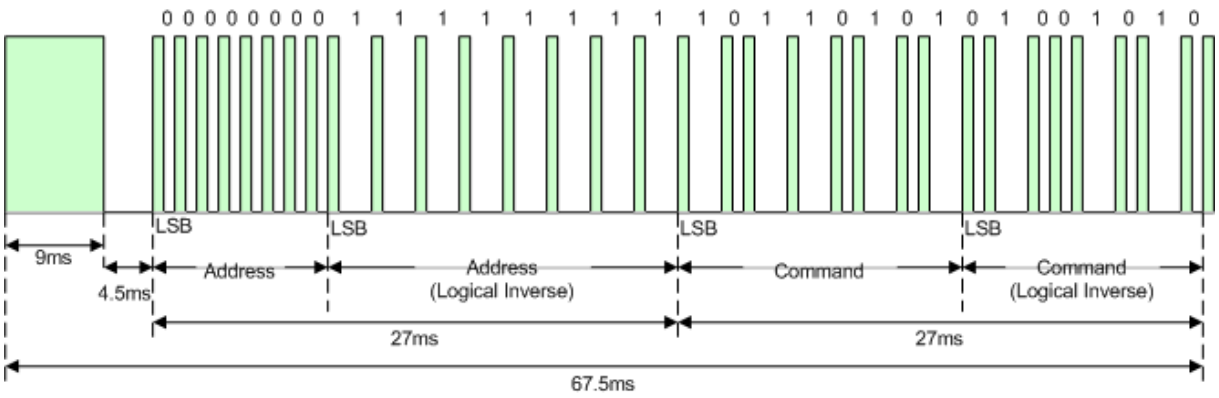
Nastavit ćemo s praksom crtanja grube algoritamske strukture projekta i nadamo se da ćete steći naviku da prije nego počnete pisati kod, prvo taj kod probate vizualizirati i „nacrtati“ ga. U ovom slučaju ne mora značiti da će algoritamska struktura odgovarati stvarnoj aplikaciji, ali sugerira vam se da je najjednostavnije krenuti od usporedbe zaprimljenog podatka s nekom konstantom te donijea



Slika pr. 5.2 *Pojednostavljena algoritamska struktura aplikacije*

Iako ćemo u ovome projektu koristiti gotovu knjižnicu, bilo bi korisno naučiti ili barem vidjeti kako to IR predajnik, u našem slučaju IR daljinski upravljač, generira poruku koju naš IR senzor mora „dekodirati“. Svaki put kada se na daljinskom upravljaču pritisne neka tipka, pošalje se poruka sljedećeg formata:

- Signal logičke 1 trajanja 9 ms
- Signal logičke 0 trajanja 4.5 ms
- 8-bitna adresa prijemnika
- 8-bitna invertirana adresa prijemnika
- 8-bitna komanda
- Invertirana 8-bitna komanda
- Puls trajanja 562,5 µs koji signalizira kraj poruke

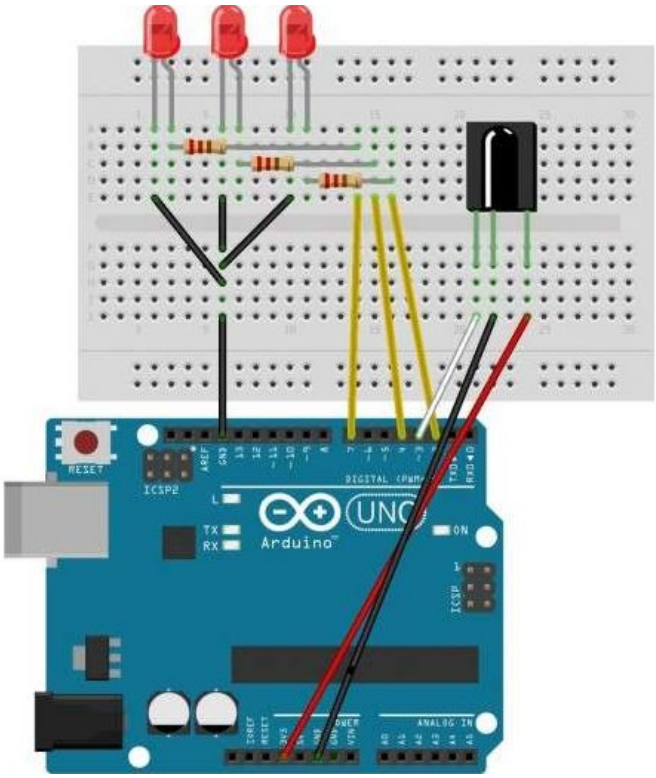


Slika pr. 4.3. Protokol IR NEC

Korištenjem knjižnice bit ćemo zakinuti za potpuno razumijevanje protokola i procesa dekodiranja poruke, ali na ovaj ćemo način puno brže doći do gotove aplikacije. Dakle da ponovimo, IR senzor HX1838 prima infracrveni signal koji generira daljinski upravljač, dekodira signal koji ćemo iskoristiti u svojoj aplikaciji za odlučivanje i ima tri pina (input pin, V_{CC} i GND).

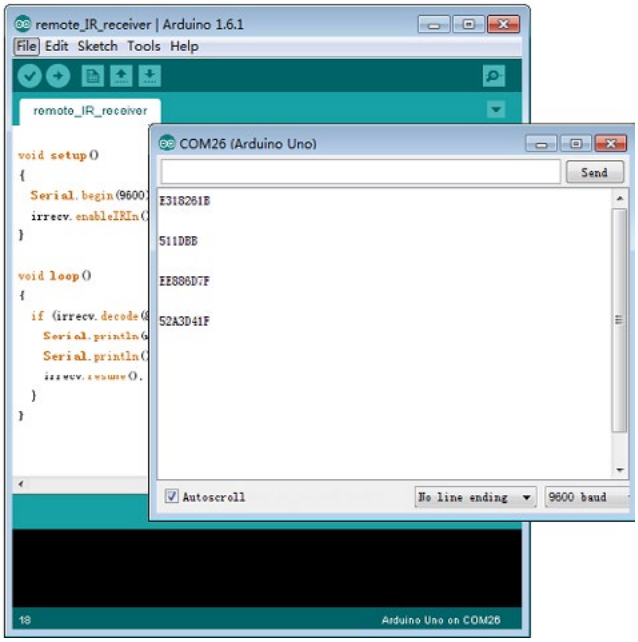
Slika pr5.4. Pinout HX1838 i izgled daljinskog upravljača

U ovom projektu potrebno je prvo programirati Arduino da prima infracrveni signal i da ga pošalje na Serial Monitor. Za ovo nam je potrebna knjižnica Arduino-IRremote-master library.



Slika pr. 5.5. *Shema spoja*

U testnoj aplikaciji za softverski *debugging* koristit ćemo aplikaciju Serial Monitor. Za preporuku je napraviti tablični ispis svih dekodiranih naredbi iz Serial monitora. Na taj ćete način sebi olakšati planiranje aplikacije, a otvara vam se brži put za definiranje konstanti u nekoj od vaših narednih aplikacija.



Slika pr. 5.6. *Izgled dekodirane komande*

Kod Arduino:

```
#include <IRremote.h>
int RECV_PIN = 11;//deklaracija IR prijemnika na pinu 11
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600); //Otvori serijski port
  irrecv.enableIRIn(); // Inicijalizacija IR prijemnika
}
void loop()
{
 if (irrecv.decode(&results)) {
 Serial.println(results.value, HEX);//Ispis u heksadecimalnom prijemnom
kodu
  Serial.println(); Radi pregleda ispisujemo jednu praznu liniju
  irrecv.resume(); //Prijem naredne vrijednosti
  }
  }
```

Ni ovaj dio koda ne trebate promatrati kao završen projekt, nego ga pokušati nadograditi već predloženim idejama ili realizirati neke svoje.

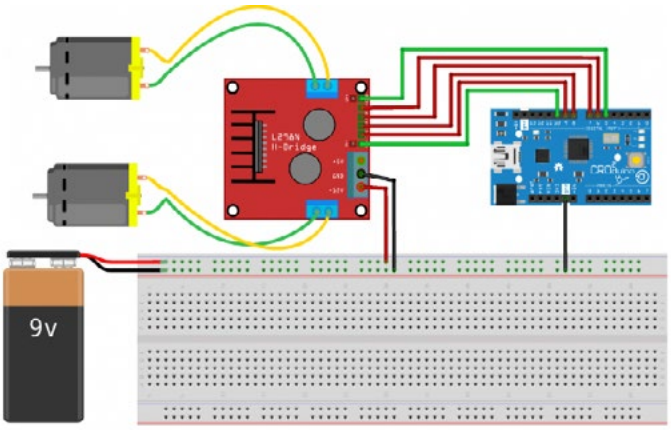
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- objašnjava ulogu senzora IR Receiver HX1838 - objašnjava primjenu IR predajnika i način odašiljanja poruka s daljinskog upravljača - opisuje vezu između daljinskog upravljača i IR senzora - opisuje primjenu LE diode u Projektu - Koristi se Serial Monitorom za učitavanje rezultata
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje senzor IR Receiver HX1838 prema shemi spoja - povezuje preostale komponente u projektu prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za daljinsko upravljanje	- programira Arduino da prima infracrveni signal i da ga šalje na Serial Monitor uz adekvatne smjernice nastavnika/-ice - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka	- koristi gotovu knjižnicu za IR predajnik – Arduino-IRremote-master library - poziva adekvatnu knjižnicu unutar programskoga koda
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - koristi se Serial Monitorom za testiranje projektnog zadatka - analizira projektni zadatak - primjenjuje senzor IR Receiver HX1838 za upravljanje na daljinu u drugim projektnim idejama - razvija vlastite ideje za primjenu senzora IR Receiver HX1838

Prostor za vježbu:

```
 void setup() {
}
void loop() {
}
```

Projekt 6.
Drajver L298

Većina robot ili CNC entuzijasta krene u pustolovinu s mikrokontrolerskim sustavima upravo s ovim drajverom, naime zbog ograničenja količine struje koju mikrokontroler može dati na svojim izlazima. Skoro je pa nemoguće izravno na izlaz mikrokontrolera spojiti bilo kakav ozbiljniji motor. Stoga ako nije riječ o hobi RC servo motoru, za pokretanje motora veće snage potreban nam je drajver. Drajver motora pored toga što osigurava dovoljnu količinu struje za pokretanje otvara nam i opciju za softversku kontrolu broja obrtaja i smjera vrtnje. Drajver motora zahtijeva i stabilno eksterno napajanje za motor, na taj način štitimo i sam mikrokontroler. Kako se budete bavili ovom problematikom, naići ćete na različite izvedbe drajvera s različitim principom djelovanja, a u ovom projektu koristit ćemo jedan od najjednostavnijih, ali u isto vrijeme vrlo pouzdani drajver L298N. I u ovom projektu koristit ćemo ga za kontrolu i upravljanje istosmjernih (DC) motora, uz napomenu da istim drajverom možemo kontrolirati i motore *stepper*. O njima će više govora biti u nekom od narednih projekata.



Slika pr. 6.1 *Shema spajanja drajvera L298 i DC motora*

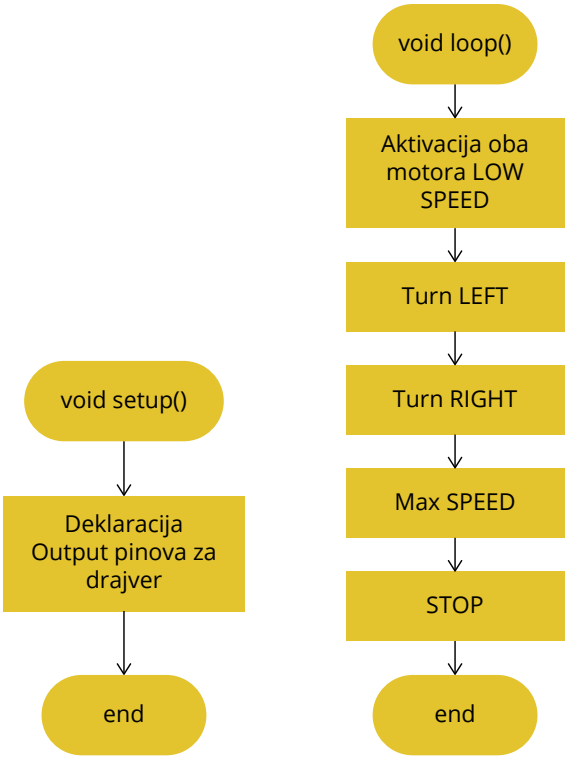
Prema postavci sa slike iznad odmah se nameće ideja o autonomnom vozilu, još ako tomu dodate i HC-SR04 i neko lijepo dizajnirano kućište, mislim da bi se Elon Musk morao malo zabrinuti. Ovo ozbiljno shvatite jer su svi veliki ljudi današnjice krenuli od malih projekata.

Razmislite o projektnim idejama u kojima se koriste drajver L298 i u kojima se koristi DC motor. Malo istražujte!

Potrebni elementi za vježbu:

- ARDUINO UNO
- Drajver L298
- Dva istosmjerna motora
- Pločica Matador
- Kablići

Prvo ćemo nacrtati grubu algoritamsku strukturu projekta, kao što smo radili i s prethodnim primjerima. Nadamo se da ćete steći naviku da prije nego počnete pisati kod, prvo taj kod probate vizualizirati i „nacrtati“ ga.

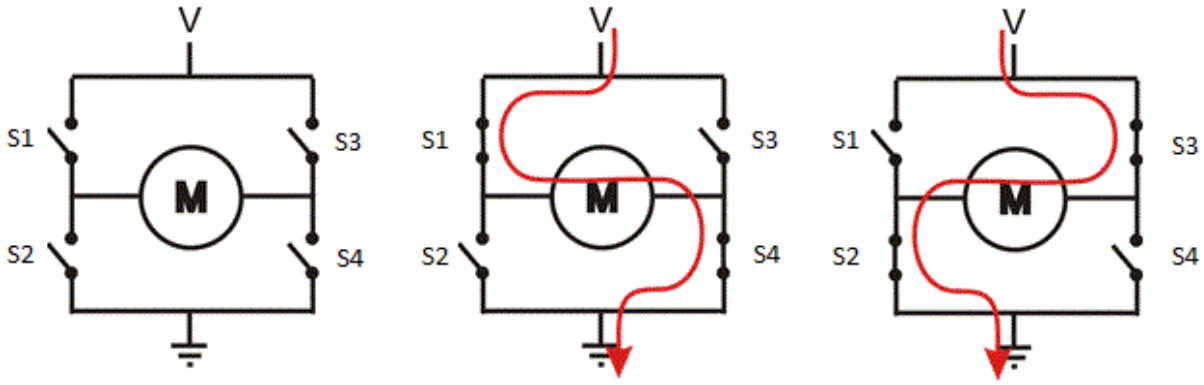


Slika pr. 6.2. *Pojednostavljena algoritamska struktura aplikacije*

OUT1 – +/- Izlaz za motor A **OUT2** – +/- Izlaz za motor A **OUT3** – +/- Izlaz za motor B **OUT4** – +/- Izlaz za motor B **CON5** – Služi za uključivanje 5V regulatora napona **EnA** – Služi za kontrolu PWM-a za motor A **EnB** – Služi za kontrolu PWM-a za motor B	**In1** – Služi za kontrolu smjera okretaja motora A **In2** – Služi za kontrolu smjera okretaja motora A **In3** – Služi za kontrolu smjera okretaja motora B **In4v** – Služi za kontrolu smjera okretaja motora B **+5V** – Ulaz u koji ide vanjski izvor napajanja **GND** – Ground **+12V** – Ulaz vanjskog izvora napajanja za motore

Slika pr. 6.3. Izgled i pinout drajvera L298

Da bismo mogli napisati kod za neki od svojih budućih projekata koji bi koristili ovaj drajver, trebalo bi opisati princip rada drajvera L298N i kao prva stvar koju treba spomenuti je da on koristi takozvani integrirani čip H-most (*engl. H-bridge*). H-most je strujni krug koji se sastoji od četiri prekidača spojenih s motorom na način prikazan na slici ispod. Ako S2 i S3 prekidače zatvorimo, a S1 i S4 prekidače otvorimo, struja će teći u određenome smjeru. Ako zamijenimo stanja prekidača tako da su S2 i S3 otvoreni, a S1 i S4 zatvoreni, struja će kroz motor proteći u suprotnome smjeru. Uz to ne smijemo zatvoriti sva četiri prekidača odjednom ili oba prekidača na jednoj strani H-mosta (npr. S1 i S2) jer time izazivamo kratki spoj.



Slika pr. 6.4. *Princip rada drajvera L298*

Tablica ispod prikazuje sve moguće kombinacije stanja prekidača i njihov ishod.

S1	0	1	0	1	0	1	**0**	1	0	**1**	0	1	0	1	0	1
S2	0	0	1	1	0	0	**1**	1	0	**0**	1	1	0	0	1	1
S3	0	0	0	0	1	1	**1**	1	0	**0**	0	0	1	1	1	1
S4	0	0	0	0	0	0	**0**	0	1	**1**	1	1	1	1	1	1
Rezultat	*Nema promjene*	*Nema promjene*	*Nema promjene*	*Kratki spoj*	*Nema promjene*	*Režim kočenja*	*Motor se okreće u lijevu stranu*	*Kratki spoj*	*Nema promjene*	*Motor se okreće u desnu stranu*	*Režim kočenja*	*Kratki spoj*	*Kratki spoj*	*Kratki spoj*	*Kratki spoj*	*Kratki spoj*

S pomoću H-mosta vrlo je jednostavno zamijeniti polaritet na trošilu. Najčešće se koristi za zamjenu smjera vrtnje DC motora. H-most se koristi i u raznim drugim aplikacijama poput DC/AC, AC/AC ili DC/DC konvertera.

Iako je namijenjen za motore, može se koristiti i za druge uređaje koji za određenu potrebu zahtijevaju zamjenu polariteta na svojim ulaznim krajevima. Uz mogućnost zamjene polariteta drajver L298N ima još mogućnost kontrole brzine okretaja motora koristeći signale PWM (engl. *Puls-Width Modulation*).

Ovaj put za jednostavnu aplikaciju nećemo koristiti knjižnicu, ali je ovo idealan hardver za koji bismo na predmetu Programiranje mogli napisati jednostavan „drajver“. Za sada je dovoljno kontrolom stanja izlaza s pomoću funkcije digitalWrite() pokrenuti motore. Za brzinu okretaja motora koristi se funkcija analogWrite().

Kod Arduino:

```
// deklaracija pinova za prvi motor
int enA = 10;
int in1 = 9;
int in2 = 8;
// deklaracija pinova za drugi motor
int enB = 5;
int in3 = 7;
int in4 = 6;

void setup()
{
  // Postavi GPIO kao izlaze
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

void loop()
{
  // aktiviraj prvi motor
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // postavi brzinu okretaja prvog motora 200 (maks. 255)
  analogWrite(enA, 200);
  // aktiviraj drugi motor
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  // postavi brzinu okretaja drugog motora 200 (maks. 255)
  analogWrite(enB, 200);
  delay(2000);

  // Zamjeni smjerove okretaja motora
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(2000);

  // iskljuci oba motora
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(2000);
}
```

Ovaj dio koda ne trebate promatrati kao završen projekt, nego ga pokušati nadograditi već predloženim idejama ili realizirati neke svoje.

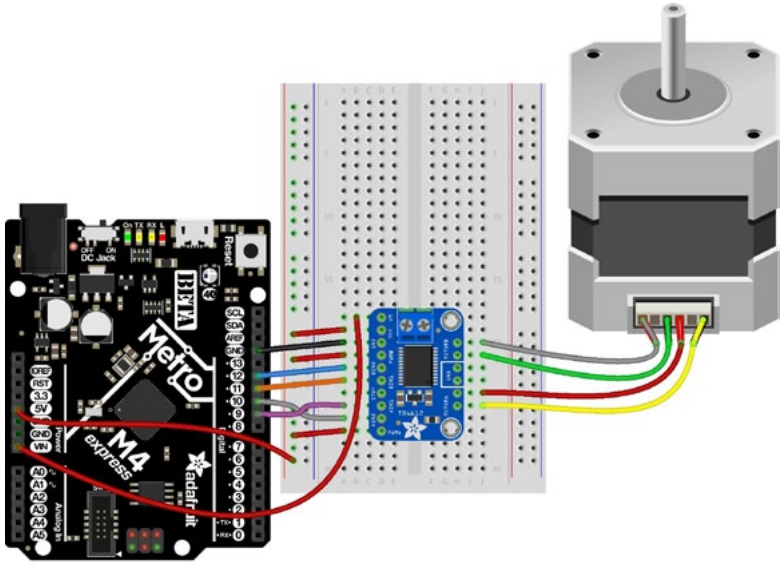
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- objašnjava ulogu drajvera motora - opisuje vezu između drajvera L298 i DC motora - demonstrira ulogu H-bridge kroz shematski prikaz - objašnjava princip rada drajvera L298
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje drajver L298 i DC motor - prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za drajver L298 i DC motor	- programira Arduino deklarirajući prvo pinove motora - rekonstruira programski kod prema vlastitim idejama
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - analizira projektni zadatak - nadograđuje programski kod vlastitim idejama

Prostor za vježbu:

```
 void setup() {
}
void loop() {
}
```

Projekt 7. TB6612
DC/Stepper motor drajver

Ovaj projekt sličan je prethodnome, s tim da se koristi drajver novije generacije. Kao i drajver L293, i TB6612 može kontrolirati bilo dva DC motora male snage ili jedan stepper motor. Princip rada potpuno je isti kao i kod drajvera L293: TB6612 sadrži dva puna H-mosta (četiri polu-H-mosta). To znači da možete pokretati 2–4 solenoida (samo dva mogu biti aktivna istodobno, na suprotnim mostovima), pojednostavljeno možemo upravljati ili s dva DC motora dvosmjerno, kao u prethodnom projektu, ili jednim stepper motorom. Važno je da strujno opterećenje ne prelazi 1,2 A budući da je to gornja granica struje ovog drajvera, mada on može podnijeti i strujni udar od 3 A, ali samo za kratko vrijeme (20 milisekundi). Treba napomenuti da TB6612 dolazi s ugrađenim diodama za povratni udar, tako da ne morate brinuti da će induktivni udar oštetiti vaš Arduino.



Slika pr. 7.1 *Shema spajanja drajvera TB6612 i stepper motor s kompatibilnom Adafruit Metro razvojnom pločom Arduino*

Na slici iznad nisu ucrtani vodiči za eksterni napon za drajver koji se mora osigurati sa stabilnog istosmjernog izvora napajanja (4.5 -13-5V).

Potrebni elementi za vježbu:

- ARDUINO UNO
- Drajver TB6612
- Stepper motor – moguće ih je naći u starim štampačima
- Pločica Matador
- Kablići

Kod ovog drajvera postoje tri skupine pinova te ćemo ih pobrojati i ukratko im objasniti njihove funkcije.

Pinovi za napajanje

Vmotor – pin za napajanje motora, preporučeno je da se napon kreće u granicama od 4.5 V do 13.5 V. Važno je napomenuti za neke vaše buduće projekte da je moguće stvaranje "smetnji" pa za slučaj da imate projekt s analognim senzorima ili RF čitačima, potrebno je da ovo napajanje bude odvojeno od ostatka kola ili barem filtrirano.

Vcc – napajanje logike drajvera. Naponska razina trebala bi biti za Arduino 5 V.

GND – motor ground.

Ulazni signali na drajveru

INA1, INA2 – ulazni pinovi za Motor A na H-mostu

PWMA – PWM ulaz za Motor A na H-mostu, ako za projekt nije potrebna PWM kontrola, pin spojiti na +5 V.

INB1, INB2 – ulazni pinovi za Motor B na H-mostu

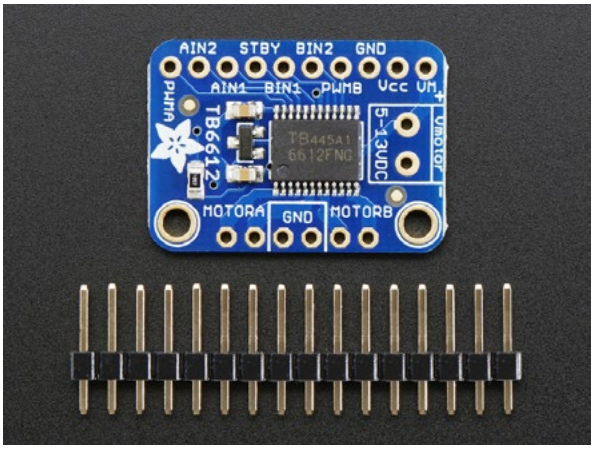
PWMB – PWM ulaz za Motor B na H-mostu, ako za projekt nije potrebna PWM kontrola, pin spojiti na +5 V.

STBY – standby pin za brzo isključivanje motora, pull up ka Vcc preko 10 K otpor. Spojiti na masu za isključenje.

Izlazni signali za motor

Motor A – izlazi za motor A ili A namotaj stepper motora kontroliran od INA1, INA2 i PWMA

Motor B – izlazi za motor B ili B namotaj stepper motora kontroliran od INB1, INB2 i PWMB



Slika pr. 7.2 Izgled drajvera TB6612

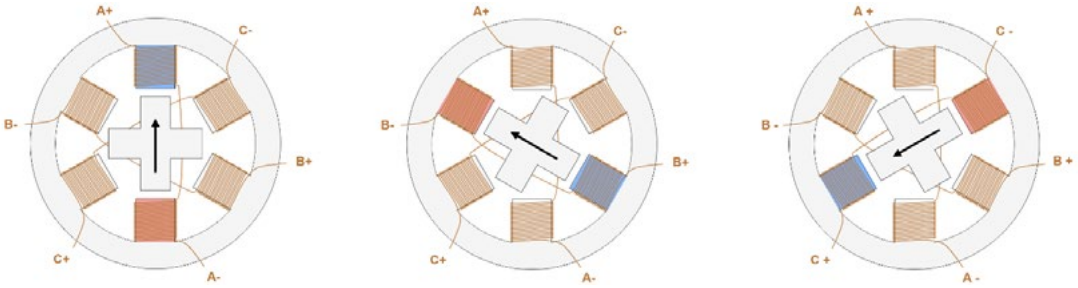
Kako drajver dolazi kao na slici iznad, potrebno ga je pripremiti za upotrebu. Najlakša je opcija iskoristiti ploču matador iz starter kita te postaviti drajver i zalemiti pinove. Tehnika lemljenja objašnjena je kroz slike u Projektu 8.

Stepper ili koračni motor je električni motor čija je glavna osobina da se njegova osovina rotira koračno, odnosno pomiče se za fiksni iznos stupnjeva. Ova je osobina postignuta zahvaljujući unutarnjoj strukturi motora i omogućuje da se zna točan kutni položaj osovine jednostavnim brojanjem koliko je koraka osovina motora napravila, bez potrebe za senzorom.

Kao i svi elektromotori, motori stepper imaju stacionarni (stator) i pokretni dio (rotor). Na statoru se nalaze zupci na kojima su namotaji žice, dok je rotor ili trajni magnet ili željezna jezgra promjenjive reluktancije.

Osnovni princip rada motora stepper je sljedeći: dovođenjem pod napon jedne ili više faza statora struja koja teče u zavojnici stvara magnetsko polje i rotor se poravnava s tim poljem. Dovođenjem pod napon različitih namotaja u sekvenci rotor se može rotirati za točno definiran kut kako bi se postigao željeni krajnji položaj. Slika pr. 7.3 prikazuje principe rada.

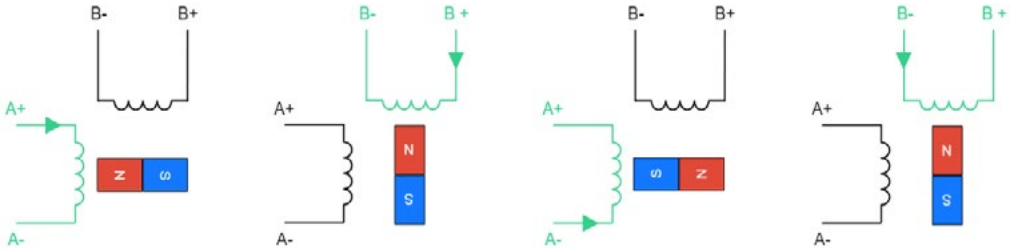
Na početku je zavojnica A pod naponom i rotor je poravnat s magnetskim poljem koje proizvodi. Kada je zavojnica B pod naponom, rotor se rotira u smjeru kazaljke na satu za 60° kako bi se uskladio s novim magnetskim poljem. Isto se događa kada je zavojnica C pod naponom. Na slikama boje zubaca statora označavaju smjer magnetskog polja koje stvara namot statora.



Slika pr. 7.3 Princip rada motora stepper

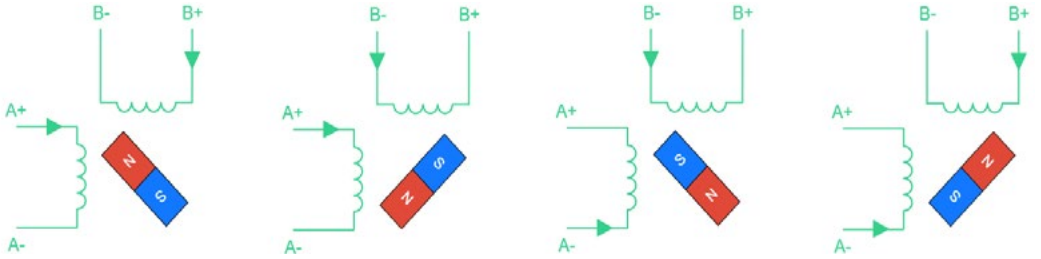
Postoje četiri različite tehnike upravljanja motorom stepper:

U modu **wave** samo je jedna faza pod naponom. Radi jednostavnosti reći ćemo da struja teče u pozitivnom smjeru od + do - (npr. od A+ do A-), u suprotnom smjer je negativan. Počevši s lijeve strane struja teče samo kroz namotaj A u pozitivnom smjeru, a rotor, predstavljen magnetom, usklađen je s magnetskim poljem koje stvara. U sljedećem koraku struja teče samo kroz namotaj B u pozitivnom smjeru, a rotor se okreće za 90° u smjeru kazaljke na satu kako bi se uskladio s magnetskim poljem koje stvara namotaj B. Kasnije se namotaj A ponovno aktivira, ali struja teče u negativnom smjeru, a rotor se ponovno okreće za 90°. U posljednjem koraku struja teče negativno u namotaju B i rotor se ponovno okreće za 90°.



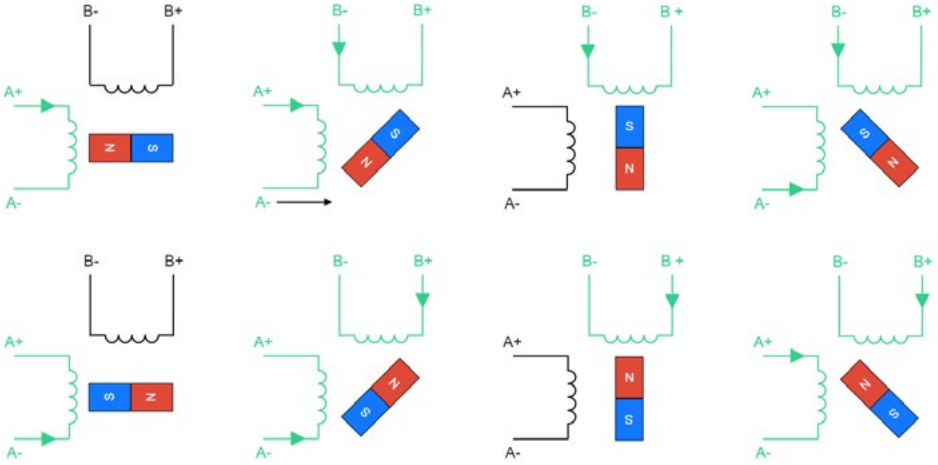
Slika pr 7.4 Wave mod

U načinu rada **full-step** dva namotaja uvijek su pod naponom u isto vrijeme. Slika ispod prikazuje različite korake ovog načina kontrole. Koraci su slični onima u načinu rada *wave*, a najznačajnija razlika je u tome što s ovim načinom rada motor može proizvesti veći moment jer u motoru teče više struje i stvara se jače magnetsko polje.



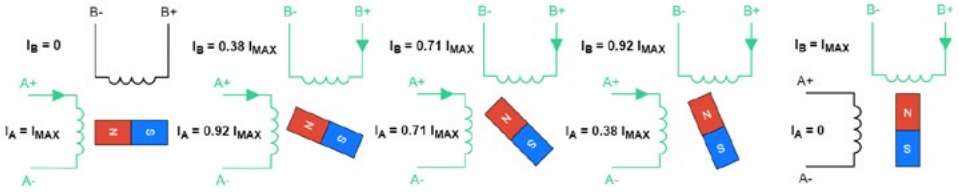
Slika pr. 7.5 Full step mod

Half-step mod je kombinacija prethodna dva načina kontrole stepper motora. Korištenje ove kombinacije omogućava da se veličina koraka smanji za polovicu (u ovom slučaju 45° umjesto 90°). Jedini je nedostatak što moment koji proizvodi motor nije konstantan, jer je veći kada su oba namotaja pod naponom, a slabiji kada je pod naponom samo jedan namotaj.



Slika pr. 7.6 Half step mod

Microstepping se može promatrati kao daljnje poboljšanje načina rada *half step*, jer omogućava još više smanjenje veličine koraka i konstantan izlazni moment. To se postiže kontroliranjem intenziteta struje koja teče u svakom namotaju. Korištenje ovakvog načina upravljanja zahtijeva kompleksniji drajver motora.



Slika pr. 7.7 Microstepping

U nastavku ćemo navesti neke od prednosti stepper motora, naime zbog svoje unutarnje strukture ovi motori ne zahtijevaju senzor za detekciju položaja motora. Budući da se motor kreće izvodeći "korake", jednostavnim brojanjem ovih koraka možete dobiti položaj motora u određenome trenutku.

Osim toga, upravljanje motorom prilično je jednostavno. Motoru je potreban drajver, ali ne trebaju složeni izračuni ili podešavanje da bi ispravno radio, odnosno kontrolni zahtjevi manji su u usporedbi s drugim motorima. S mikrokorakom može se postići visoka točnost položaja, do približno 0,007°.

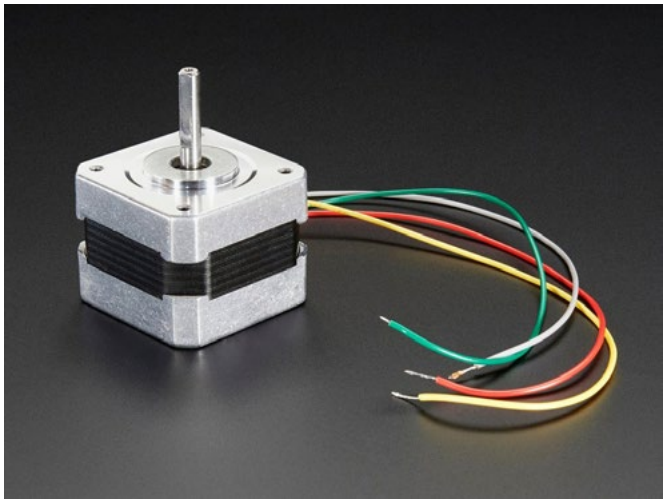
Ovi motori nude dobar zakretni moment pri malim brzinama, izvrsni su za držanje položaja, a također imaju dug životni vijek.

Naravno, ovi motori i pored svega imaju i nedostatke. Kao prvo mogu "propustiti" korak ako je moment opterećenja previsok. To negativno utječe na upravljanje, jer ne postoji način da se sazna stvarni položaj motora. Korištenje mikrokoraka povećava vjerojatnost da će koračni motori imati ovaj problem.

Ovi motori uvijek troše maksimalnu struju čak i kada su u praznom hodu, što pogoršava učinkovitost i može uzrokovati pregrijavanje, pri tome i imaju mali okretni moment i postaju prilično bučni pri velikim brzinama.

Ukratko, stepper motori dobri su kada trebate jeftino rješenje koje se lako kontrolira i kada učinkovitost i veliki zakretni moment pri velikim brzinama nisu potrebni. Ovo ih čini idealnim rješenjem za hobiste, a odlični su i kao prvi korak u smislu istraživanja robotike.

Mi ćemo u svom primjeru koristiti bipolarni stepper motor jer takvi motori imaju namotaje s dva izvoda te zahtijevaju H-most za upravljanje. Unipolarni stepper motori imaju središnji izvod na namotaju i mogu se prepoznati po tome što uvijek imaju više od četiri vodiča na sebi.



Slika pr. 7.8 Bipolarni stepper motor

Spojit ćemo svoj drajver prema shemi spoja i to:

- Vmotor na 12 V
- Vcc na 5 V
- GND na masu
- AIN2 u pin 10
- AIN1 u pin 9
- BIN1 u pin 11
- BIN2 u pin 12
- PWMA i PWMB na Vcc

Zatim zakačite jedan namotaj motora na motor A priključak (crvena i žuta), a drugu namotaj na motor B priključak (zelena i siva/smeđa). Ako imate drugi motor, morat ćete malo eksperimentirati da shvatite koje su žice koji namotaj. Možete koristiti multimetar za mjerenje između žica, one s malim otporom između njih su nekoliko namotaja. Ako motor vibrira, ali se ne okreće, provjerite jesu li svi vodiči spojeni i pokušajte okrenuti nekoliko vodiča ili ponovno provjeriti parove.

Kod Arduino:

```
#include <Stepper.h>

// promijenite ovo na broj koraka na svom motoru #define STEPS 200
// kreiraj instancu stepper klase, specificirajući broj koraka motora i
pinova (STEPS, 4, 5, 6, 7);
void setup(){
  Serial.begin(9600);
  Serial.println(„Stepper test!“);
  // postavite brzinu motora 30 obrtaja u min
  stepper.setSpeed(60);}
void loop(){
  Serial.println(„Forward“);
  stepper.step(STEPS);
  Serial.println(„Backward“);
  stepper.step(-STEPS);}
```

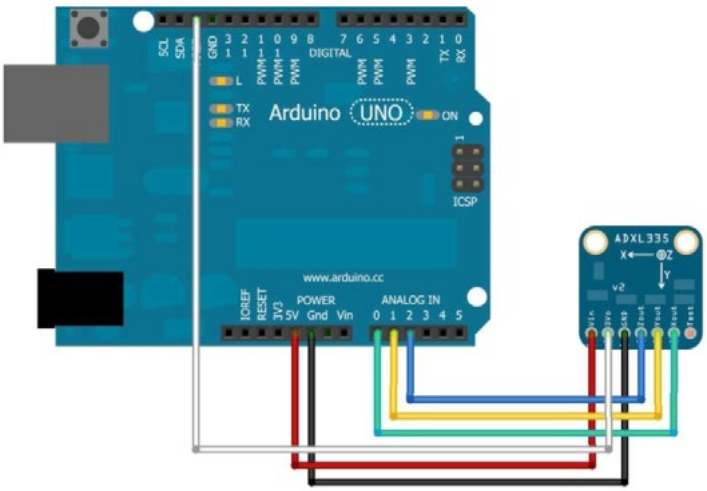
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- objašnjava ulogu elektronskih komponenti koje su korištene u projektu	- opisuje ulogu elektromotora - objašnjava princip rada stepper motora - navodi ulogu drajvera TB6612 - razvrstava pinove drajvera u tri skupine - povezuje stečena znanja iz Fizike, oblast elektricitet i magnetizam - razlikuje bipolarni stepper od unipolarnog steppera prema njihovoj ulozi i načinu rada
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje stepper motor i drajver TB6612 s Arduino - koristi lemilicu uz mjere opreza i nadzor nastavnika/-ice
- kreira programski kod za stepper motor	- programira Arduino uvodeći knjižnicu Stepper.h - rekonstruira programski kod prema vlastitim idejama
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - analizira projektni zadatak - nadograđuje programski kod vlastitim idejama

Prostor za vježbu:

```
 void setup() {

}
void loop() {

}
```

Projekt 8.
Troosni brzinomjer ADXL335

U ovom projektu koristit ćemo jedan MEMS uređaj (MEMS – mikroelektromehanički sustav). Sam senzor sastoji se od kombinacije mikrostrojno obrađene strukture na silikonskoj pločici. Konstrukcija je poduprta polisilicijskim oprugama koje joj omogućuju skretanje u trenutku ubrzanja po osama x, y i z. Otklon uzrokuje promjenu kapacitivnosti između fiksnih ploča i ploča pričvršćenih na viseću konstrukciju. Ova promjena kapacitivnosti na svakoj osi pretvara se u izlazni napon proporcionalan ubrzanju po toj osi. OK, ovo zvuči baš komplicirano, ali mi svakodnevno koristimo benefit ovakve tehnologije u svojim pametnim telefonima. Pored ovog integriranog kola na PCB akcelerometru nalazi se i jedan naponski regulator, s pomoću kojega je olakšana integracija ovog uređaja s razvojnom platformom Arduino.



Slika pr. 8.1 *Shema spajanja senzora ADXL 335*

NAPOMENA:
Ako spojite pin EV0 na pin AREF, morate analognu referencu postaviti na EXTERNAL prije pozivanja analogRead() u funkciji setup(). U suprotnom ćete internu referencu kratko spojiti s vanjskom referencom, što bi moglo oštetiti vašu ploču Arduino.

Nadamo se da vidite da vaši projekti postaju sve zanimljiviji, te da vam odmah padaju na pamet neke nove ideje kako biste ovaj senzor mogli iskoristiti u nekom od prethodnih projekata, a evo ponovno nekih ideja:

1. Kontrola smjera vrtnje motora
2. Praćenje nagiba pri kretanju autonomnog automobila
3. Integracija 2x16 LCD zaslona u projekt

Potrebni elementi za realizaciju projekta navedeni su ispod:

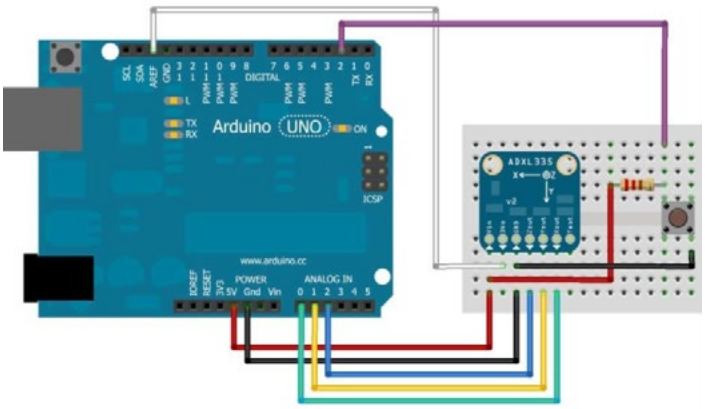
1. ARDUINO UNO
2. 220Ω potenciometar
3. Ploča Matador
4. Taster
5. Kablići

Kao i kod većine senzora postoje neke varijacije kod izlaznih vrijednosti sa senzora. Za neke nekritične aplikacije tipa kontrolera za igre ili detekcije nagiba ove varijacije nisu tako „strašne“ i možemo ih zanemariti u izradi svoje aplikacije. Ali za aplikacije kada se mora striktno voditi računa o preciznosti mjerenja kalibracija senzora je najbolja metoda.

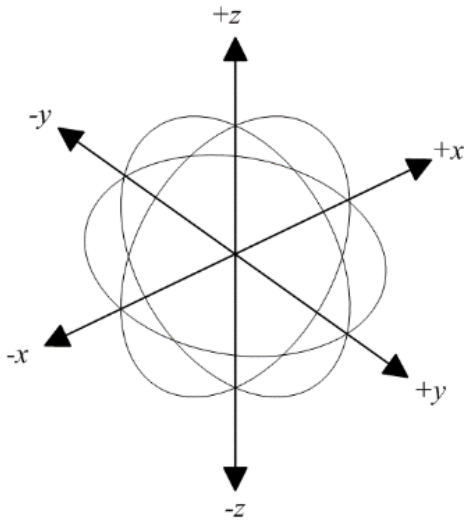
Ubrzanje kod ovog senzora mjeri se u jedinicama gravitacijske sile, gdje 1G predstavlja gravitacijsko povlačenje na površini. Kako je gravitacijska sila stabilna veličina, to je čini dobrom referencom za kalibraciju, osim ako ne živite u oblacima 😊. Samo mala napomena prije nego probate kalibrirati svoj senzor: proporcionalni izlaz ovog senzora znači da se izlazni napon linearno povećava s ubrzanjem u rasponu od 0V pri -3G do 3.3V pri +3G.

Za kalibraciju senzora na gravitacijsku referencu potrebno je odrediti izlaz senzora kada je on precizno pozicioniran spram ose gravitacijske sile. Naravno, u specijaliziranim laboratorijama koriste se posebni pristroji ili prihvati za ovo, ali vi ćete dobiti dobre rezultate i bez toga.

Kao prvo potrebno je na standardnu shemu spajanja dodati još dva elementa kao na slici ispod, dodajući na ploču matador 220Ω otpor i jedan taster.



Slika pr. 8.2 *Kalibriranje senzora ADXL 335*



Slika pr. 8.3 *Smjer djelovanja gravitacijskih sila*

Procedura za kalibraciju je sljedeća:

1. Učitati kod za kalibraciju
2. Otvoriti aplikaciju Serial Monitor
3. Postavite matador na ravnu površinu
   - Stisnuti i držati taster dok se na Serila Monitoru ne ispiše „Calibrate“
   - Na ovaj način kalibrirali ste minimalnu vrijednost za **-Z** osu
4. Postavite matador na prednju ivicu i ponovite postupak za kalibraciju **+Y**
5. Ponoviti proceduru za ostale tri ivice ploče matador za kalibraciju **+X, -Y i -X**.
6. Okrenite matador naopako i uradite proceduru za kalibraciju **+Z**.

Nakon završene kalibracije na serijskom monitoru dobit ćete sirove kalibracijske podatke za svaku osu. Ovi podatci mogu se koristiti kao konstante u vašim aplikacijama.

Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625 511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625 511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625 511, 511, 625 :: -0.01G, 0.07G, 1.00G
Raw Ranges: X: 408-616, Y: 398-610, Z: 422-625 511, 511, 625 :: -0.01G, 0.07G, 1.00G

Kod Arduino:

```
const int xInput = A0;
const int yInput = A1;
const int zInput = A2;
const int buttonPin = 2;

// "Sirovi" podaci:
// inicijalizacija na "sredinu mogućeg raspona podataka"
int xRawMin = 512;
int xRawMax = 512;

int yRawMin = 512;
int yRawMax = 512;

int zRawMin = 512;
int zRawMax = 512;

// multisampling radi smanjenja smetnji
const int sampleSize = 10;

void setup()
{
  analogReference(EXTERNAL);
  Serial.begin(9600);
}
void loop()
{
  int xRaw = ReadAxis(xInput);
  int yRaw = ReadAxis(yInput);
  int zRaw = ReadAxis(zInput);

  if (digitalRead(buttonPin) == LOW)
  {
    AutoCalibrate(xRaw, yRaw, zRaw);
  }
  else
  {
    Serial.print(„Raw Ranges: X: „);
    Serial.print(xRawMin);
    Serial.print(„-");
    Serial.print(xRawMax);

    Serial.print(„, Y: „);
    Serial.print(yRawMin);
    Serial.print(„-");
    Serial.print(yRawMax);

 Serial.print(„, Z: „);
    Serial.print(zRawMin);
    Serial.print(„-");
    Serial.print(zRawMax);
    Serial.println();
    Serial.print(xRaw);
    Serial.print(„, „);
```

```
    Serial.print(yRaw);
    Serial.print(„, „);
    Serial.print(zRaw);

    // pretvorba sirovih podataka na ‚milli-G-s“
    long xScaled = map(xRaw, xRawMin, xRawMax, -1000, 1000);
    long yScaled = map(yRaw, yRawMin, yRawMax, -1000, 1000);
    long zScaled = map(zRaw, zRawMin, zRawMax, -1000, 1000);

    // reskaliranje na ubrzanje po osama
    float xAccel = xScaled / 1000.0;
    float yAccel = yScaled / 1000.0;
    float zAccel = zScaled / 1000.0;

    Serial.print(„ :: „);
    Serial.print(xAccel);
    Serial.print(„G, „);
    Serial.print(yAccel);
    Serial.print(„G, „);
    Serial.print(zAccel);
    Serial.println(„G“);

  delay(500);
  }
}

//
// Čitanje uzoraka „sampleSize“ i traženje srednje vrijednosti
//
int ReadAxis(int axisPin)
{
  long reading = 0;
  analogRead(axisPin);
  delay(1);
  for (int i = 0; i < sampleSize; i++)
  {
    reading += analogRead(axisPin);
  }
  return reading/sampleSize;
}

   //
// traženje ekstremnih vrijednosti za svako osu
//
void AutoCalibrate(int xRaw, int yRaw, int zRaw)
{
  Serial.println(„Calibrate“);
  if (xRaw < xRawMin)
  {
    xRawMin = xRaw;
  }
  if (xRaw > xRawMax)
  {
    xRawMax = xRaw;
  }
```

```
  if (yRaw < yRawMin)
  {
    yRawMin = yRaw;
  }
  if (yRaw > yRawMax)
  {
    yRawMax = yRaw;
  }

  if (zRaw < zRawMin)
  {
    zRawMin = zRaw;
  }
  if (zRaw > zRawMax)
  {
    zRawMax = zRaw;
  }
}
```

Kao što vidite, ova aplikacija za kalibraciju malo je kompliciranija od svega što ste do sada radili, ali to samo tako izgleda. Dvije su stvari koje su nove i to eksterne funkcije, pri čemu je jedna tipa **int**, a druga tipa **void**. Prvo ćemo objasniti funkciju AutoCalibrate(), koja ne vraća ništa (povratni tip je void). Naime, ta funkcija može imati argumente kao u primjeru:

```
void AutoCalibrate(int xRaw, int yRaw, int zRaw)
```

Dakle kod deklaracije funkcije unutar zagrada imamo moguće argumente, u ovom slučaju tri argumenta tipa int. Kod poziva ove funkcije unutar petlje *loop* mi samo toj funkciji proslijedimo tri argumenta koju ona iskoristi za proračun, ali ne vrati nam nikakvu vrijednost.

```
if (digitalRead(buttonPin) == LOW)
{
 AutoCalibrate(xRaw, yRaw, zRaw);
}
```

Drugi tip funkcije koji vidimo u prethodnoj aplikacije je funkcija koja ima „povratnu“ vrijednost, odnosno nakon proračuna vraća vrijednost tipa int.

```
int ReadAxis(int axisPin)
{
  long reading = 0;
  analogRead(axisPin);
  delay(1);
  for (int i = 0; i < sampleSize; i++)
  {
    reading += analogRead(axisPin);
  }
  return reading/sampleSize;
}
```

Funkciju deklariramo na način da ispred naziva funkcije definiramo vrstu podatka koje će ta funkcija nakon proračuna vratiti. U ovom konkretnom slučaju funkcija ima jedan argument tipa *int*:

```
(int axisPin)
```

Nakon proračuna vratit će:

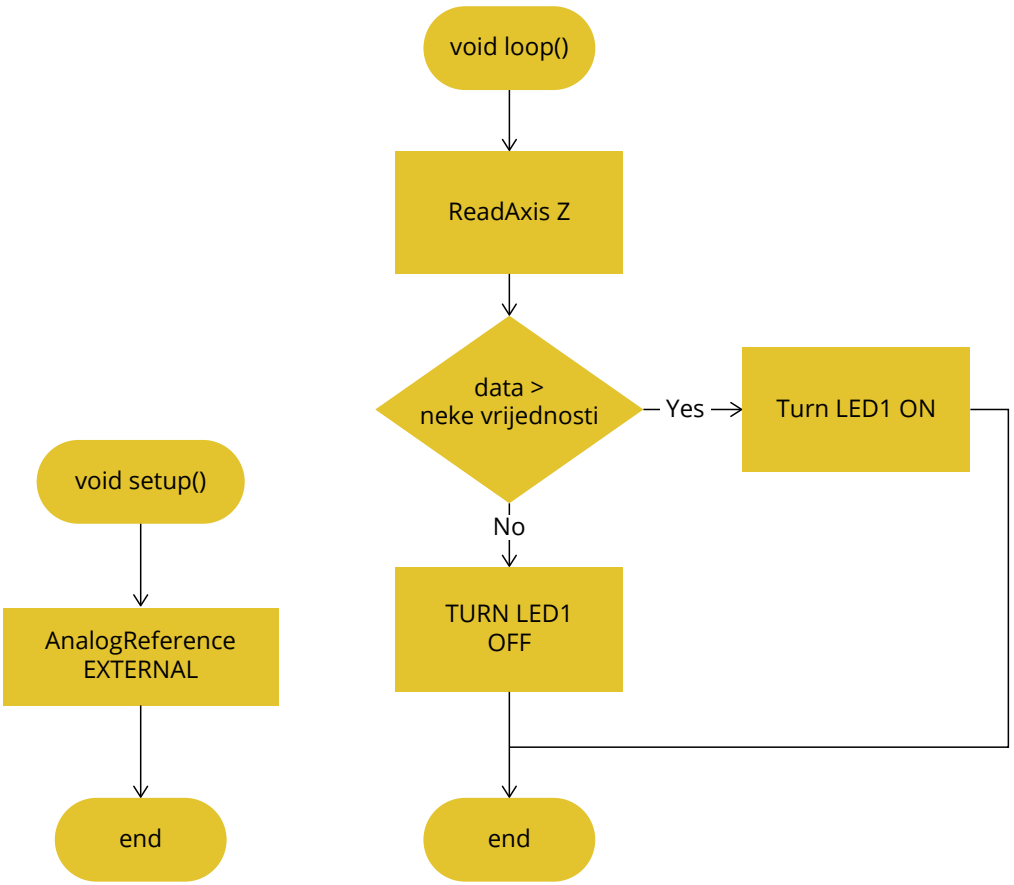
```
return reading/sampleSize;
```

Poziv funkcije u petlji *loop* urađen je na sljedeći način:

```
int xRaw = ReadAxis(xInput);
int yRaw = ReadAxis(yInput);
int zRaw = ReadAxis(zInput);
```

Pa i nije tako strašno, ovako u narednim projektima možete pisati malo ozbiljnije aplikacije, jer možete segmente aplikacije izdvojiti iz petlje *loop* te početi pisati funkcije koje će vam olakšati rješavanje problema. Uz napomenu da to bude jednostavno (engl. *Keep it simple!*).

Za kraj ovog projekta ostali smo dužni pojednostavljenu algoritamsku strukturu za jedan jednostavan projekt s ADXL 335.



Slika pr. 8.4. *Pojednostavljena algoritamska struktura aplikacije*

ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica: - opisuje ulogu elektronskih komponenti koje su korištene u projektu - proširuje stečena znanja iz oblasti gravitacije	**Učenik/-ica:** - opisuje ulogu MEMS uređaja - proširuje stečena znanja iz oblasti gravitacije - objašnjava ulogu troosnog brzinomjera ADXL335 - Koristi se Serial Monitorom za učitavanje rezultata
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- izvršava kalibraciju senzora ADXL 335 prema shemi spoja i proceduri - povezuje preostale komponente u projektu prema shemi spoja
- kreira algoritamsku strukturu spoja	- procjenjuje prednosti i ograničenja algoritamskog pristupa u rješavanju problema - vizualizira projektni zadatak kroz algoritamsku shemu
- kreira programski kod za daljinsko upravljanje	- programira projekt uz adekvatne smjernice nastavnika/-ice - deklarira funkcije i promjenljive u programskom kodu - rekonstruira programski kod prema vlastitim idejama
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - koristi se Serial Monitorom za testiranje projektnog zadatka - analizira projektni zadatak - razvija vlastite ideje za primjenu MEMS uređaja

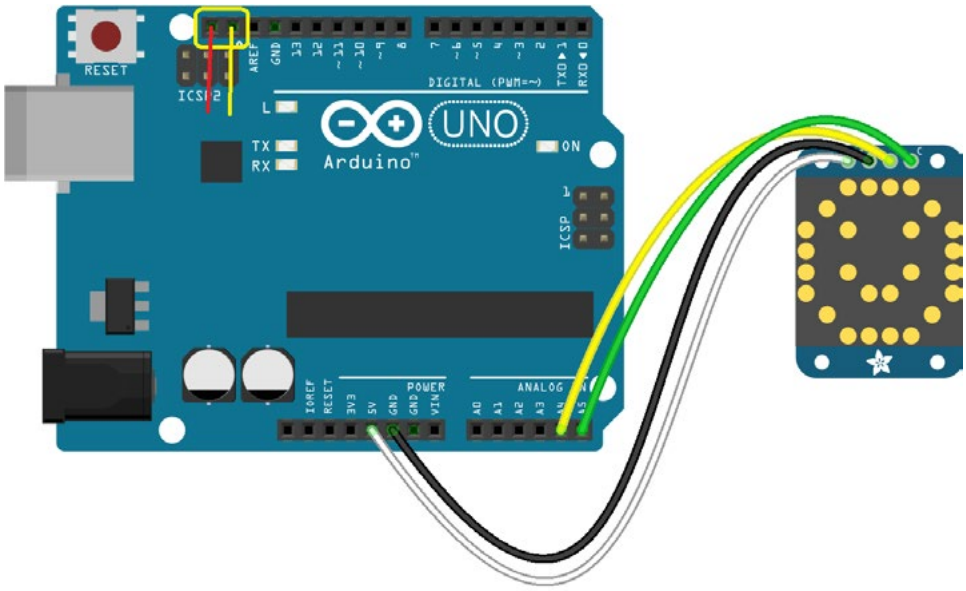
Prostor za vježbu:

```
.....

.....

.....

 void setup() {

.....

.....

.....

}

void loop() {

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

}
```

Projekt 9.
LED Matrix drajver HT16K33/MAX7219

Ponovno LED, ali samo 64 komada istih. Odličan način da napravimo mali zaslon koji koristi matricu od 8x8 led dioda kojom upravlja čip HT16K33, koji ima sposobnost da upravlja multipleksiranom matricom 16x8, što predstavlja 128 LED-a. Impresivno! Komunikacijski protokol koji čip koristi je I²C, stoga kao kod serijske komunikacije za transfer podataka dovoljna su nam samo dva pina, pri čemu nećemo detaljno ulaziti u tip komunikacije I²C, ali se mora reći da je to komunikacijska sabirnica bazirana na komunikaciji između uređaja *master* (Arduino) i uređaja *slave*, od kojih svaki ima jedinstvenu adresu. Kako led matrix drajver dolazi u vidu *breakout boarda*, na sebi ima opciju da možemo podesiti osam različitih I²C adresa, te na taj način možemo kontrolirati 1024 LED-e.

Važno je napomenuti da čip ima integriran modul za držanje konstantne struje upravljanja te LED matrica uvijek ima konzistentno isijavanje svjetlosti na LED-u. Također je moguće mijenjati intenzitet svjetlosti svih LE dioda istodobno (nije moguće pojedinačno) u matrici i to u 16 različitih razina.



Slika pr. 9.1. *Shema spajanja LED matrice*

Ok, potpuno vas razumijem ako voljenoj osobi želite na ovoj matrici nacrtati srce i sami smo to prvo uradili 😊. Ali kao i uvijek do sada evo nekoliko ideja koje možete iskoristiti za nadogradnju početnog projekta:

1. Bedž reklama
2. Igra balansiranja s ADXL335
3. Scrolling reklama s više zaslona

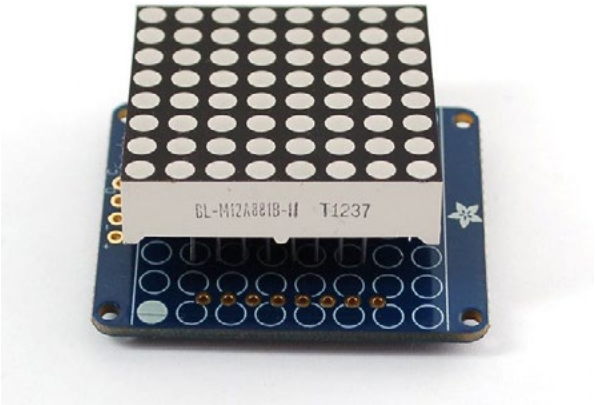
Naravno, ne morate stati na ovome. Ako dobijete neku svoju ideju, budite slobodni da je provedete, na kraju krajeva istraživanjem se najviše uči. Svaki od navedenih prijedloga kao i sve druge projekte možete nadograditi drugim komponentama te tako učiniti projekt zanimljivim.

Potrebni elementi za realizaciju projekta navedeni su ispod:

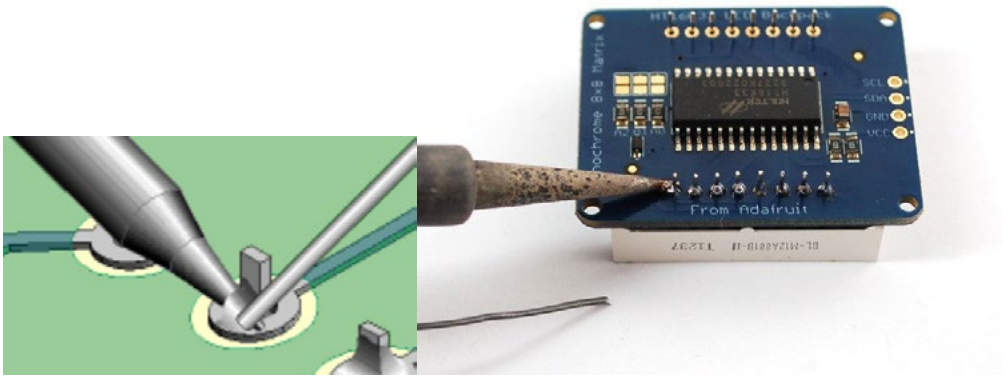
1. ARDUINO UNO
2. HT16K33 backpack
3. Matrica 8x8 led
4. Ploča Matador
5. Kablići

Kako u većini slučajeva ovaj drajver i zaslon budu dostavljeni kao DIY kit, priloženo vam je i uputstvo za njegovo sastavljanje prije nego ga počnemo testirati.

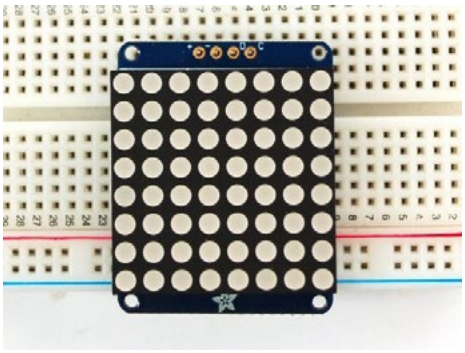
Korak 1: Podesite zaslon kao na slici ispod, tekst tipa zaslona treba stajati pod 90° u odnosu na pinove komunikacijske sabirnice.



Korak 2: Koristeći lemilicu zalemite pinove zaslona za PCB. Napomena: tinol žicu dodati na već prethodno zagrijano mjesto za lemljenje i pri izvlačenju povući lemilicu uz pin zaslona.

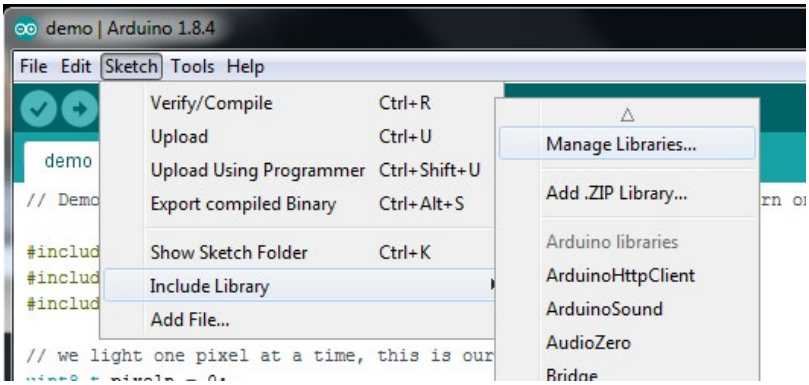


Korak3: Ubaciti 4-pinski konektor u ploču matador i potom montirati backpack PCB kao što je to prikazano na slici. Pinove zalemiti koristeći istu tehniku kao što je to gore napomenuto.



Kako će se za ovaj projekt koristiti gotova knjižnica, u nastavku će biti prikazano kako ju je moguće instalirati koristeći Library Manager. Pored knjižnice Adafruit LED Backpack, potrebno je instalirati i knjižnicu Adafruit GFX.

Prvo je potrebno iz izbornika Sketches izabrati opciju *Include Library* pa potom izabrati opciju *Manage Libraries...*



Slika pr. 9.2. *Putanja do opcije Manage Libraries*

U pop-up prozoru u polju za traženje upišite Adafruit LED backpack i instalirajte istu knjižnicu, a isto ponovite i za knjižnicu Adafruit GFX. Ako koristite stariju inačicu Arduino IDE od 1.8.10, potrebno je da instalirate i knjižnicu Adafruit_DusIO.

Ako ste pažljivo pratili uputstvo, onda ćete u svom IDE-u na putanji **File->Examples->Adafruit_LEDBackpack->matrix88 naći demo skicu koju je potrebno učitati. Kako je skica intelektualno vlasništvo, ostavit ćemo neizmijenjene neke postojeće komentare.**

Kod Arduino:

```
This is a library for our I2C LED Backpacks

  Designed specifically to work with the Adafruit LED Matrix backpacks
  ----> http://www.adafruit.com/products/872
  ----> http://www.adafruit.com/products/871
  ----> http://www.adafruit.com/products/870

  These displays use I2C to communicate, 2 pins are required to
   interface. There are multiple selectable I2C addresses. For back-
packs
  with 2 Address Select pins: 0x70, 0x71, 0x72 or 0x73. For backpacks
  with 3 Address Select pins: 0x70 thru 0x77

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ****************************************************/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include „Adafruit_LEDBackpack.h“

Adafruit_8x8matrix matrix = Adafruit_8x8matrix();

void setup() {
  Serial.begin(9600);
  Serial.println(„8x8 LED Matrix Test“);

  matrix.begin(0x70);  // inicijaliziraj s navedenom i2c adresom
}

static const uint8_t PROGMEM
  smile_bmp[] =
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10100101,
    B10011001,
    B01000010,
    B00111100 },
  neutral_bmp[] =
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10111101,
    B10000001,
    B01000010,
    B00111100 },
```

```
  frown_bmp[] =
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10011001,
    B10100101,
    B01000010,
    B00111100 };

void loop() {
  matrix.clear();
  matrix.drawBitmap(0, 0, smile_bmp, 8, 8, LED_ON);
  matrix.writeDisplay();
  delay(500);

  matrix.clear();
  matrix.drawBitmap(0, 0, neutral_bmp, 8, 8, LED_ON);
  matrix.writeDisplay();
  delay(500);

  matrix.clear();
  matrix.drawBitmap(0, 0, frown_bmp, 8, 8, LED_ON);
  matrix.writeDisplay();
  delay(500);

  matrix.clear();       // očisti displej
  matrix.drawPixel(0, 0, LED_ON);
  matrix.writeDisplay();  // prikaži na LED matrici
  delay(500);

  matrix.clear();
  matrix.drawLine(0,0, 7,7, LED_ON);
  matrix.writeDisplay();  // prikaži na LED matrici
  delay(500);

  matrix.clear();
  matrix.drawRect(0,0, 8,8, LED_ON);
  matrix.fillRect(2,2, 4,4, LED_ON);
  matrix.writeDisplay();  // prikaži na LED matrici
  delay(500);

  matrix.clear();
  matrix.drawCircle(3,3, 3, LED_ON);
  matrix.writeDisplay();  // prikaži na LED matrici
  delay(500);

  matrix.setTextSize(1);
  matrix.setTextWrap(false);  // ne želimo wrap tekst, već skrolan
  matrix.setTextColor(LED_ON);
  for (int8_t x=0; x>=-36; x--) {
    matrix.clear();
    matrix.setCursor(x,0);
    matrix.print(„Hello“);
    matrix.writeDisplay();
```

```
    delay(100);
  }
\  matrix.setRotation(3);
  for (int8_t x=7; x>=-36; x--) {
    matrix.clear();
    matrix.setCursor(x,0);
    matrix.print(„World");
    matrix.writeDisplay();
    delay(100);
  }
  matrix.setRotation(0);
}
```

Sada kada ste vidjeli kako Vaša LED matrica radi, možete početi provoditi svoje ideje. Matrica 8x8 podržava sve funkcije iz knjižnice Adafruit GFX– crtanje pixeal, linije, pravokutnika, kao i malih bitmapa.

Potrebno je spomenuti dvije funkcije koje vam mogu koristiti u projektima, a to su:

- **setBrightness (*brightness*)** – omogućava vam podešavanje intenziteta svjetlosti čitavog zaslona na skali od 0 do 15.
- **blinkRate (*rate*)** – omogućava blinkanje cijelog zaslona na skali od 0 do 3.

LED matrix je zanimljiv *gadget* i ovaj će se put u projektu izostaviti algoritamska struktura kojom sugeriramo zadatak i pustiti vašu maštu na volju da osmislite zanimljiv projekt, a i ne sumnjam da ćete u tome uspjeti. Sretno!

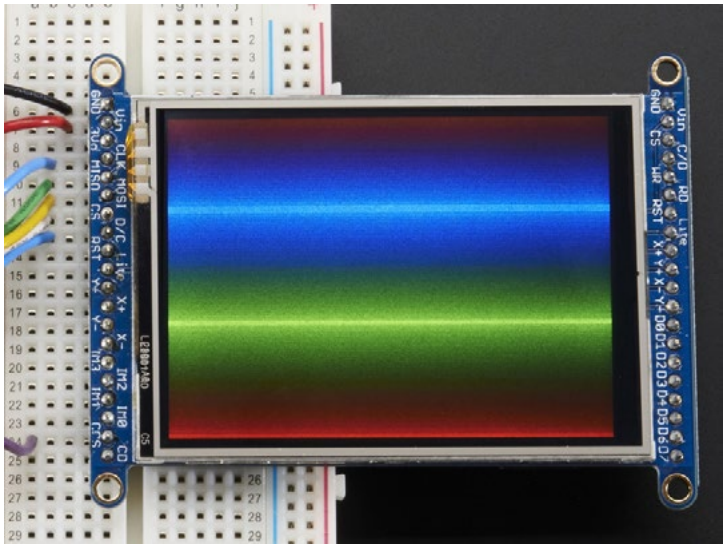
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- kreira mali zaslon koji koristi matricu od 8x8 LE dioda kojom upravlja čip HT16K33 - objašnjava ulogu tipa komunikacije I²C - opisuje osnovna svojstva primjene čipa HT16K33 - navodi uloge funkcija setBrightness (brightne-ss) i blinkRate (rate)
- koristi električnu shemu spoja za povezi-vanje Arduina i elektronskih komponenti	- sastavlja drajver i zaslon prema priloženom upustvu DIY kita - spaja pinove mikrokontrolera prema shemi spoja - povezuje preostale komponente u projektu prema shemi spoja
- kreira programski kod za daljinsko upra-vljanje	- programira Arduino za upotrebu LED Matrix drajvera HT16K33/MAX7219 - primjenjuje stečeno znanje iz nizova i matrica - učitava demo skicu File->Examples->Adafru-it_LEDBackpack->matrix88 - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar pro-grama za realizaciju projektnog zadatka	- instalira knjižnice Adafruit LED Backpack i Adafruit GFX iz Library Managera - po potrebi instalira i knjižnicu Adafruit_DusIO u zavisnosti od inačice Arduino IDE - poziva knjižnice unutar programskog koda
- verificira i izvršava program	- analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika/-ice
- testira projektni zadatak	- učitava kod na platformu Arduino - izvršava testiranje zaslona - analizira projektni zadatak - primjenjuje zaslon u drugim projektnim ideja-ma - razvija vlastite ideje za primjenu zaslona

Prostor za vježbu:

```
.....

.....

.....

 void setup() {

.....

.....

.....

}

void loop() {

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

}
```

Projekt 10.
TFT LCD s ekranom osjetljivim na dodir

Postajemo digitalni. Za razliku od nas naši učenici i učenice u potpunosti su TFT generacija. Naime, odmaknut ćemo se od matrica i 2x16 LCD zaslona i krećemo u projekt koji je malo teži, ali utoliko više zabavan. TFT LCD je oblik LCD zaslona koji koristi tehnologiju tankog filmskog tranzistora, stoga naziv TFT (skraćenica od *Thin-Film-Transistor*). Zaslon posjeduje i rezistivni *touch* tako da je moguće napraviti i interaktivnu aplikaciju. Sam zaslon posjeduje mikrokontroler s RAM međuspremnikom, što olakšava njegovu kontrolu. Kada smo već kod kontrole, ovaj zaslon može raditi u dva moda: 8-bitnom i SPI modom. Za 8-bitni mod vam je potrebno 12 digitalnih pinova, dok SPI mod zahtijeva samo 5 pinova za komunikaciju i 4 pina za rezistivni *touch*, pri čemu je ipak sporiji od 8-bitnog moda. Zaslon se može nabaviti u veličinama od 2.8“ ili 3.2“ s rezolucijom od 240X320 RGB piksela.



Slika pr. 10.1. *Prikaz boja na TFT zaslonu*

Možda vam trenutačno pada na pamet da kreirate aplikaciju koja će na vizualno lijep način prikazati temperaturu u prostoriji u kojoj boravite i uz to nacrtati grafikon promjene temperature u vremenu, ali evo nekoliko ideja koje možete iskoristiti za neku nadogradnju ovog početnog projekta:

1. Jednostavna aplikacija za crtanje
2. TFT džepna reklama
3. Log-on konzola

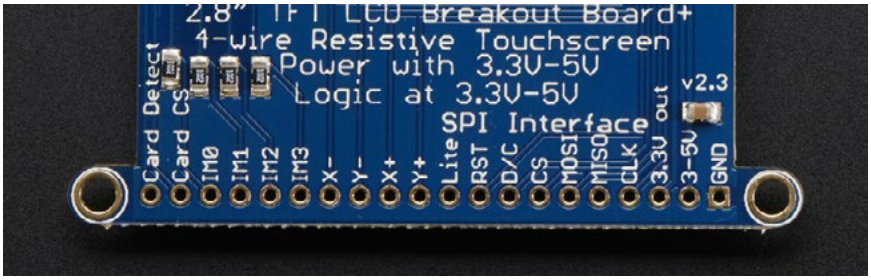
Svaki od spomenutih prijedloga možete nadograditi uvodeći dodatne komponente.

Kada proširite svoje znanje, onda neće biti teško na platformu Arduino spojiti modul Bluetooth ili, što da ne, WiFi modul te napraviti malu vremensku stanicu.

Potrebni elementi za realizaciju projekta navedeni su ispod:

1. ARDUINO UNO
2. Ploča Matador
3. Kablići
4. Adafruit ILI9341 TFT

SPI mod je popularniji te omogućava i korištenje SD kartice, i iako je brži dva do tri puta u odnosu na 8-bitni mod, to ćemo ovu vježbu realizirati upravo koristeći taj specijalni oblik serijske komunikacije. Na taj način smanjit ćemo i mogućnost za pogrešku prilikom ožičenja zaslona.

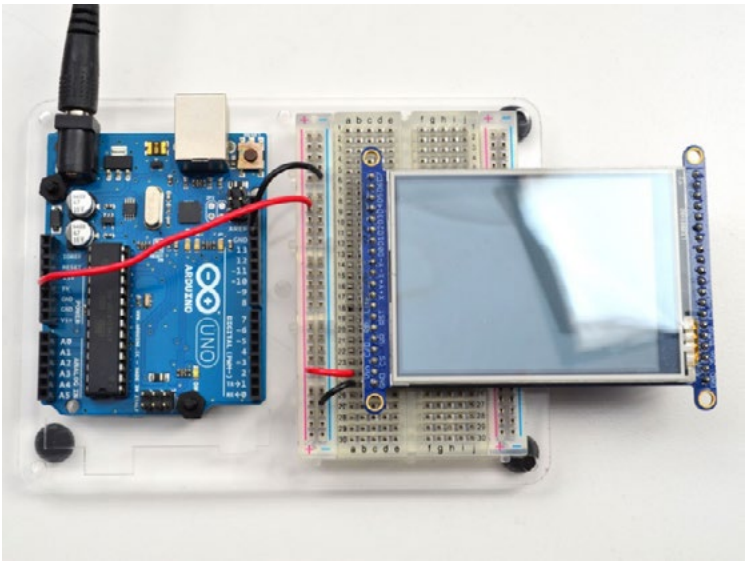


Slika pr. 10.2. *Pinout SPI sučelja na TFT zaslonu*

- **GND** – pin
- **3-5V / Vin** – pin za napon 3 – 5V DC – posjeduje zaštitu od zamjene polariteta!
- **3.3Vout** – Izlaz s regulatora napona 3.3V
- **CLK** – ulazni SPI clock pin
- **MISO** – skraćeno od *Microcontroller In Serial Out* pin, obično se koristi za SD karticu ili za debugiranje, nije potreban za upotrebu TFT zaslona u modu *write-only*
- **MOSI** – skraćeno od SPI *Microcontroller Out Serial In* pin, koristi se za slanje podataka s mikrokontrolera na SD karticu ii li TFT zaslon
- **CS** – TFT SPI chip select pin
- **D/C** – TFT SPI data ili *command selector* pin
- **RST** – TFT reset pin. Spojiti ga s masom da resetirate TFT
- **Lite** – PWM ulaz za kontrolu pozadinskog osvjetljenja
- **IM3 IM2 IM1 IM0** – jumperi za odabir moda
- **Card CS / CCS** – SD card chip select, koriste se za čitanje s SD kartice
- **Card Detect / CD** – SD card detect pin

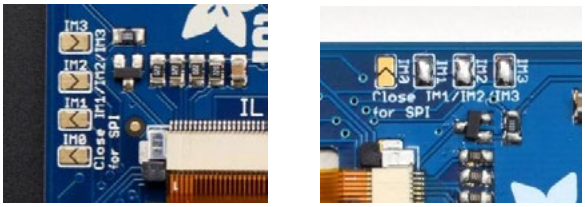
Ako želite koristiti rezistivni *touch*, pinovi **Y+ X+ Y- X-** mogu se spojiti na analogne ulaze za određivanje položaja točke dodira.

Prvi korak u realizaciji ovog projekta predstavlja test zaslona. Naime potrebno je spojiti 3 i 5V pinove na 5V, a GND pinove na GND pin Arduina, kako je to prikazano na slici ispod.



Slika pr. 10.3. *Test TFT-a*

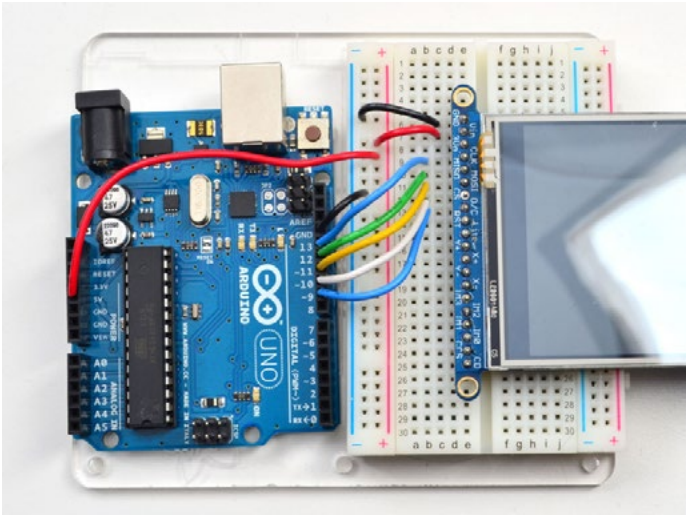
Ako se uključi pozadinsko osvjetljenje, korak smo bliže uspješnoj realizaciji projekta. Nadalje je potrebno zaslon staviti u SPI mod. Na slici ispod prikazani su *jumperi* IMx, koje je potrebno kratko spojiti tinol žicom i lemilicom. Dakle, pinove IM1,IM2 i IM3 potrebno je kratko spojiti, a pin IM0 ostaviti u stanju kakvom jeste.



Slika pr. 10.4. *Jumperi IMx*

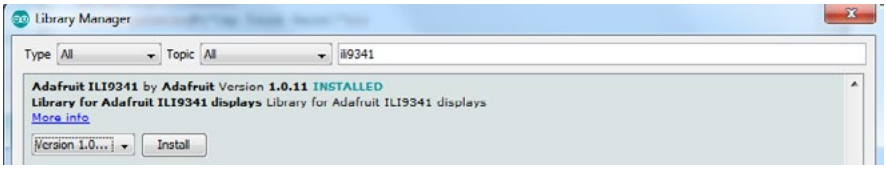
Sada je još potrebno spojiti pet kablića za SPI komunikaciju.

- CLK -> Digital 13
- MISO -> Digital 12
- MOSI -> 11
- CS -> Digital 10
- D/C -> Digital 9



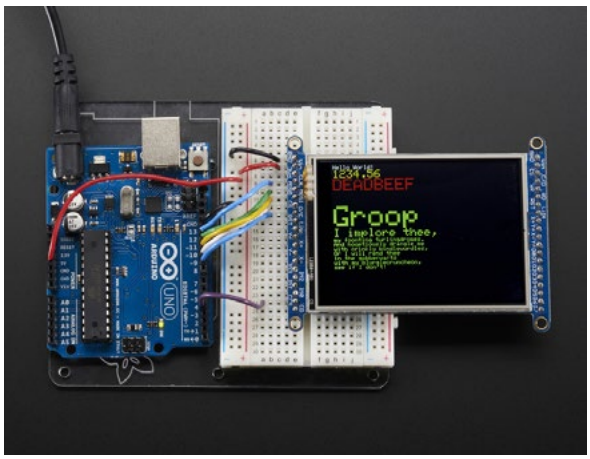
Slika pr. 10.5. *Shema spajanja TFT-a*

I ovaj put instalirat ćemo knjižnicu koja će nam olakšati kreiranje naše aplikacije. Na isti način kao u prethodnom projektu potrebno je koristeći Library Manager dodati knjižnicu Adfruit ILI9341 TFT. Ako vam prije instaliranja knjižnice IDE naglasi da su potrebne još neke od knjižnica, budite slobodni i potvrdite da ih želite instalirati.



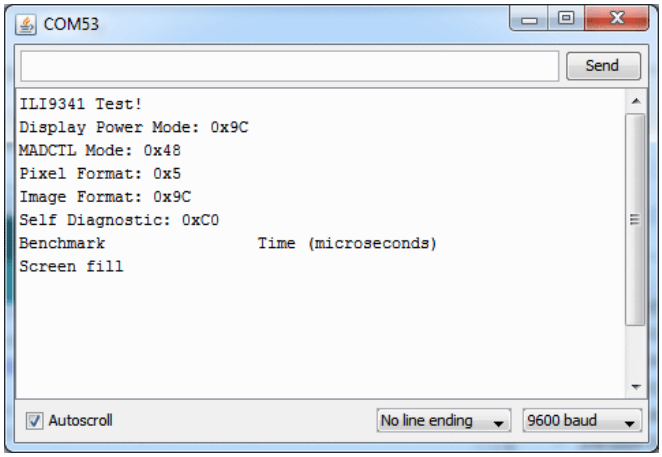
Slika pr. 10.6. *Knjižnica ILI9341*

Nakon instalacije knjižnice i reseta Arduino IDE-a potrebno je u primjerima naći i učitati aplikaciju **graphictest**. Rezultat bi trebao izgledati kao na slici ispod.



Slika pr. 10.7. *Test TFT app*

Ako niste uspjeli iz prve dobiti ovakav prikaz na Vašem TFT-u, potrebno je otvoriti aplikaciju Serial Monitor i provjeriti jeste li dobili *feedback* kao na slici ispod, ako to nije slučaj, ne zaboraviti provjeriti ožičenje.



***Slika pr**ž. **10.8.** Test TFT u Serial Monitor app*

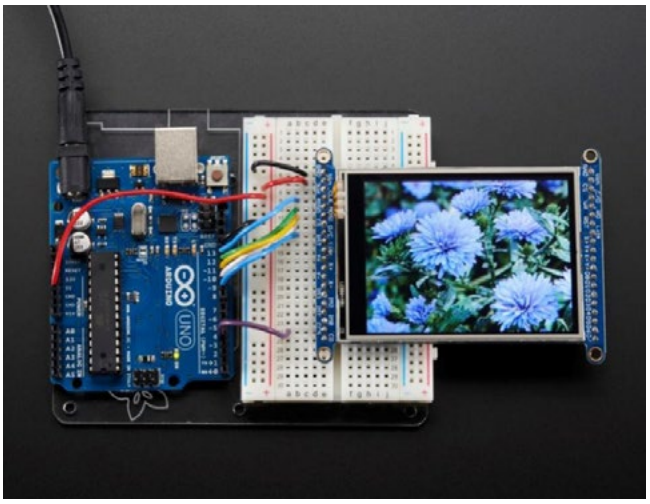
Kako je na TFT modulu montirana i SD kartica, to nam otvara mogućnost čitanja slika koje su spremljene u formatu 24-bit BMP i nisu veće od 240x230 piksela. Prije je potrebno instalirati još jednu knjižnicu kroz Library Manager i to **imageread.**

***Slika pr. 10.9.** Knjižnica imageread*

Nakon instalacije knjižnice potrebno je spojiti **CCS** pin na **Digital4** pin mikrokontrolera, potom naći i pripremiti sliku u preporučenom formatu te ju potom prebaciti na SD karticu. Imena dosjea moraju imati manje od 8 karaktera, a pozivamo ih na sljedeći način:

bmpDraw(bmpfilename, x, y);

Ako ste dobro pratili uputstvo, rezultat bi trebao izgledati kao na slici ispod.



***Slika pr. 10.10.** SD bitmap read*

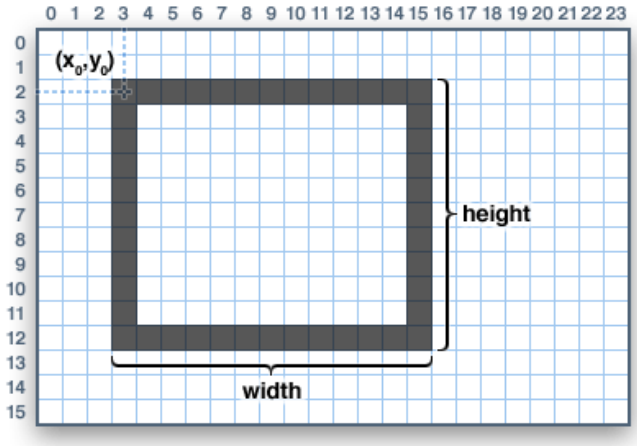
I za sami kraj *gadget* koji koristimo svaki dan i zbog kojega nas dosta ne zna više koristiti onaj stari telefon s krugom i brojevima. Iako ovaj model TFT-a može imati i kapacitivni *touchscreen*, zbog niže cijene puno se češće sreće rezistivni *touch*. Potrebno je prvo instalirati knjižnicu **adafruit touchscreen** te ožičiti Arduino na sljedeći način.

- Y+ to Arduino **A2**
- X+ to Arduino **D9** (Same as **D/C**)
- Y- to Arduino **D8**
- X- to Arduino **A3**

Kod Arduino:

```
#include <stdint.h>
#include „TouchScreen.h“

#define YP A2  // Analogni pin 2!
#define XM A3  // Analogni pin 3!
#define YM 8   // digital pin 8
#define XP 9   // digital pin 9

// Za bolje očitaje potrebno je izmjeriti vrijednost otpora između X+
// i X- koristeći Multimetar
// Za primjer je navedena vrijednost 300 Ohma
// Kreiranje nove instance klase ts
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

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

void loop(void) {
  // point objekt “čuva” vrijednosti koordinata

  TSPoint p = ts.getPoint();

  // za svaki pritisak veći od 0 ćemo reći da je ‚valid‘

  if (p.z > ts.pressureThreshhold) {
     Serial.print(„X = „); Serial.print(p.x);
     Serial.print(„\tY = „); Serial.print(p.y);
     Serial.print(„\tPressure = „); Serial.println(p.z);
  }

  delay(100);
}
```

Ovaj dio koda ne trebate promatrati kao završen projekt. Za kraj ovog projekta pokušat ćemo vam dati prijedlog kako da nadogradite posljednji kod te realizirate jednostavnu aplikaciju koja će sve oduševiti.



Slika pr. 10.11. *Organizacija koordinatnog sustava na TFT-u*

Koristeći sliku iznad i funkcije za crtanje jednostavnih geometrijskih oblika iz GFX knjižnice možete pokušati napraviti dva *buttona* na vašem TFT zaslonu.

```
void drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);

void fillRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);
```

Vidi se da ove dvije navedene funkcije iznad imaju po pet argumenata, te se može napraviti poveznica sa slikom iznad: prva dva argumenta predstavljaju početnu poziciju pravokutnika, druga dva predstavljaju duljinu i širinu stranica pravokutnika, a zadnji argument definira koja je boja linije koja tvori pravokutnik na TFT zaslonu. Druga funkcija ima iste argumente, ali ona ispunjava pravokutnik definiranom bojom.

Ako se detektira pritisak na *touch* u polju gdje se nalazi nacrtani pravokutnik, promijenite mu pozadinsku boju. Kada ovo uspijete, samo nebo je granica. Samo naprijed.

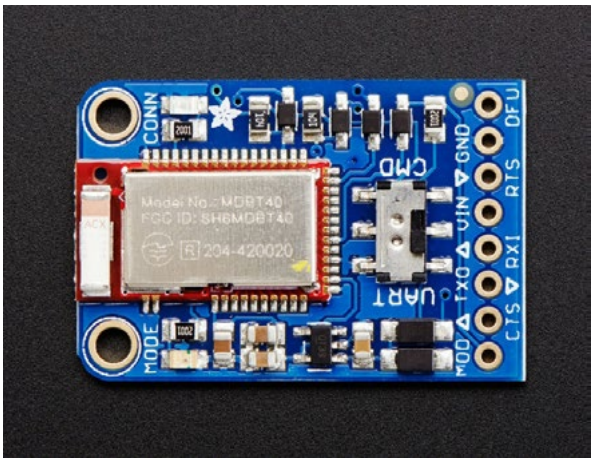
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- opisuje svojstva TFT LCD-a s ekranom osjetljivim na dodir - prepoznaje razliku između modova kod zaslona - klasificira pinove prema 8-bitnom i SPI modu - navodi ulogu pinova SPI sučelja na TFT zaslonu - po potrebi se koristi Serial Monitorom za učitavanje rezultata
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- spaja pinove mikrokontrolera prema shemi spoja - izvršava testiranje zaslona prema shemi spajanja ili smjernicama nastavnika - povezuje preostale komponente u projektu prema shemi spoja
- kreira programski kod za daljinsko upravljanje	- programira Arduino za upotrebu TFT LCD zaslona - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka -	- koristeći Libray Manager dodaje knjižnicu Adfruit ILI9341 TFT - pronalazi aplikaciju graphictest radi učitavanja - instalira imageread knjižnicu kroz Library Manager - po potrebi instalira knjižnicu adafruit touchscreen - poziva knjižnice unutar programskog koda
- verificira i izvršava program	- koristi aplikaciju Serial Monitor radi testiranja zaslona - izvršava testiranje zaslona prema shemi spajanja ili smjernicama nastavnika - analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika
- testira projektni zadatak	- učitava kod na platformu Arduino - analizira projektni zadatak - primjenjuje TFT LCD zaslon u drugim projektnim idejama - Razvija vlastite ideje za primjenu TFT zaslona - koristi TFT zaslon za kreiranje *buttona* i slično

Prostor za vježbu:

```
 void setup() {
}
void loop() {
}
```

Projekt 11.
BLE Bluetooth Low Energy

Vjerojatno ćete u jednome trenutku željeti na svoju aplikaciju dodati i modul Bluetooth Low Energy. Vjerojatno znate da su ovi BLE moduli sveprisutni u našim pametnim uređajima, prijenosnim računalima pa čak i u automobilima. Prvobitni naziv BLE modula je Bluefruit LE UART Friend, pri čemu pod UART (*Universal asynchronous receiver-transmitter*) podrazumijevamo protokol koji mi pojednostavljeno nazivamo serijskom komunikacijom. S pomoću ovog modula jednostavno je dodati BT opciju u aplikaciju gdje god postoji softverski ili hardverski serijski port. Dakle može se spojiti bilo na Arduino ili neku drugu obitelj mikrokontrolera ili jednostavno na FTDI kabel za *debugging* i testiranje. Ovaj mali multifunkcionalni modul može dosta toga, ali za većinu prosječnih korisnika dovoljna je opcija standardnoga konekcijskog profila Nordic UART Rx/Tx. Tada se BLE modul ponaša kao komunikacijski tunel ili kanal koji može slati i primati podatke s aplikacije iOS ili Android. Moguće je koristiti pripadajuću aplikaciju koju možete skinuti s App Store ili s Google Playa.



Slika pr. 11.1. *Izgled BLE modula*

Slika pr. 11.2. *Bluefruit Connect app*

Potrebni elementi za realizaciju projekta su:

1. ARDUINO UNO
2. Ploča Matador
3. Kablići
4. BlueFruit Android/iOS app
5. Modul Bluefruit LE UART Friend

Ovaj mali modul može puno više od pukog bežičnog slanja stringova; on posjeduje jednostavan AT set komandi s pomoću kojega možemo kontrolirati ponašanje, pa čak i mijenjati način na koji je vidljiv drugim BT uređajima. AT komandni set može se iskoristiti i za potraživanje informacija o temperaturi, naponu baterije, provjeru MAC adresa i slično, dakle koristimo samo mali dio potencijala ovog malog, ali moćnog uređaja.

Prvi zadatak je da preuzmete aplikaciju Bluefruit na svoje uređaje Apple ili Android. Unutar aplikacije moguće je odabrati četiri glavna moda rada i to:

- Color Picker
- Accelerometer Data
- GPS Data
- Control game pad

Za početak ćemo u svom projektu koristiti ovaj zadnji mod, a Vi kako budete napredovali sa znanjem, možete iskoristiti neki od drugih modova.

Da biste razumjeli zašto ovaj modul ima ovako veliki broj mogućnosti korištenja, kada pogledate njegovu tehničku specifikaciju, shvatit ćete da je on programabilni uređaj baziran na nRF51822 ultra low power SoC (*System on a Chip*). Bez obzira na to što se jedan ovakav uređaj koristi tek kao *tuneling* uređaj za prosljeđivanje podatak s nekog smart uređaja, to ne umanjuje težinu i važnost ovog projekta. Dio osnovnih tehničkih specifikacija nRF51822 prikazan je ispod:

- ARM Cortex M0 core - 16MHz
- Memorija 256KB flash
- 32KB SRAM
- 2.4 GHz Transceiver

Prije no opišemo *pinout*, potrebno je navesti da se s druge strane modula nalazi priprema za opcijski baterijski priključak, te opcijsko mjesto za montažu oscilatora 32 KHz. Osim ovoga na pozadini modula su i sljedeći pinovi:

SWC: SWC clock pin, 3v logička razina – hakerski pinovi

SWD: SWD data pin, 3v logička razina – hakerski pinovi

3Vo: Izlaz sa 3V naponskog regulatora

FCR: Pin za vraćanje na tvorničke postavke

Pored ovih specijalnih pinova na BLE modulu su i standardni pinovi koje ćete koristiti u ovom projektu:

Pinovi za napajanje

- **VIN**: pin za napajanje naponom od 3.3 do 16V
- **GND**: zajednički pin za napajanje i logiku

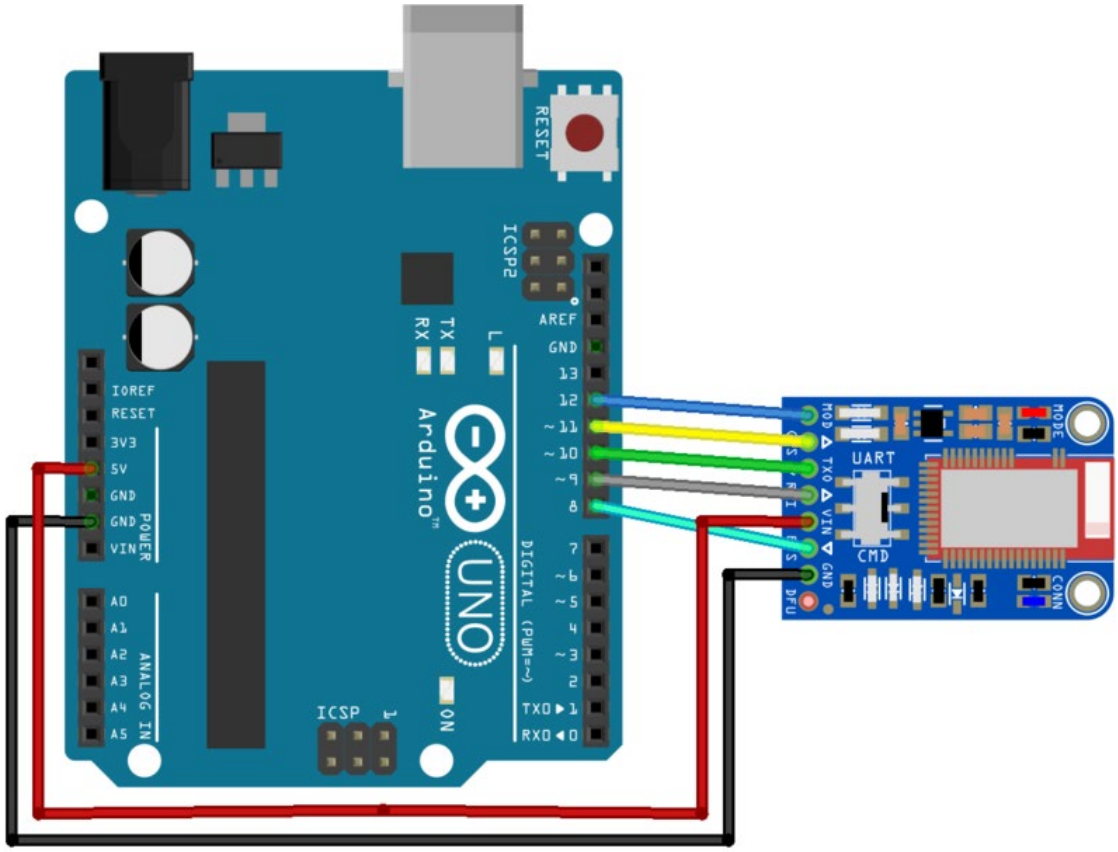
UART pinovi

- **TXO** – UART Transmit pin, pin za slanje podataka (Bluefruit LE --> MCU) 3.3V logička razina
- **RXI** – UART Receive pin, pin za primanje podataka (MCU --> Bluefruit LE) 3-5V logička razina
- **CTS** – Ovaj je pin po *defaultu* u stanju logičke nule. Da biste osigurali transfer podataka, potrebno je ovaj pin staviti u stanje logičke nule, 3-5V logička razina
- **RTS** – Pin Ready to Send za kontrolu protoka podataka

Ekstra pinovi

- **MOD**: Selekcija moda rada. Bluefruit ima dva moda rada: komandni mod ili data mod
- **DFU**: Mod za ažuriranje firmwarea OTA (Over the Air)

Iako se BLE modul može spojiti na bilo koji mikrokontroler koji ima bilo hardverski ili softverski UART komunikacijski kanal, u ovom projektu iskoristit ćemo knjižnicu SoftwareSerial i BLE modul spojiti na taj softverski UART port.

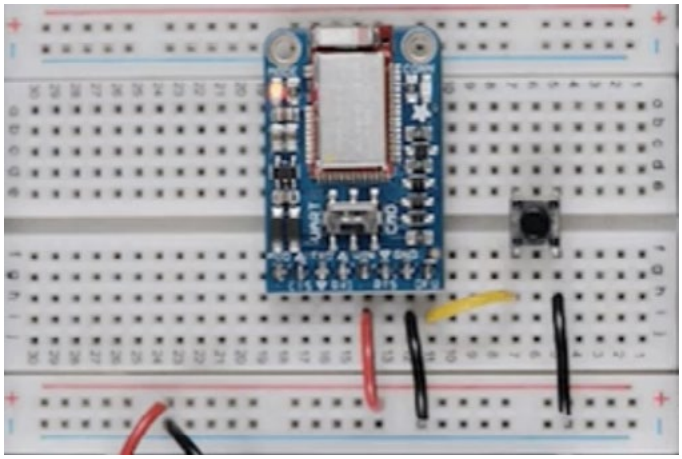


Slika pr. 11.3. *Bluefruit shema spajanja sa softverskim UART portom*

Dakle vodeći se slikom iznad potrebno je pinove BLE modula spojiti s Arduino UNO na sljedeći način:

- **MOD** to **Pin 12**
- **CTS** to **Pin 11**
- **TXO** to **Pin 10**
- **RXI** to **Pin 9**
- **VIN** to **5V**
- **RTS** to **Pin 8**
- **GND** to **GND**

Prije no počnemo s primjerima, potrebno je da upoznate barem jedan od načina vraćanja na tvorničke postavke. Na tvorničke postavke BLE trebate vratiti ako ste ga, recimo, pogrešno konfigurirali. Koristit ćemo možda najjednostavniji način, pogotovo ako je na BLE modul postavljen DFU taster. Naime dovoljno je taster držati duže od pet sekundi, dok je BLE pod naponom. Pri tome će prvo zablinkati crvena LED pa potom plava LED i to je znak da je BLE resetiran. Ako DFU taster nije montiran na BLE modul, potrebno je ožičiti BLE kao na slici ispod i uraditi proceduru resetiranja kao i s montiranim DFU tasterom.



Slika pr. 11.4. *Shema Bluefruit Factory reset*

Ponovno ćemo koristeći već poznatu proceduru i instalirati knjižnicu Bluefruit koristeći Library Manager. Kako je knjižnica univerzalno napisana za široki spektar uređaja, to je potrebno prije samog korištenja primjera iz knjižnice podesiti knjižnicu za BLE modul.

Prije svega pregledat ćemo pripadajući *header* file *BluefruitConfig.h*, u kojem možemo mijenjati neke od standardnih postavki u aplikaciji, naprimjer ako vam je pin 12 zauzet, možete uraditi sljedeće:

```
// ZAJEDNIČKA UART PODEŠENJA
// ------------------------------------------------------------------------
-----
// opcijsko podešavanje Mode pina, preporučuje se, ali nije obvezno
// ------------------------------------------------------------------------
-----
#define BLUEFRUIT_UART_MODE_PIN       3    // postavite na -1 ako se pin
ne koristi
```

Možete u *BluefruitConfig.h* pin 3 proglasiti MOD pinom u vašoj aplikaciji, ispod je cijeli originalni *header* dosje:

```
// STANDARDNA PODEŠENJA
// ------------------------------------------------------------------------
-------------------------
// podešenja za SW UART, HW UART i SPI
 // ------------------------------------------------------------------------
 -------------------------
#define BUFSIZE                         128   // veličina buffera za do-
lazeće podatke
#define VERBOSE_MODE                    true  // 'true' omogućava debug


// PODEŠENJA SOFTVERSKOG UART-A
// ------------------------------------------------------------------------
-------------------------
// sljedeći makroi deklariraju koji će pinovi biti korišteni za 'SW'
serijsku komunikaciju.
//
// ------------------------------------------------------------------------
-------------------------
#define BLUEFRUIT_SWUART_RXD_PIN        9    // rx sw serial pin
#define BLUEFRUIT_SWUART_TXD_PIN        10   // tx sw serila pin
#define BLUEFRUIT_UART_CTS_PIN          11   // cts sw serial pin
#define BLUEFRUIT_UART_RTS_PIN         -1   // Opciono, setovati na  -1
ako se ne koristi


// PODEŠENJA HARDWARESKOG UART-A
// ------------------------------------------------------------------------
-------------------------
// sljedeći makroi deklariraju koji će pinovi biti korišteni za 'HW'
serijsku komunikaciju
//
// ------------------------------------------------------------------------
-------------------------
#ifdef Serial1    // rezerviranje Serial1 konstante
  #define BLUEFRUIT_HWSERIAL_NAME      Serial1
#endif


// DIJELJENA UART PODEŠENJA
// ------------------------------------------------------------------------
-------------------------
// opcioni Mode pin
// ------------------------------------------------------------------------
-------------------------
#define BLUEFRUIT_UART_MODE_PIN         12    // postaviti na -1 ako se
ne koristi


// DIJELJENA SPI PODEŠENJA
// ------------------------------------------------------------------------
-------------------------
```

```
// sljedeći makroi deklariraju pinove koji se koriste za HW i SW SPI
komunikaciju.
// -------------------------------------------------------------------
-------------------------
#define BLUEFRUIT_SPI_CS               8
#define BLUEFRUIT_SPI_IRQ              7
#define BLUEFRUIT_SPI_RST               4    // postaviti na -1 ako se
ne koristi

// SOFTVERSKI SPI PODEŠENJA
// -------------------------------------------------------------------
-------------------------
// sljedeći makroi deklariraju pinove koji se koriste za SW SPI komu-
nikaciju.
// -------------------------------------------------------------------
-------------------------
#define BLUEFRUIT_SPI_SCK              13
#define BLUEFRUIT_SPI_MISO             12
#define BLUEFRUIT_SPI_MOSI             11
```

I došli smo konačno do dijela koji nas najviše zanima, naime u biblioteci ima puno primjera, ali ovdje ćemo objasniti samo dva – prvo *Bleuart_cmdmod*, nadam se da se nećete uplašiti možda nekih nepoznatih dijelova koda, primjeri su tu da se iz njih uči nešto novo.

```
This is an example for our nRF51822 based Bluefruit LE modules

 Pick one up today in the adafruit shop!

 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/
#include <Arduino.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#if SOFTWARE_SERIAL_AVAILABLE
  #include <SoftwareSerial.h>
#endif
```

```
/*=========================================================================
=====
    APPLICATION SETTINGS

    FACTORYRESET_ENABLE       reset na fabrička podešenja
     -----------------------------------------------------------------------
-----*/
    #define FACTORYRESET_ENABLE         1
    #define MINIMUM_FIRMWARE_VERSION    "0.6.6"
    #define MODE_LED_BEHAVIOUR          "MODE"
/*=========================================================================
=====*/

// Kreiranje  bluefruit objekta

SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);

// error debugging
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

/**************************************************************************
*****/
/*!
    @brief  Podešavanje HW  BLE modula (ova funkcija poziva se automat-
ski ¸ prilikom startupa)
*/
/**************************************************************************
*****/
void setup(void)
{
  while (!Serial);  // za razvojne ploče Flora & Micro
  delay(500);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Command Mode Example"));
  Serial.println(F("---------------------------------------"));

  /* Inicijalizacija modula */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode &
check wiring?"));
  }
  Serial.println( F("OK!") );

  if ( FACTORYRESET_ENABLE )
  {
```

```
/* reset na fabričke postavke */

    Serial.println(F("Performing a factory reset: "));
    if ( ! ble.factoryReset() ){
      error(F("Couldn't factory reset"));
    }
  }

  /* Onemogući eho opciju sa Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit info */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in
UART mode"));
  Serial.println(F("Then Enter characters to send to Bluefruit"));
  Serial.println();

  ble.verbose(false);  // isključi debug info !

  /* Čekaj konekciju */
  while (! ble.isConnected()) {
      delay(500);
  }

  // LED Activity command je podržana od fw verzije 0.6.6
  if ( ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
  {
    // Promijeni Mode LED Activity
    Serial.println(F("******************************"));
    Serial.println(F("Change LED activity to " MODE_LED_BEHAVIOUR));
    ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEHAVIOUR);
    Serial.println(F("******************************"));
  }
}

/**************************************************************************
*****/
/*!
@brief  Konstantno provjeravamo ima li novih komandi ili odziva.
*/
/**************************************************************************
*****/
void loop(void)
{
  // Provjera user inputa
  char inputs[BUFSIZE+1];

  if ( getUserInput(inputs, BUFSIZE) )
  {
    // Pošalji karaktere ka Bluefruit
    Serial.print("[Send] ");
    Serial.println(inputs);
```

```
    ble.print("AT+BLEUARTTX=");
    ble.println(inputs);

    // Provjeri status odziva
    if (! ble.waitForOK() ) {
      Serial.println(F("Failed to send?"));
    }
  }

  // Provjeri dolazne karaktere s  Bluefruita
  ble.println("AT+BLEUARTRX");
  ble.readline();
  if (strcmp(ble.buffer, "OK") == 0) {
    // ako nema podataka
    return;
  }
  // Imamo neke podatke, u bufferu su
  Serial.print(F("[Recv] "));
  Serial.println(ble.buffer);
  ble.waitForOK();
}

/**************************************************************************
***/
/*!
    @brief  Provjera unosa od korisnika(preko Serial Monitora)
*/
/**************************************************************************
***/
bool getUserInput(char buffer[], uint8_t maxSize)
{
  // timeout  100 milisekundi
  TimeoutTimer timeout(100);

  memset(buffer, 0, maxSize);
  while( (!Serial.available()) && !timeout.expired() ) { delay(1); }

  if ( timeout.expired() ) return false;

  delay(2);
  uint8_t count=0;
  do
  {
    count += Serial.readBytes(buffer+count, maxSize);
    delay(2);
  } while( (count < maxSize) && (Serial.available()) );

  return true;
}
```

Kao prvo neki su komentari u vezi s primjerom ostavljeni u originalnom obliku radi autorskih prava pisca knjižnice. Nije cilj objasniti svaku liniju koda, jer za razumijevanje aplikacije morate samostalno iščitati dokumentaciju ove knjižnice. Vidjet ćete da će to dati odgovore na mnoga pitanja. To je jedini ispravan put, da ne zalutate u bespuće rješenja *copy/paste*.

Jedna nepoznanica, vjerojatno da je prvo što ste primijetili sljedeće:

```
Serial.print(F("[Recv] "));
```

Ovo *F* u osnovi znači sljedeće:

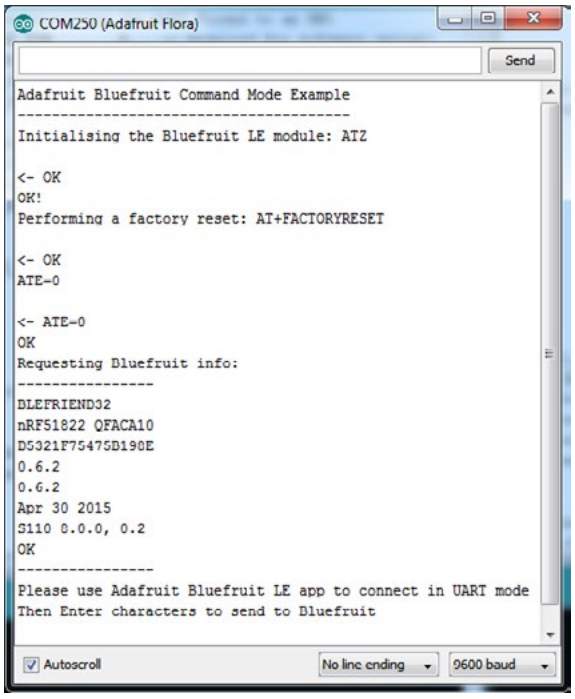
```
Serial.println(F("Ja ne koristim RAM za ovu instrukciju!!!"));

Serial.println(F("Ja sam konstanta u FLASH memoriji!!!"));

Serial.println(F("Pojednostavljeno ja sam na HDD Arduina!!!"));
```

Možda i ovo:

```
if (strcmp(ble.buffer, "OK") == 0)
```

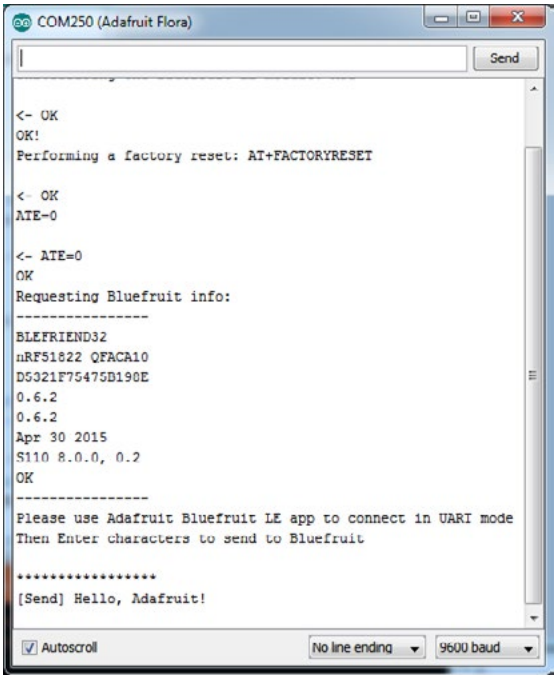
ali samo ako niste imali osnove programiranja, riječ je o instrukciji *string compare*, koja uspoređuje karakter po karakter dva stringa.

Većina ostalih stvari je poznata ili ih je vrlo lako razumjeti. Sada je potrebno vidjeti kako je moguće testirati aplikaciju. Nakon što ste kompajlirali i učitali app u Arduino, potrebno je otvoriti Serial Monitor app, podesiti u donjem desnom kutu BaudRate na 115200. Ako ste dobro ožičili BLE modul, učitali *example* kod s prepravkama, onda biste na Serila Monitoru trebali dobiti sljedeće:

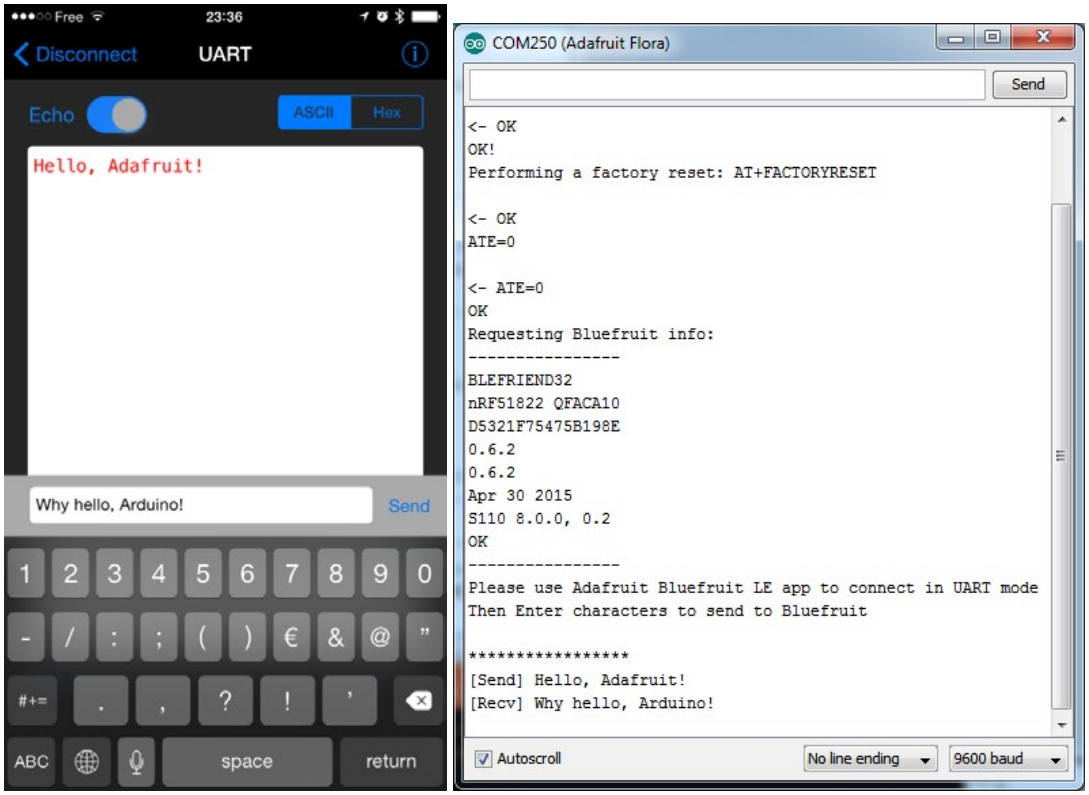


Slika pr. 11.5. Serial Monitor odziv uspješno spojenog i programiranog example projekta

Sada je potrebno pokrenuti instaliranu aplikaciju na telefonu ili nekom drugom uređaju, skenirati i naći Bluefruit LE modul te se povezati u UART modu. Kada ste to uspješno uradili, možete slati tekst ka telefonu izravno iz Serial Monitora.



Slika pr. 11.6. *Slanje stringa ka aplikaciji Bluefruit*



Slika pr. 11.6. *Zaprimanje stringa i odgovor ka Arduinu*

Strojevi komuniciraju, dakle dva različita uređaja, jedan pametni i drugi *embedded* međusobno razmjenjuju stringove, ne znam kako se to vama čini, ali mislim da je dovoljno nekoliko instrukcija *if...else...endif* da zadivimo prijatelje i napravimo neki jednostavni ChatBot ili scenarijem definiran prividno inteligentan razgovor dva stroja, samo korak smo do moda Terminator 😊.

Naravno, ne možemo se zaustaviti ovdje, slijedi primjer aplikacije koju možemo iskoristiti u kombinaciji s nekim od naših prethodnih projekata.

U primjerima nađite primjer *controller* i pregledajte dobro primjer iz priručnika jer su neki segmenti izmijenjeni da bi odgovarali našoj hardverskoj konfiguraciji.

```
This is an example for our nRF51822 based Bluefruit LE modules

 Pick one up today in the adafruit shop!

 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#if SOFTWARE_SERIAL_AVAILABLE
  #include <SoftwareSerial.h>
#endif

/*=========================================================================
=====
    APPLICATION SETTINGS

    FACTORYRESET_ENABLE       reset na fabrička podešenja
     ------------------------------------------------------------------
-----*/
    #define FACTORYRESET_ENABLE         1
    #define MINIMUM_FIRMWARE_VERSION    "0.6.6"
    #define MODE_LED_BEHAVIOUR          "MODE"
/*=========================================================================
=====*/
```

```
// Kreiranje bluefruit objekta

SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);


// error debugging
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// prototipovi funkcija u packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// paket buffer
extern uint8_t packetbuffer[];


/**************************************************************************
*****/
/*!
    @brief  Podešavanje HW  BLE modula (ova funkcija se poziva automat-
ski ¸ prilikom startupa)

*/
/**************************************************************************
*****/
void setup(void)
{
  while (!Serial);  // za razvojne ploče Flora & Micro
  delay(500);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit App Controller Example"));
  Serial.println(F("-----------------------------------------"));

  /* Inicijalizacija modula */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode &
check wiring?"));
  }
  Serial.println( F("OK!") );

  if ( FACTORYRESET_ENABLE )
  {
/* reset na fabričke postavke*/
```

```
    Serial.println(F("Performing a factory reset: "));
    if ( ! ble.factoryReset() ){
      error(F("Couldn't factory reset"));
    }
  }


  /* Onemogući eho opciju sa Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in
Controller mode"));
  Serial.println(F("Then activate/use the sensors, color picker, game
controller, etc!"));
  Serial.println();

  ble.verbose(false);  // debug info is a little annoying after this
point!

  /* Čekaj konekciju */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("******************************"));

  // LED Activity command je podržana od fw verzije 0.6.6
  if ( ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
  {
    // Promijeni Mode LED Activity
    Serial.println(F("Change LED activity to " MODE_LED_BEHAVIOUR));
    ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEHAVIOUR);
  }

  // postavi Bluefruit u DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("******************************"));

}

/**************************************************************************
*****/
/*!
    @brief  Konstantno provjeravamo ima li novih komandi ili odziva
*/
/**************************************************************************
*****/
```

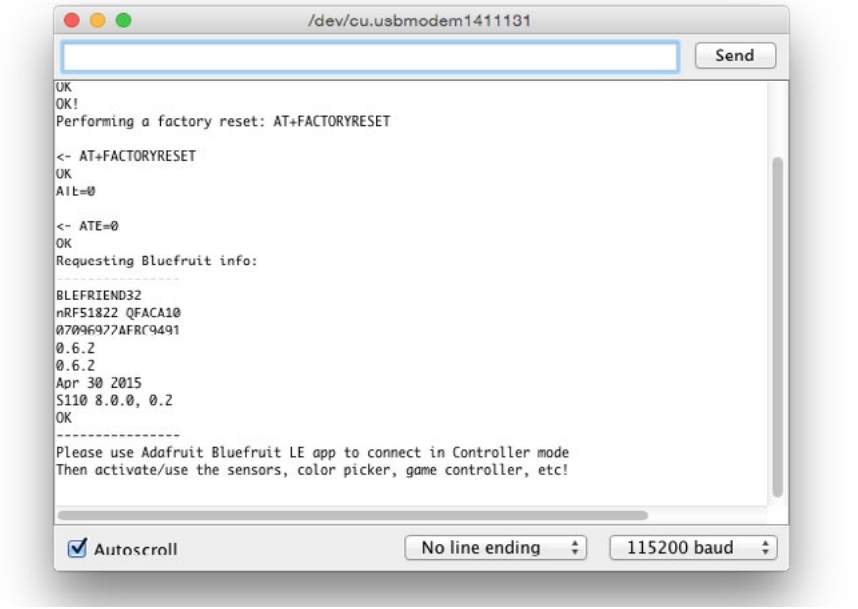
```
void loop(void)
{
 /* Čekanje novih podataka */
  uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
  if (len == 0) return;

  /* Paket zaprimljen! */
  // printHex(packetbuffer, len);

  // Boja
  if (packetbuffer[1] == 'C') {
    uint8_t red = packetbuffer[2];
    uint8_t green = packetbuffer[3];
    uint8_t blue = packetbuffer[4];
    Serial.print ("RGB #");
    if (red < 0x10) Serial.print("0");
    Serial.print(red, HEX);
    if (green < 0x10) Serial.print("0");
    Serial.print(green, HEX);
    if (blue < 0x10) Serial.print("0");
    Serial.println(blue, HEX);
  }
  // Gumb
  if (packetbuffer[1] == 'B') {
    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';
    Serial.print ("Button "); Serial.print(buttnum);
    if (pressed) {
      Serial.println(" pressed");
    } else {
      Serial.println(" released");
    }
  }

  // GPS Lokacija
  if (packetbuffer[1] == 'L') {
    float lat, lon, alt;
    lat = parsefloat(packetbuffer+2);
    lon = parsefloat(packetbuffer+6);
    alt = parsefloat(packetbuffer+10);
    Serial.print("GPS Location\t");
    Serial.print("Lat: "); Serial.print(lat, 4); // 4 decimale!
    Serial.print('\t');
    Serial.print("Lon: "); Serial.print(lon, 4); // 4 decimale!
    Serial.print('\t');
    Serial.print(alt, 4); Serial.println(" meters");
  }

  // Accelerometar
  if (packetbuffer[1] == 'A') {
    float x, y, z;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    Serial.print("Accel\t");
```

```
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.println();
  }

  // Magnetometar
  if (packetbuffer[1] == 'M') {
    float x, y, z;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    Serial.print("Mag\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.println();
  }

  // Žiroskope
  if (packetbuffer[1] == 'G') {
    float x, y, z;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    Serial.print("Gyro\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.println();
  }

  // Kvaternioni
  if (packetbuffer[1] == 'Q') {
    float x, y, z, w;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    w = parsefloat(packetbuffer+14);
    Serial.print("Quat\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.print('\t');
    Serial.print(w); Serial.println();
  }
}
```

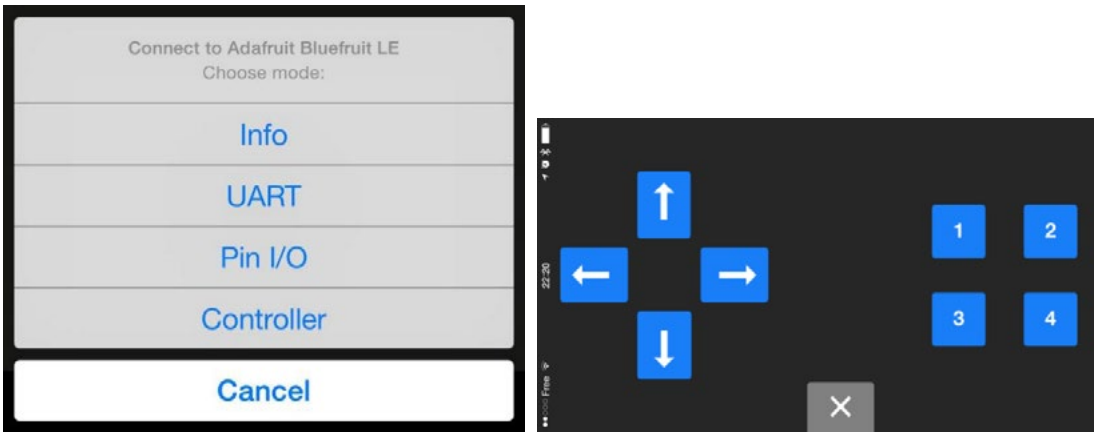
Ovaj projekt sadrži i dodatni dosje *packetParser.cpp*, koji sadrži API za parsiranje podataka koji dolaze s aplikacije Bluefruit.

Gore navedeni primjer omogućava nam da pretvorimo uređaj iOS ili Android u daljinski kontroler ili eksterni izvor podataka o GPS lokaciji ili podatcima sa senzora za detekciju ubrzanja integriranog u telefon ili tablet.

Ako ste BLE ožičili bez grešaka te učitali gore navedeni primjer, trebali biste u Serial monitoru dobiti sljedeći odziv.



Slika pr. 11.7. *Uspješan odziv controler.ino u Serial Monitoru*



Slika pr. 11.8. *Uspješan odziv controler.ino u Serial Monitoru*

Pritiskom dugmadi na aplikaciji Bluefruit u modu Control Pad registrirat će se na aplikaciji Serial Monitor s ID brojem dugmeta koji je korišten:

```
Button 8 pressed
Button 8 released
Button 3 pressed
Button 3 released
```

Mislim da je sada jasno da uz pomoć provjere podataka koji dolaze s BLE aplikacije možemo integrirati u svaki dosadašnji projekt, sada je samo potrebno prepustiti se svojoj mašti.

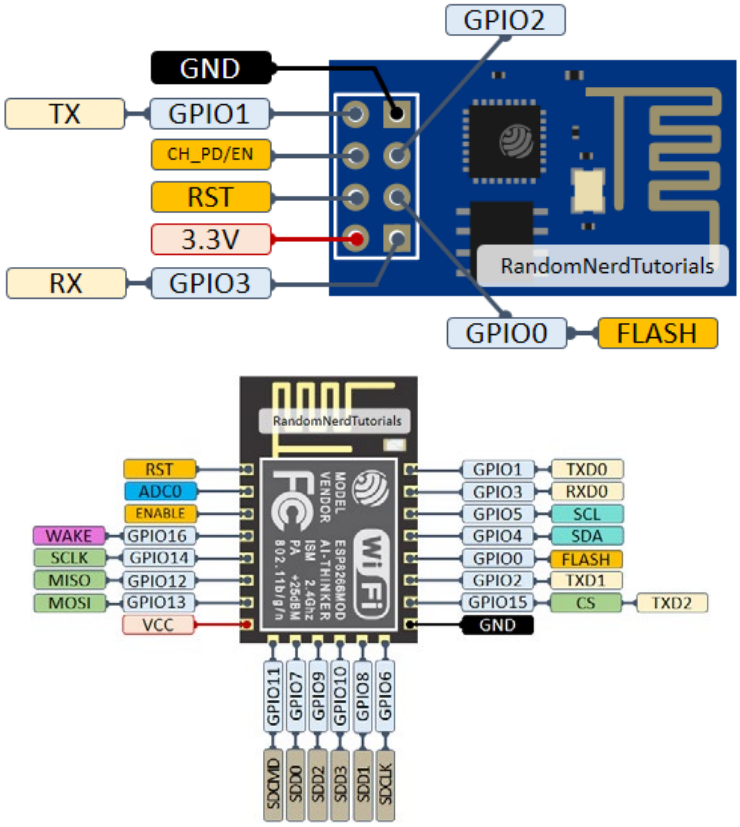
ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- opisuje svojstva modula BLE Bluetooth Low Energy - navodi primjere primjene BLE modula - nabraja modove unutar aplikacije Bluefruit Connect - koristi aplikaciju Bluefruit Connect za povezivanje
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje pinove BLE modula s Arduino UNO prema shemi spoja - izvršava testiranje BLE modula - povezuje preostale komponente u projektu prema shemi spoja
- kreira programski kod za upravljanje koristeći BLE Bluetooth Low Energy prema uputama	- preuzima, instalira i primjenjuje odgovarajuću aplikaciju Bluefruit Connect u zavisnosti od operativnog sustava (iOS ili Android) - konfigurira BLE modul - vraća BLE modul na tvorničke postavke ako je pogrešno konfiguriran - koristi gotove primjere programskog koda kako bi naučio/-la nešto novo - rekonstruira programski kod prema vlastitim idejama
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka	- instalira knjižnicu SoftwareSerial koristeći Library Manager - upravlja BluefruitConfig.h vršeći potrebne izmjene - poziva knjižnice unutar programskog koda
- verificira i izvršava program	- koristi aplikaciju Serial Monitor radi testiranja aplikacije - analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika
- testira projektni zadatak	- učitava kod na platformu Arduino - primjenjuje Serial Monitor za testiranje aplikacije - povezuje mobitel ili neki drugi uređaj na modul Bluefruit LE - testira slanje teksta putem Bluetootha - uočava ulogu dodatnog dosjea packetParser.cpp - analizira projektni zadatak - razvija vlastite ideje za primjenu BLE Bluetooth Low Energy

Prostor za vježbu:

```
 void setup() {

}
void loop() {

}
```

Projekt 12.
ESP8266 IoT

IoT ili Internet of Things počeo se rapidno razvijati pojavom ESP8266 WiFi SoC-a s potpuno integriranim TCP/IP protokolom, to je *low-cost* WiFi modul sposoban da radi hosting mrežne aplikacije ili osigura sve mrežne funkcije mikrokontroleru koji inače nema mogućnosti pristupa WiFi mreži. Modul dolazi s programiranim firmwareom za kontrolu s pomoću AT komandnog seta, te se jednostavno može priključiti na Arduino. Ogroman je spektar aplikacija koje se mogu realizirati s ovim modulom, a mi ćemo se bazirati na nešto što je u posljednje vrijeme postalo jako popularno jer svi žele živjeti u okruženju Smart Home. Dakle krenut ćemo u ovaj projekt s jednostavnim zadatkom daljinske kontrole uređaja s pomoću WiFi mreže. Treba napomenuti da je moguće projekt realizirati i s modulom ESP8266 SMD, čiji je pinout prikazan na slici ispod.



Slika pr. 12.1. *ESP8266 Pinout i ESP8266 smd pinout*

Po općeprihvaćenoj definiciji tehnologija Smart Home koristi uređaje tipa senzora ili uređaja koji su spojeni na IoT te se mogu daljinski nadzirati, kontrolirati ili nude opciju pristupa za osiguranje servisa prema zahtjevima korisnika.

Krenut ćemo od postavke problema, i to na način da ćemo postaviti sebi zadatak da kreiramo jednostavni mrežno bazirani kontrolni panel za upravljanje uređajima koji su spojeni na lokalnu WiFi

mrežu. Ovo odmah postaje multidisciplinarni projekt jer sad pored *embedded* aplikacije imamo zadatak kreiranja mrežne aplikacije, te moramo osigurati način da te dvije aplikacije komuniciraju.

Za izradu mrežne aplikacije potrebno je koristiti sljedeće tehnologije:

- HTML
- CSS
- Jquery

Mrežna stranica će slati zahtjeve ka ESP8266, koji će raditi kao mrežni server spojen preko UART komunikacijskog kanala s Arduinom. U zavisnosti od podataka koji dolaze na Arduino dešavat će se korespondentne akcije na njegovim izlazima, tipa uključenje ili isključenje nekog od potrošača.

Potrebni elementi za realizaciju projekta navedeni su ispod:

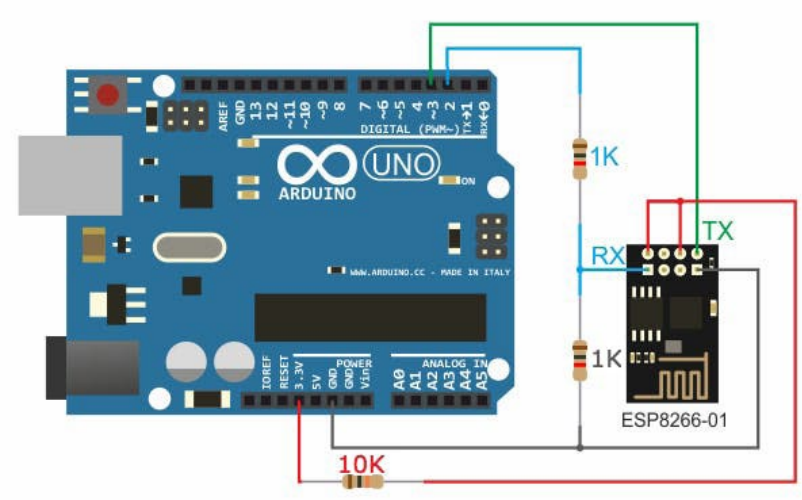
1. ARDUINO UNO
2. Ploča Matador
3. Kablići
4. 1x10K otpornik
5. 2x 1K otpornik
6. Modul ESP8266

Detaljni pinout WiFi modula moramo navesti s posebnim naglaskom da modul nema integriran naponski regulatora te da se mora spojiti na napon od 3.3V.

- **VCC**: +3.3V, spajanje na 5V izvor oštetit će modul
- **GND**: zajednički pin
- **CH_PD**: Chip enable pin, Potrebno je pin spojiti 3.3V da bi se modul boota-o
- **RST**: Chip Reset pin. Spajanje na GND resetira modul
- **GPIO0, GPIO2**: Pinovi opće namjene
- **Tx**: pin za slanje podataka
- **Rx**: pin za primanje podataka

Kao što smo već naveli, modul ESP8266 komunicira s modulom Arduino koristeći serijsku komunikaciju. To znači da bismo trebali iskoristiti hardverske pinove UART modula pin 0 i pin 1, ali na taj način nismo u mogućnosti koristiti aplikaciju Serial monitor za *debugging* softvera i

troubleshooting. Ovaj ćemo problem riješiti korištenjem knjižnice SoftwareSerial te WiFi modul spojiti prema shemi spoja kao na slici ispod.

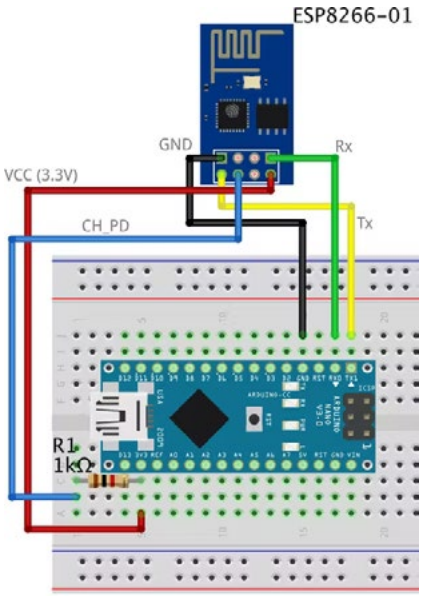


Slika 12.2. *Shema spoja ESP8266 i Arduino Uno*

Knjižnica SoftwareSerial radi pouzdano dokle god su brzine prijenosa podataka podešene na 19200 i manje. ESP8266 tvornički dolazi podešen za rad na brzini prijenosa podataka od 115200, stoga je potrebno rekonfigurirati modul koristeći AT komande.

Prije rekonfiguracije potrebno je u Arduino učitati skicu BaseMinumum, s putanje *Examples->Basic->BaseMiinumum.*

NAPOMENA:
Za potrebe rekonfiguracije modula EPS8266 potrebno ga je spojiti na hardverski UART modul Arduina, dakle RX pin ESP modula spojiti na TX pin Arduina, TX pin ESP-a spojiti na RX pin Arduina. Primjer koji je dan kao vodilja urađen je s drugom razvojnom pločom Arduino, ali princip spajanja ostaje isti, bilo da se radi o modulu ESP8266 -01 ili o modulu ESP8266 SMD.



Primjer spajanja ESP modula za rekonfiguraciju s pomoću AT komandi

AT komandni set je standardni način komunikacije s modemima, BT i GPS modulima, AT je kratica od riječi „ATention“ i predstavlja prefiks u svim naredbama, pri čemu modul zna da poslije dolaska ove ključne riječi ide naredba. Na slici ispod predstavljene su neke od osnovnih AT naredbi:

No.	Command	Example	Parameters	Description	returns
1	AT	AT	Nothing	Check if the module is working properly.	OK
2	AT+RST	AT+RST	Nothing	Resets the Module.	OK
3	AT+CWMODE= <MODE>	AT+CWMODE=1	1 = Station 2 = AP 3 = Both	sets the Wifi mode	OK
4	AT+CIOBAUD= <baud> OR AT+UART_DEF= <baud>,8,1,0,0	AT+CIOBAUD=9600 OR AT+UART_DEF=9600,8,1,0,0	baud rate	changes the module communication speed.	OK
5	AT+CWJAP= <SSID>,<PASS>	AT+CWJAP="meme", "69696969696"	SSID = Wifi Network name PASS = Wifi Password	Connects to a Wifi Network	WIFI CONNECTED WIFI GOT IP
6	AT+CIFSR	AT+CIFSR	Nothing	Tells you the module IP address and the MAC Address	IP Address Mac Address
7	AT+CIPSERVER= <MODE>,<PORT>	AT+CIPSERVER=1,80	MODE = 0 to close the server, 1 to open the server. PORT = port number	Sets the module as a server.	OK

Slika pr. 12.3. *AT komande*

Potrebno je otvoriti aplikaciju SerialMonitor i u njoj u donjem desnom kutu podesiti brzinu komunikacije na 115200 i odabrati opciju **Both NL & CR.**

Naredni je korak provjera komunikacije s ESP modulom, primjer uspješno ostvarene komunikacije prikazan je na slici ispod.

Slika pr. 12.4. AT odziv s modula ESP8266

U zavisnosti od toga koji je firmware došao učitan na vašem modulu ESP8266-01 ili ESP8266 SMD promjenu brzine protoka podataka možete probati s jednom od ove dvije naredbe:

```
command AT+UART_DEF=<baud>,8,1,0,0

AT+CIOBAUD=<baud>.
```

Ako je odziv na jednu od ove dvije naredbe **"OK", vaš ESP modul nadalje komunicira na brzini od 9600 bit/s. Ne zaboravite u svom Serial monitoru odabrati u donjem desnom kutu 9600**!

Nadalje je potrebno modul ESP8266 spojiti na vašu lokalnu WiFi mrežu, pri čemu je potrebno upisati u AT komandu naziv lokalne mreže kao i šifru za nju.

```
AT+CWJAP= "imemojeWiFimreze","paswordmojeWiFimreze"
```

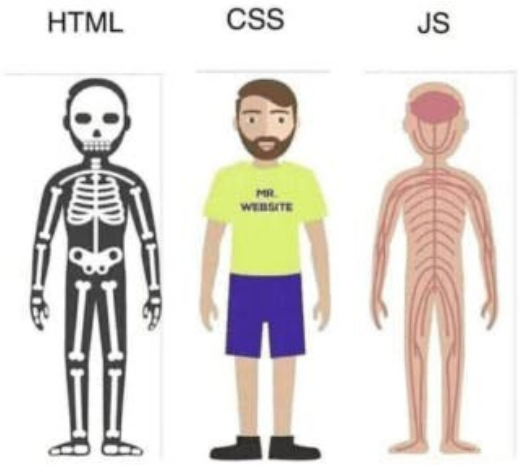
Ako se ostvari uspješna konekcija na lokalnu WiFi mrežu, odziv ESP modula trebao bi biti:

```
WIFI DISCONECTED
WIFI CONNECTED
WIFI GOT IP
```

Da biste provjerili koja je IP adresa dodijeljena vašem ESP modulu u lokalnoj WiFi mreži, potrebno je poslati AT naredbu ***AT+CIFSR.***

Za neke od vas ovaj će dio biti prvi susret s mrežnim tehnologijama, stoga će ovaj segment biti malo detaljnije opisan.

Kao što smo već naveli, za izradu jednostavne mrežne aplikacije potrebni su nam HTML, CSS i JavaScript. Najbolji opis ova tri segmenta dat je u slici ispod.

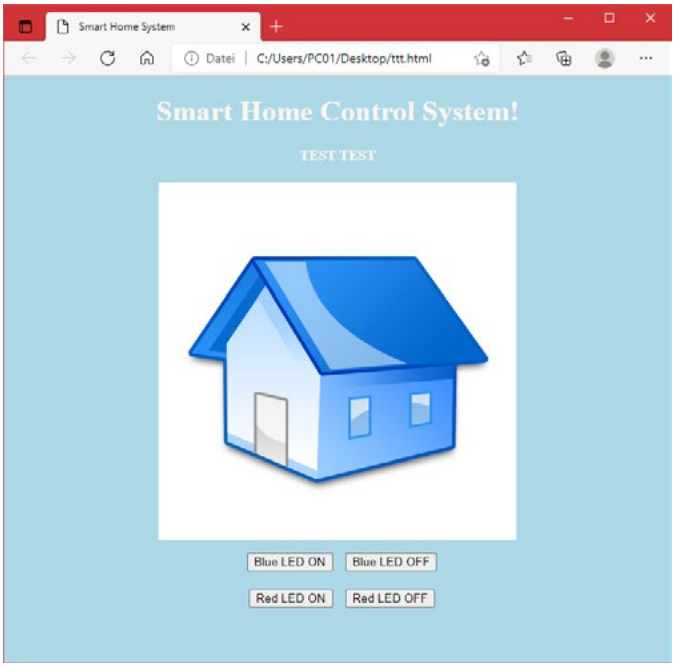


Slika pr. 12.5. *Struktura mrežne stranice*

HTML je skraćenica od "HyperText markup language" i koristi se za izradu glavne strukture mrežne stranice te s pomoću njega radimo dodavanje dugmadi, slika, zaglavalja, tablica kao i ostalih elemenata. HTML kao jezik posjeduje niz elemenata s pomoću kojega pregledniku šalje informaciju kako da prikaže korisniku mrežni sadržaj. Ovi elementi jezika predstavljaju tzv. tagove.

CSS je skraćenica od *cascading style sheet*, te nakon izrade glavne strukture stranice služi za fino podešavanje njenog izgleda. Dakle funkcija mu je da definira izgled pojedinih HTML elemenata.

Javascript je programski jezik s pomoću kojega mrežne stranice nude interakciju s korisnikom. U ovom konkretnom primjeru koristit će se za slanje HTTP zahtjeva sa strane klijenta (mrežnog preglednika) ka mrežnom serveru (ESP8266), za uključenje ili isključenje potrošača.



Slika pr. 12.6. *Izgled testne stranice*

Za pisanje HTML koda možete koristiti Notepad++ editor.

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Home System</title>
</head>
<body>
    <h1> Smart Home Control System!</h1>
    <h4>TEST TEST!</h4>
     <img src=" http://home.citycable.ch/fayachi/design/image/icone/
home.jpg" width="400px" height="400px">
    <br>
    <button id="111" class="button">LAMP ON</button>
    <br>
    <button id="110" class="button">LAMP OFF</button>
</body>
</html>
```

Opis sintakse

- **<!DOCTYPE html>:** deklaracija da je riječ o HMTL5 dokumentu
- **<html>:** root element HTML stranice
- **<head>:** metainformacije o stranici
- **<title>:** naziv stranice, vidljiv u tabu preglednika
- **<body>:** vidljivi dio sadržaja mrežne stranice nalazi se unutar ovih tagova
- **<h1>:** element definira velike elemente, recimo format teksta
- **<img>:** dodaje sliku na mrežnu stranicu
- **
:** novi red
- **<button>:** dodaje *button* element na mrežnu formu

Svaki *button* unutar mrežne aplikacije ima nekoliko atributa koji su jako važni za našu konkretnu stranicu, a to su ***id*** i ***class.***

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Home System</title>
</head>
<body style="background-color: seagreen; color: seashell; text-align: center;">
    <h1> Smart Home Control System!</h1>
    <h4>TEST TEST!</h4>
      <img src=" http://home.citycable.ch/fayachi/design/image/icone/home.jpg" width="400px" height="400px">
    <br>
     <button style="margin: 10px;" id="111" class="button">LAMP ON</button>
    <br>
     <button style="margin:10px;" id="110" class="button"> LAMP OFF</button>
</body>
</html>
```

U ovoj iteraciji na čisti HTML kod dodali smo malo stila, da bi izgledao malo ljepše. Unutar ***body*** taga dodali smo style="background-color: seagreen; color: seashell; text-align: center;" da bi stranica bila centrirana u pregledniku i promijenili smo joj pozadinsku boju, te boju fonta.

Također u button tagu dodano je style="margin: 10px;" da bismo napravili marginu od 10 piksela oko buttona.

Nakon što smo završili sa strukturom mrežne stranice, potrebno je dodati funkcionalnost na nju. Ideja je kada korisnik klikne na button "Lamp ON", browser pošalje zahtjev u kojem

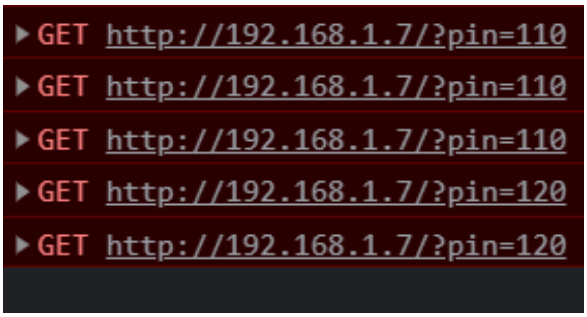
se nalaze jedinstveni podatci za klik tog buttona. Kod Arduino treba organizirati da prepozna te podatke te odradi korespondentnu akciju na svojim digitalnim izlazima.

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Home System</title>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
</head>
<body style="background-color: seagreen; color: seashell; text-align:
center;">
    <h1> Smart Home Control System</h1>
    <h4>TEST TEST!</h4>
    <img src="dogMeme.jpg" width="400px" height="400px">
    <br>
      <button style="margin: 10px;" id="111" class="button">LAMP ON</
button>
    <br>
<button style="margin:10px;" id="110" class="button"> LAMP OFF</button>
   //start Javascript.
        <script>
          //ako je aktiviran button iz ".button" klase
        $(".button").click(function () {
         //pokupi id i spremi ga u "p" variablu.
            var p = $(this).attr('id');
         // button id koji se salje ka web-serveru.
            pin: p
        //posalji request

            $.get("http://192.168.1.4:80/", {  pin: p
            });
        });
    </script>
</body>
</html>
```

Sva se magija dešava u sljedećem dijelu koda:

```
<script>
        $(„.button").click(function () {
            var p = $(this).attr(,id');
            pin: p
            $.get(„http://192.168.1.4:80/", {
            pin: p
            });
        });
    </script>
```

Kada korisnik klikne na bilo koji button na mrežnoj stranici, on u pozadini pokrene funkciju **.click**. Ta funkcija "pokupi" atribute, o kojima smo ranije pisali, "id" i "p", koji bi trebali biti jedinstveni u vašoj aplikaciji i spremi ih unutar "p" varijable, te se potom pošalje **GET** zahtjev s pripadajućom vrijednošću **id-a** kliknutog buttona. Ako aktivirate opciju *developer tools* u svom pregledniku, vidjet ćete sljedeće zahtjeve.



Slika pr. 12.7. *Izgled GET zahtjeva*

S ovim smo završili s mrežnim dijelom projekta. Vaš zadatak bi mogao biti da umjesto jednog buttona na mrežnu formu dodate onoliko buttona koliko imate slobodnih digitalnih pinova na svom Arduinu. Dosje s mrežnom aplikacijom možete otvoriti u nekom od mrežnih preglednika, obvezno uključiti opciju *developers tool*. Važno je da oba uređaja u ovom slučaju PC i ESP8266 budu na istoj WiFi mreži.

Kada ste kreirali svoju mrežnu stranicu, bilo bi je potrebno "hostati" bilo na neki remote mrežni server ili lokalno. Najjednostavnija opcija je da se iskoristi integrirani IIS servis unutar Windows operativnog sustava, a njega je potrebno aktivirati prateći sljedeće korake:

- Otvorite **Control Panel** i odite na **Programs and Features > Turn Windows features on or off**.
- Aktivirajte **Internet Information Services**.
- Proširite **Internet Information Services** stavku i provjerite je li mrežni server komponenta označena.
- Kliknite **OK**.
- Kopirajte svoju mrežnu stranicu na lokaciju C:/inetpub/wwwroot/imevasestranice.html
- Provjerite IP adresu svog PC-a
- Mrežnoj stranici možete pristupiti s mobilnog uređaja koji je na istoj WiFi mreži na sljedeći način
- http://ipadresavasegPC:80/imevasestranice.html

```
//pozivanjem SoftwareSerial biblioteke će omogućiti korištenje pinova 2
i 3 kao Rx, Tx.
#include <SoftwareSerial.h>
//podesiRx ==> Pin 2; TX ==> Pin3.
SoftwareSerial esp8266(2,3);

//kreiraj konstantu imena "serialCommunicationSpeed".

#define serialCommunicationSpeed 9600

//kreiraj konstantu "DEBUG" i pridruži vrijednost true.

#define DEBUG true

 //pridruži varijabli imena "redLED" vrijednost 12

int redLED =12;

 //pridruži varijabli imena "blueLED" vrijednost 11

int blueLED =11;

void setup()
{
//postavi pin 12 kao izlazni pin.

  pinMode(redLED,OUTPUT);

//postavi pin 11 koa izlazni pin.

  pinMode(blueLED,OUTPUT);

//deaktiviraj LED na početku aplikacije.

  digitalWrite(redLED,LOW);

//aktiviraj LED na početku aplikacije.

  digitalWrite(blueLED,HIGH);

//inicijaliziraj hardversku serijsku komunikaciju na pinovima (0, 1) i
brzini 9600.

  Serial.begin(serialCommunicationSpeed);

//inicijaliziraj softversku serijsku komunikaciju na pinovima (2, 3) i
brzini 9600.

  esp8266.begin(serialCommunicationSpeed);
//poovi funkciju "InitWifiModule()" za inicijalizaciju komunikacije iz-
među  ESP8266 i WiFi Routera ili mobilnog hotspota.

 InitWifiModule();

//poslije uspješne inicijalizacije, isključi LED.
```

```
digitalWrite(blueLED,LOW);
}

void loop()
{
//ako su zaprimljeni podaci parsirajte ih ako ne ulazi u petlju

  if(esp8266.available())
  {
//traži"+IPD," string u zaprimljenim podacima ako traženi string posto-
ji ".find()" vraća true.

if(esp8266.find("+IPD,"))
    {

//sačekaj 1 da se buffer napuni podacima.

      delay(1000);

//Oduzimanje 48 od zaprimljenih podatka se radi jer funkcija read() vra-
ća ASCII decimalne vrijednosti.

     int connectionId = esp8266.read()-48;

//traži "pin=" string unutar zaprimljenih podataka.

      esp8266.find("pin=");

//pročitaj prvi bajt iz ulaznog buffera.


      int pinNumber = (esp8266.read()-48)*10;

//pročitaj drugi bajt iz ulaznog buffera.

       pinNumber = pinNumber + (esp8266.read()-48);

//pročitaj treći bajt iz ulaznog buffera. Sačuvaj ga u "statusLed" va-
rijablu.

     int statusLed =(esp8266.read()-48);

// ON/OFF LED "pinNumber" u zavisnosti od vrijednosti "statusLed" va-
rijable.


      digitalWrite(pinNumber, statusLed);

//print  "connectionId" vrijednost u serial monitoru za debugging.

      Serial.println(connectionId);

 //print "pinNumber" vrijednost u serial monitoru za debugging.

      Serial.print(pinNumber);
```

```
//print nekoliko praznih mjesta radi lakšeg praćenja informacija.

      Serial.print("      ");

//print "statusLed" vrijednost u serial monitoru za debugging.

      Serial.println(statusLed);

//zatvori TCP/IP konekciju.


String closeCommand ="AT+CIPCLOSE=";

//dodaj "connectionId" vrijednost u string.

closeCommand+=connectionId;

//dodaj "\r\n" stringu, simuliramo "novi red kao na tastaturi"

closeCommand+="\r\n";

//Pošalji komandu ka ESP8266 module na izvršavanje.

sendData(closeCommand,1000,DEBUG); /
    }
  }
}
/*****************************************************************
*********************************
* Name: sendData
* Description: this Function regulates how the AT Commands will ge sent
to the ESP8266.
*
* Params: command - the AT Command to send
*                 - timeout - the time to wait for a response
*                  - debug - print to Serial window?(true = yes, false
= no)
*
* Returns: The response from the esp8266 (if there is a reponse)
*/
String sendData(String command, const int timeout, boolean debug)
{

//inicijaliziraj String varijablu "response".

String response = "";

//posalji AT komandu ka esp8266 (ARDUINO -> ESP8266).

esp8266.print(command);
//pokupi vrijeme od kada je aplikacija aktivna i sačuvaj ga u varijablu
"time".

long int time = millis();
```

```
//kod unutar vitičaste zagrade izvrši svake sekunde.

    while( (time+timeout) > millis())

    {
//provjeri da li postoji neki odziv sa ESP8266-a sačuvan u ulaznom
bufferu Arduina?

    while(esp8266.available())
{
//ako je ovo tačno, uzmi sljedeći karakter iz ulaznog buffera i sačuvaj
ga u varijabli "response" String.

    char c = esp8266.read();

//puni elemente stringa sa svakim novopristiglim karakterom.

    response+=c;
      }
}

//ako je "debug" TRUE, printaj odziv u Serial monitoru.

   if(debug)
{
      Serial.print(response);
    }
    return

 //vrati  String "response".

response;
}
/*******************************************************************
****
* Name: InitWifiModule
* Description: this Function gives the commands that we need to send to
the sendData() function to send it.
*
* Params: Nothing.
*
* Returns: Nothing (void).
*/

void InitWifiModule()
{

//reset ESP8266 module.

  sendData("AT+RST\r\n", 2000, DEBUG);
//delay(1000);

//spoji se na WiFi mrežu.
  sendData("AT+CWJAP=\"PUT YOUR SSID\",\"PUT YOUR PASSWORD\"\r\n", 2000,
DEBUG);
```

```
  delay (3000);
//podesi ESP8266 WiFi mode na station mode.
  sendData("AT+CWMODE=1\r\n", 1500, DEBUG);
delay (1000);
//Prikaži IP Address, and the MAC Address.
  sendData("AT+CIFSR\r\n", 1500, DEBUG);
delay (1000);
//Omogući više konekcija.
  sendData("AT+CIPMUX=1\r\n", 1500, DEBUG);
  delay (1000);
//početak komunikacije na portu 80 sa web-serverom kroz http zahtjeve.
  sendData("AT+CIPSERVER=1,80\r\n", 1500,DEBUG);
}
```

Sam bi kod u ovoj fazi trebao biti u velikoj mjeri jasan: provedene su dvije funkcije InitWifiModule() i sendData(). Pri čemu sendData() regulira način na koji se AT naredbe šalju ka modulu ESP8266. InitWifiModule() definira koje će AT naredbe funkcija sendData() slati ka ESP8266.

Unutar funkcije loop() čeka se dolazni HTTP request header i u njemu se traži "+IPD", što je znak da je zahtjev uspješno došao, potom se čita vrijednost varijable pin ->p, "111" se sastoji iz dva broja: prvi dio 11 je broj pina, a zadnja je znamenka vrijednost koju želimo da pin ima, recimo ako želimo da je pin uključen, to znači da će zadnja znamenka u troznamenkastom broju biti 1 ->HIGH. Odnosno, ako korisnik klikne na button "LAMP OFF", vrijednost varijable pin->p će biti "110".

Jedna od važnijih stvari na koju morate obratiti pažnju je da upišete akreditive svoje WiFi mreže u liniji

```
sendData("AT+CWJAP=\"PUT YOUR SSID\",\"PUT YOUR PASSWORD\"\r\n", 2000, DEBUG);
```

Vjerojatno ćete na neki prethodnih projekata dodati WiFi opciju. Sretno kodiranje!

ISHOD UČENJA	RAZRADA ISHODA – INDIKATORI
Učenik/-ica:	**Učenik/-ica:**
- opisuje ulogu elektronskih komponenti koje su korištene u projektu	- navodi ulogu i osnovne značajke modula ESP8266 IoT - navodi pinout WiFi modula definirajući ulogu svakog pojedinačno - objašnjava osnove AT komandnog seta navodeći neke od komandi
- koristi električnu shemu spoja za povezivanje Arduina i elektronskih komponenti	- povezuje ESP8266 i Arduino Uno prema shemi spoja - spaja modul ESP8266 na lokalnu WiFi mrežu - povezuje preostale komponente u projektu prema shemi spoja
- kreira jednostavni mrežno bazirani kontrolni panel za upravljanje uređajima koji su spojeni na lokanu WiFi mrežu	- kreira mrežnu aplikaciju za realizaciju projektne ideje - kreira embedded aplikacije za realizaciju projektne ideje - uvezuje embedded i mrežnu aplikaciju - proširuje stečeno znanje iz HTML-a, CSS-a i Javascript programskog jezika - primjenjuje HTML tagove za kreiranje mrežne aplikacije - kreira buttone unutar mrežne aplikacije koristeći atribute id i class - formatira mrežnu aplikaciju koristeći adekvatne stilove - objašnjava ulogu funkcija InitWifiModule() i sendData() u kodu
- koristi odgovarajuće knjižnice unutar programa za realizaciju projektnog zadatka	- koristi knjižnice SoftwareSerial za debugging softvera i troubleshooting
- verificira i izvršava program	- rekonfigurira modul EPS8266 koristeći AT komande - analizira programski kod - otklanja pogreške ako su nastale u kodu prema smjernicama nastavnika
- testira projektni zadatak	- učitava kod na platformu Arduino radi testiranja projektnog zadatka - analizira projektni zadatak - koristi ESP8266 IoT za razvijanje projektnih ideja Smart Home

Prostor za vježbu:

```
 void setup() {

}
void loop() {

}
```

Zaključak

Prije svega se nadamo da vas ovaj praktikum nije uplašio. Jasno je da ne postoji knjiga koja će vas naučiti sve. Ali one koje vam otvore vrata i prošire vidike su tu, možda je i ovo jedna od njih. Kao i u svim stvarima koje su vezane za život, jednostavno moramo skupiti hrabrost i zakoračiti dalje. A kada govorimo o programiranju, onda moramo otići dalje od aplikacije "Hello World".

Ovaj je priručnik tu da vam pokuša približiti mogućnosti razvojne platforme Arduino uz kombinaciju s dodatnim senzorima i aktuatorima, koje nisu dio Arduino Starter kita. Neki su projekti jednostavni, neki su multidisciplinarni. Najvažnije je i kada ne ide kako treba, nemojte odustajati.

Priručnik je tu da vas uvede u malo kompliciranije vježbe, ali njegov pravi cilj je da pokušate samostalno kombinirati različite senzore, zaslone i aktuatore te osmisliti neki novi *cool gadget* koji će zaludjeti svijet, a Vas učiniti planetarno poznatim. Kako god ovo zvučalo trenutačno nemoguće, nikada ne smijete zaboraviti da je za uspjeh pored truda i zalaganja potrebna još samo ideja.

Naravno, trebate reći sebi "samo nebo je granica" i ne zaboraviti **pravilo 1** kod realizacije bilo kojeg projekta: "**keep it simple**" (neka bude jednostavno). Ne pokušavajte zadiviti tehnikama pisanja koda, već njegovom funkcionalnošću.

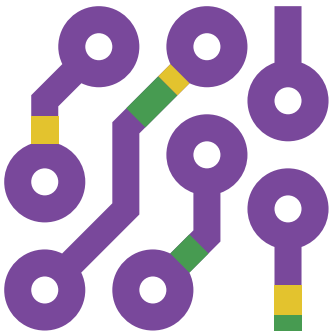
IT Girls klub vam želi uspješno kodiranje i pregršt projekata u kojima ćete uživati.

Literatura

Halilović, M. (2019). *ARDUINO ZA SVE*. ITGirls inicijativa u okviru projekta "IT Girls dolaze u vaše škole".

Vuković, M. (2017). *Mikroupravljački sustav za emulaciju električnih.* Preuzeto od https://repozitorij.etfos.hr/islandora/object/etfos:1559

1. https://docs.arduino.cc/
2. https://docs.arduino.cc/tutorials/
3. https://www.arduino.cc/reference/en/
4. https://learn.adafruit.com/category/learn-arduino
5. https://www.monolithicpower.com/
6. https://izradi.croatianmakers.hr/lessons/?technology=54



2021.