



Marko Tanasković

INTERNET STVARI

INTERNET OF THINGS

IOT

UNIVERZITET SINGIDUNUM

Marko Tanasković

INTERNET STVARI

ARDUINO PRAKTIKUM

PRVO IZDANJE

Beograd, 2020.

INTERNET STVARI

Autor:

doc. dr Marko Tanasković

Recenzenti:

Prof. Mladen Veinović

Prof. Marina Marjanović

Izdavač:

UNIVERZITET SINGIDUNUM

Beograd, Danijelova 32

www.singidunum.ac.rs

Za izdavača:

dr Milovan Stanišić

Tehnička obrada:

Miloš Višnjić

Dizajn korica:

Aleksandar Mihajlović

Godina izdanja:

2020.

Tiraž:

400 primeraka

Štampa:

Caligraph, Beograd

ISBN: 978-86-7912-724-2

Copyright:

© 2020. Univerzitet Singidunum

Izdavač zadržava sva prava.

Reprodukcija pojedinih delova ili celine ove publikacije nije dozvoljena.

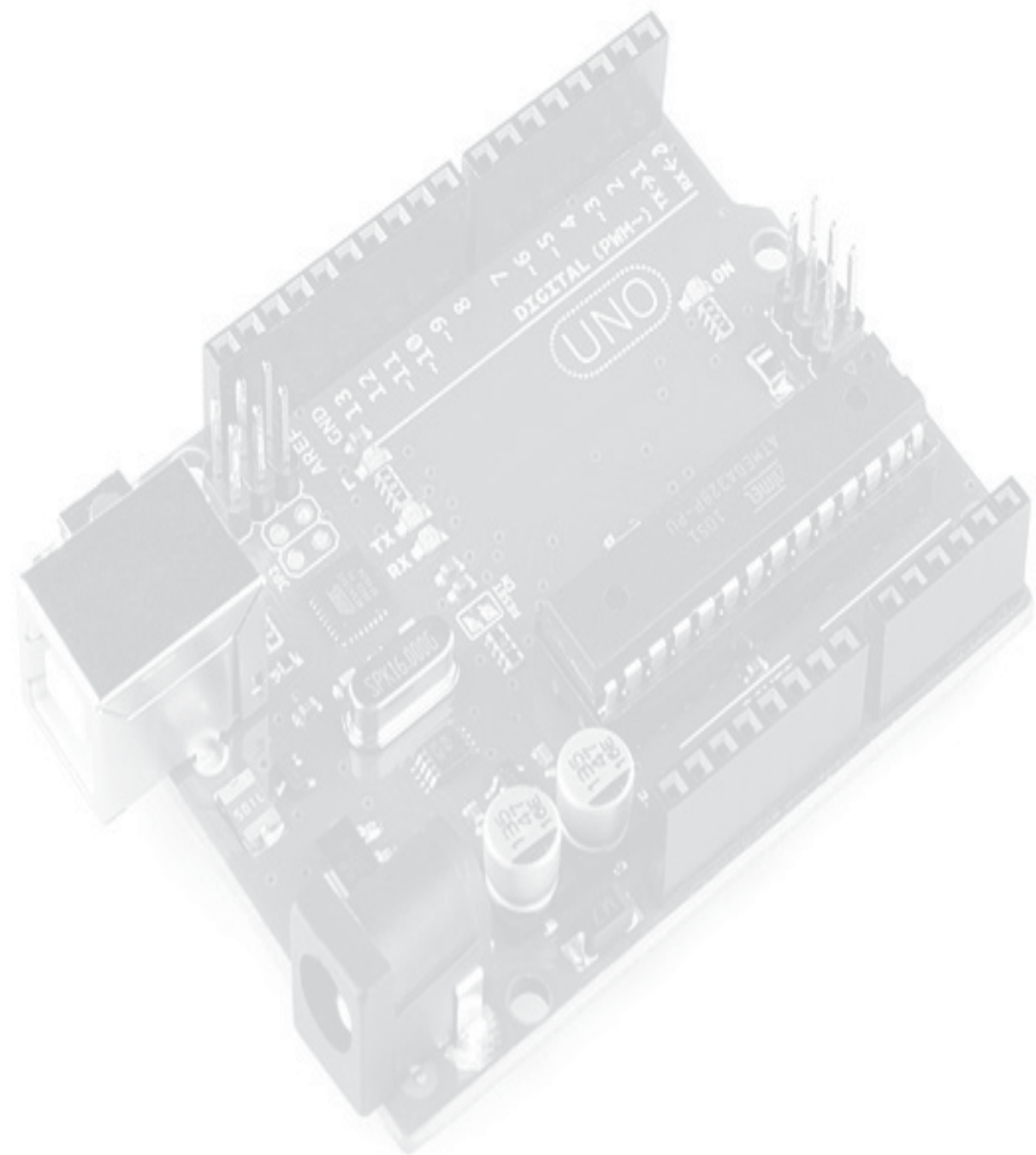
SADRŽAJ

1. UVOD	1
2. ARDUINO RAZVOJNA PLATFORMA	3
2.1.Arduino kao hardverska platforma	4
2.2.Softversko okruženje za pisanje Arduino programa	7
2.3.Povezivanje Arduina sa PC računarom	9
2.4.Struktura Arduino programa	11
2.5.Proces instalacije programa na Arduino platformu	12
3. SKLAPANJE ELEKTRIČNIH KOLA	15
3.1.Osnovne električne komponente	15
3.2.Povezivanje komponenti pomoću protoborda	19
3.3.Merenje napona i struja u električnom kolu	21
Primer 3.1. - Merenje jednosmernog napona	22
Primer 3.2. - Testiranje kratkog spoja	24
4. DIGITALNI IZLAZI I RAD SA DIODAMA	27
4.1. Digitalni izlazi	27
4.2. Dioda	28
4.2.1. Light Emitting (LE) dioda	29
4.2.2. Red Green Blue (RGB) dioda	30
4.3.Primeri	31
Primer 4.1. - Treperenje diode	31
Primer 4.2. - Dve diode koje se naizmenično pale	34
Primer 4.3. - RGB Dioda	36
4.4.Zadaci za samostalan rad	38
5. DIGITALNI ULAZI I RAD SA TASTERIMA	41
5.1. Tasteri	41
5.2.Primeri	46
Primer 5.1. - Upotreba tastera	46
Primer 5.2. - Uključenje diode pritiskom na taster	48

Primer 5.3. - Dve diode i taster	51
5.3.Zadaci za samostalan rad	53
6. NAPREDNE OPERACIJE SA DIGITALNIM ULAZIMA I IZLAZIMA: MEHANIZAM PREKIDA I IMPULSNO ŠIRINSKA MODULACIJA	55
6.1. Mehanizam prekida	55
6.2.Impulsno širinska modulacija	57
6.3.Primeri	58
Primer 6.1. - Dve diode i taster	58
Primer 6.2. - Promena intenziteta osvetljaja diode	61
6.4.Zadaci za samostalan rad	63
7. TAJMERI	65
7.1. Primeri	66
Primer 7.1. - Treperenje diode	66
Primer 7.2. - Promena periode treperenja diode	68
Primer 7.3. - Dve diode koje trepere i taster	69
7.2 Zadaci za samostalni rad	72
8. ANALOGNI ULAZI I SENZORI	75
8.1. Analogni ulazi	77
8.2.Fotootpornik	77
8.3.Senzor temperature	78
8.4.Primeri	80
Primer 8.1. - Kontrola intenziteta osvetljaja diode putem potenciometra	80
Primer 8.2. - Automatska regulacija osvetljenja	82
Primer 8.3. - Interaktivna indikacija stanja osvetljenosti	84
Primer 8.4. - Indikacija temperature prostorije	86
8.5. Zadaci za samostalan rad	91
9. DISPLEJI	93
9.2.Sedmosegmentni LED displej	93
9.3.LCD (Liquid Crystal display) displej	99
9.4.Primeri	102
Primer 9.1. - Ispis jednocifrenog broja na LED displej	102
Primer 9.2. - Promena aktivnog LED displeja	104
Primer 9.3. - Ispis dvocifrenog broja na LED displej (Multipleksiranje)	106
Primer 9.4. - Ispis sadržaja na LCD displej	109

Primer 9.5. - Promena sadržaja na LCD ekranu pritiskom na taster	112
Primer 9.6. - Merenje temperature i osvetljenosti	115
Primer 9.7. - Digitalni merač otpornosti	121
Primer 9.9. - Štoperica	124
9.5.Zadaci za samostalan rad	129
10. SERIJSKA KOMUNIKACIJA I MODULI	131
10.1. Bluetooth modul	135
10.2. Ultrazvučni modul	138
10.3. Modul za detekciju vibracija	139
10.4. Modul osetljiv na dodir	139
10.5. Digitalni barometar	140
10.6. Modul za detekciju zvuka	140
10.7. Primeri	141
Primer 10.1. - Serijska veza dva Arduina	141
Primer 10.2. - Kontrola diode putem serijske komunikacije	144
Primer 10.3. - Slanje poruka putem Bluetooth modula	146
Primer 10.4. - Slanje očitane temperature putem Bluetooth modula	148
Primer 10.5. - Merenje razdaljine	150
Primer 10.6. - Instrument za merenje razdaljine sa LED displejom	154
Primer 10.7. - Detekcija vibracija	157
Primer 10.8. - Senzor osetljiv na dodir	159
Primer 10.9. - Detekcija zvuka	160
Primer 10.10. - Očitavanje visine primenom digitalnog barometra	162
10.8. Zadaci za samostalni rad	166
11. RAD SA RELEJIMA	169
11.1.Primeri	170
Primer 11.1. - Kontrola releja tasterom	170
Primer 11.2. - Kontrola serijske veze releja	173
Primer 11.3. - Relej sa šifrom	176
11.2.Zadaci za samostalni rad	179
12. MAŠINE STANJA	181
12.1. Primeri	181
Primer 12.1. - Kontrola diode tasterom	181
Primer 12.2. - Detekcija sekvence	184
Primer 12.3. - Sekvenca treperenja diode	188

Primer 12.4. - Semafor	192
12.2. Zadaci za samostalni rad	198
13. REALIZACIJA KORISNIČKIH BIBLIOTEKA	201
13.1. Hardverska realizacija MPU-9250 modula	202
13.2. Kreiranje i implementacija biblioteke	205
13.3. Primeri	211
Primer 13.1. - Očitavanje ubrzanja duž X, Y i Z ose primenom MPU_9250 modula	211
13.4. Zadaci za samostalni rad	212
14. WIFI KOMUNIKACIJA	215
14.1. Konfiguracija modula kao AccessPoint-a i uspostavljanje servera	217
14.2. Primeri	223
Primer 14.1. - Implementacija servera na ESP8266 modulu	223
14.3. Zadaci za samostalni rad	227
15. LITERATURA	229



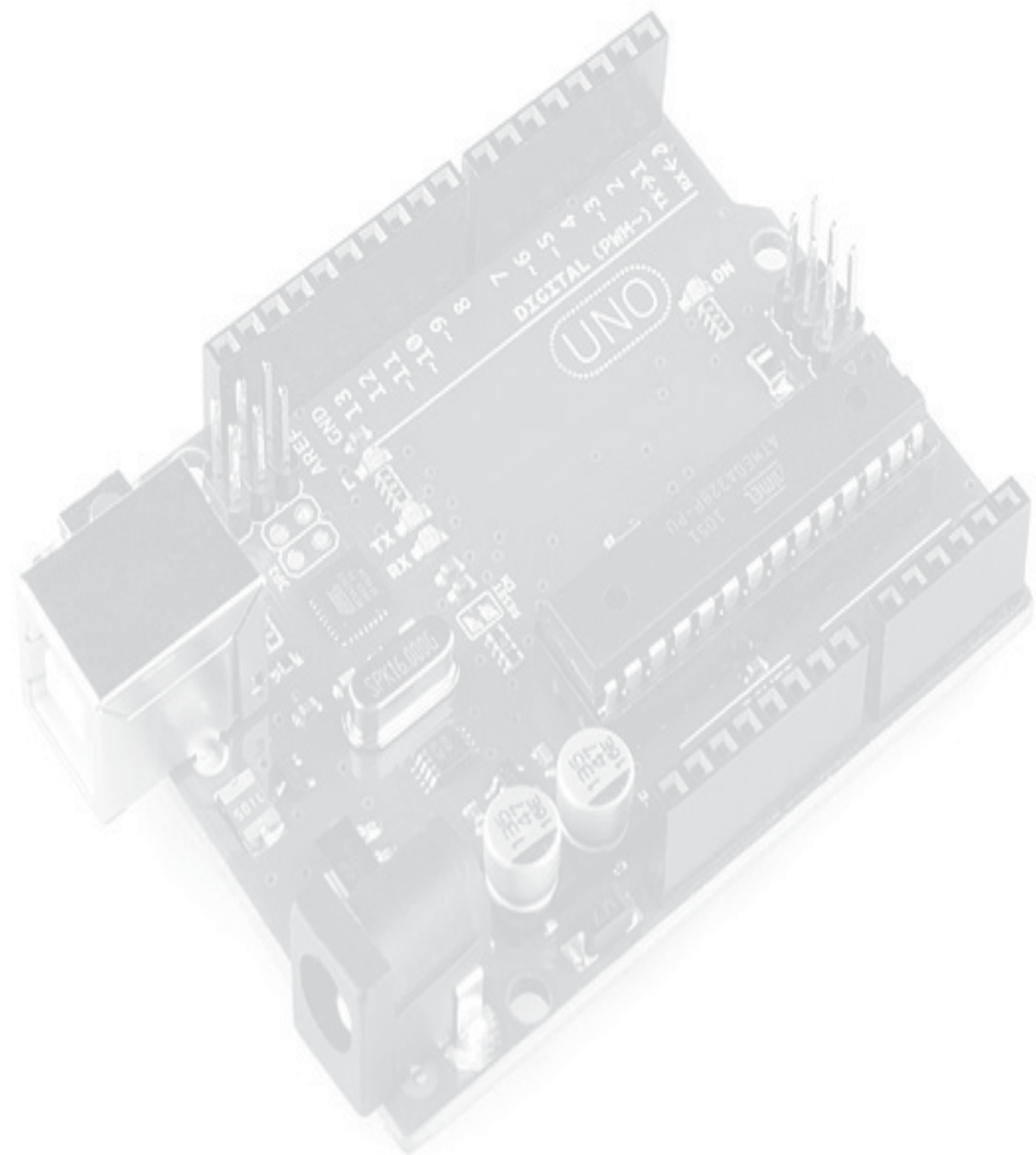
1. UVOD

Ovaj praktikum je namenjen pre svega studentima softverskog inženjerstva, kao podloga za izradu laboratorijskih vežbi za predmet internet stvari korišćenjem Arduino razvojnog sistema. Međutim mogu je koristiti svi oni koje zanimaju oblasti programiranja mikrokontrolera i interneta stvari. Priručnik je pisan tako da ga mogu koristiti i oni koji imaju početna znanja iz oblasti elektronike i programiranja mikrokontrolera, ali i oni koji tek prave svoje prve korake u ovim oblastima.

Priručnik obrađuje teme koje su važne za oblast interneta stvari polazeći od poglavlja koja daju osnove Arduino sistema, objašnjavaju kako se fizički sklapaju električne šeme, preko poglavlja koja objašnjavaju kako se prave jednostavne aplikacije koristeći digitalne i analogne ulaze, pa sve do poglavlja u kojima su predstavljeni složeniji projekti koji podrazumevaju rad sa modulima, komunikaciju sa drugim uređajima ili izlazak na internet uz pomoć wi-fija. Poglavlja su organizovana tako da su osnovne stvari i najjednostavniji problemi pokriveni u prvim poglavljima, dok težina problema koji se obrađuju raste kroz svako sledeće poglavlje.

Materijal koji je izložen u svakom od poglavlja je organizovan tako da je prvo dat kratak teoretski uvod koji opisuje temu od interesa, a potom je ta teoretska osnova ilustrovan kroz primere. Svaki primer koji je u priručniku rešen sadrži šematski prikaz električnog kola kojim se taj problem rešava, spisak neophodnih hardverskih komponenti, kao i ilustraciju njihovog fizičkog povezivanja. Arduino kod sa komentarima takođe je dat za svaki primer, a sam kod je dodatno tekstualno opisan u smislu ključne ideje rešenja. Potencijalne greške u povezivanju hardvera koje se često javljaju, kao i stvari o kojima posebno treba voditi računa u rešavanju pojedinih problema posebno su naglašene.

Autor želi da se zahvali svima koji su pomogli da ovaj priručnik dobije ovu formu. U nadi da će on biti koristan onima koja ga čitaju, autor želi uspešan rad i puno zabave svima koji se upuštaju u rad sa Arduinoom.



2. ARDUINO RAZVOJNA PLATFORMA

Arduino predstavlja open source [1] platformu za kreiranje projekata iz oblasti elektronike. Arduino razvojna platforma se sastoji iz dva dela. Jedan deo je hardverski, koji se odnosi na Arduino razvojnu ploču na kojoj se nalazi mikrokontroler i prateće hardverske komponente, dok drugi deo predstavlja softver ili IDE (Integrated Development Environment), koji služi za pisanje programskog koda, prevođenje koda na mašinski jezik (kompajliranje) i spuštanje koda na hardversku platformu.

Arduino je brzo postao popularan, kako u svetu profesionalne elektronike, tako i kod ljudi kojima elektronika predstavlja hobi. Jedan od glavnih razloga je činjenica da je programiranje mikrokontrolera u okviru Arduino platforme dosta jednostavno i zahteva samo USB kabl i PC računar. Pored toga, Arduino softver koristi pojednostavljenu verziju C++ programskog jezika, što dodatno olakšava učenje programiranja mikrokontrolera [2]. Zahvaljujući svojoj popularnosti i otvorenosti, oko Arduino platforme je nastala čitava jedna zajednica entuzijasta koji slobodno razmenjuju iskustva i dele svoje projekte. Na internetu se može naći veliki broj zanimljivih projekata [3] ali i softverskih biblioteka koje olakšavaju pisanje složenih programa i interakciju Arduino platforme sa raznim drugim hardverskim uređajima.

Arduino projekat je započeo 2003 godine i razvili su ga studenti iz Italije, grada Ivrea (Intersection Design Institute Ivrea) koji su za cilj imali da razviju što jednostavniju platformu koja će omogućiti interakciju sa senzorima, aktuatorima i drugim razvojnim pločama. Naziv Arduino potiče od imena Arduin koji zapravo predstavlja ime bara u kome su se studenti okupljali. To je staro rimsko ime jednog slavnog vojskovođe iz Ivrea [4].

Postoje razne Arduino hardverske platforme. Svaka od njih ima drugačija svojstva tj. različit broj pinova, različite mikrokontrolere i različite brzine rada. U ovoj knjizi se obrađuje Arduino UNO platforma koja se najčešće i koristi u praksi. Međutim kada se steknu osnovna znanja pri radu sa jednom platformom, vrlo lako se može preći na rad sa nekim drugim platformama. Spisak trenutno aktivnih platformi dat je u tabeli 2 - 1 [5].

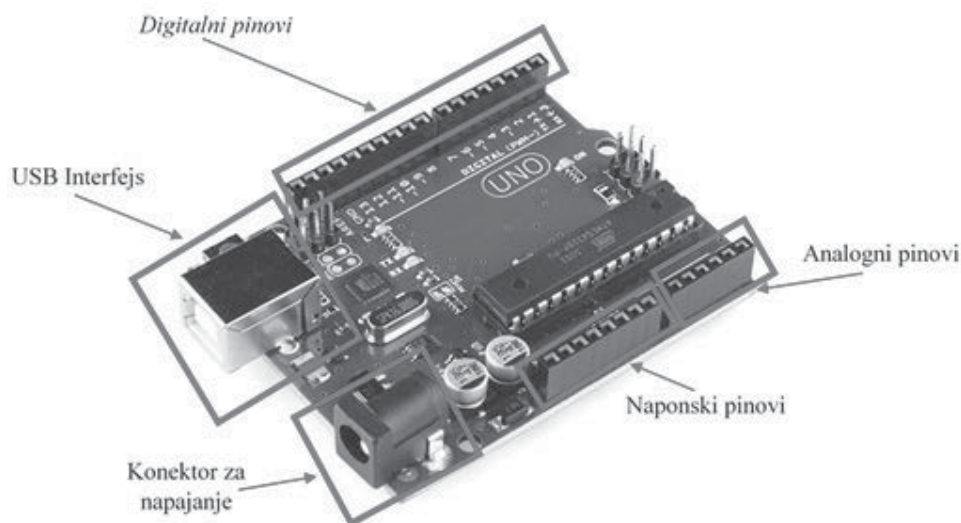
Tabela 2 - 1 Specifikacije nekih od Arduino platformi

Naziv platforme	Korišćeni mikrokontroler – bitska širina	Brzina rada	Broj pinova	Radni napon
Arduino UNO	ATmega328P – 8 bit	16MHz	14 – Digitalnih 6 – Analognih	5V
Arduino LEONARDO	ATmega32u4 – 8bit	16MHz	20 – Digitalnih 7- Analognih	5V
Arduino MICO	ATmega32u4 – 8 bit	16MHz	20 – Digitalnih 7 - Analognih	5V

Arduino NANO	ATmega328 – 8bit	16MHz	22 – Digitalnih 8 - Analognih	5V
Arduino MEGA 2560	ATmega2560 – 8bit	16MHz	54 – Digitalnih 16 - Analognih	5V
Arduino DUE	AT91SAM3X8E – 32bit	84MHz	54 – Digitalnih 12 - Analognih	5V

2.1. ARDUINO KAO HARDVERSKA PLATFORMA

Arduino UNO hardverska platforma sa naznakama osnovnih interfejsa prikazana je na slici 2 - 1. Detaljna specifikacija Arduino UNO hardverske platforme može se naći na zvaničnom sajtu Arduino projekta [6].



Slika 2 - 1 Arduino hardverska platforma sa označenim osnovnim interfejsima

Konektor za napajanje služi za napajanje Arduino hardverske platforme. Po preporukama proizvođača napon napajanja može biti u opsegu od 5V do 20V, dok optimalan napon napajanja iznosi 7V.

USB interfejs služi za povezivanje razvojne ploče sa PC računarom. Preko ovog USB linka se vrši programiranje Arduina. USB konektor se takođe može koristiti i kao serijski interfejs za razmenu podataka između Arduina i PC računara. Putem ovog porta se može napajati ceo sistem, jer on pruža mogućnost napajanja sistema naponom od 5V.

Naponski pinovi predstavljaju izvedene napone napajanja od 5V i 3.3V koji se koriste kada je neophodno dovesti napajanje ostatku sistema koji se povezuje sa Arduino hardverskom platformom. Ova dva napona predstavljaju dva najčešće korišćena napona pri radu sa elektronskim komponentama. Pored napona od 5V i 3.3V tu se nalaze i 2 GND pina koji predstavljaju masu sistema. Masa sistema je tačka nultog potencijala. Tako da kada pričamo o naponu nekog pina na Arduino hardverskoj platformi, onda se podrazumeva napon, odnosno razlika potencijala između datog pina i GND pinova. Pinovi koji se nalaze u ovom delu kao i njihova svrha izlistani su u tabeli 2 - 2.

Tabela 2 - 2 Specifikacija naponskih pinova Arduino UNO platforme

Naziv pina	Opis
IOREF	Ulazno/izlazna naponska referenca koja daje napon/uzima napon na kome radi mikrokontroler
RESET	Signal koji resetuje mikrokontroler
3.3V	Daje izlaznu naponsku referencu od 3.3V
5V	Daje izlaznu naponsku referencu od 5V
2xGND	Pin koji obezbeđuje masu (0V) ostatku sistema
Vin	Služi za napajanje arduino hardverske platforme nekim spoljašnjim naponom napajanja

Analogni pinovi omogućavaju interfejs razvojne platforme sa analognim senzori-
ma i ostalim analognim električnim kolima. Na analogne pinove se dovodi analogni
napon u opsegu od 0V do 5V. Ovi pinovi su u direktnoj vezi sa analogno-digitalnim
konverterima unutar mikrokontrolera, koji imaju zadatak da konvertuju analogni na-
pon u odgovarajuću digitalnu vrednost (binarni broj). Kada govorimo o Arduino UNO
platformi, analogna vrednost u opsegu od 0-5V se konvertuje u digitalni broj u rasponu
0-1023. To znači da će analognoj vrednosti od 0V odgovarati digitalna vrednost 0 dok
će analognoj vrednosti 5V odgovarati digitalna vrednost 1023. Pinovi koji se nalaze u
ovom delu kao i njihova svrha izlistani su u tabeli 2 - 3. Više detalja o analognim pi-
novima dato je u poglavlju 8.

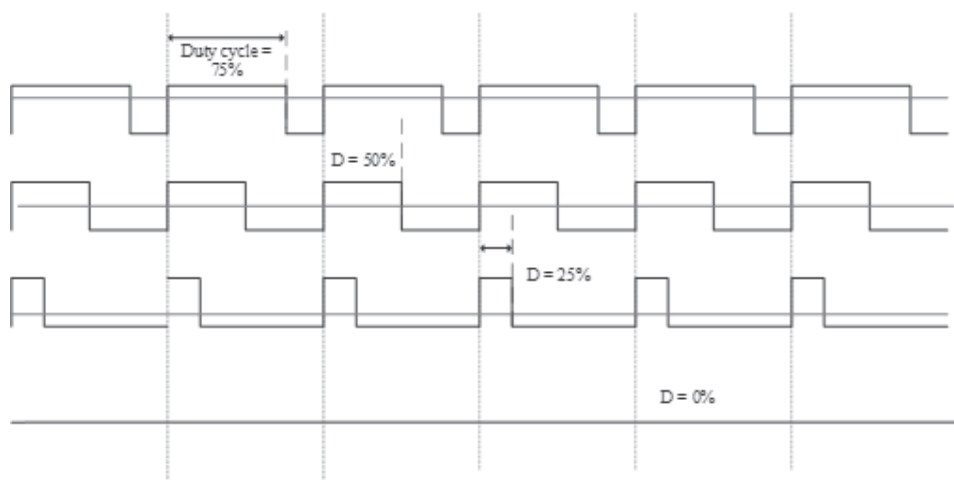
Tabela 2 - 3 Specifikacija analognih pinova Arduino UNO platforme

Naziv pina	Opis	Napomena
A0...A5	Analogni Ulazi	A4 i A5 se koriste kao SDA i SCL linije za I2C serijsku komunikaciju (detaljnije u poglavlju 10)

Digitalni pinovi se koriste za interakciju Arduino hardverske platforme sa digitalnim električnim kolima. Na digitalnom pinu se mogu softverski upisati dve vrednosti „0“ (logička nula) i „1“ (logička jedinica). Svi digitalni pinovi imaju dva režima rada: ulazni i izlazni. Ukoliko se digitalni pin konfiguriše kao izlazni nakon upisa logičke nule ili logičke jedinice na neki pin, na tom pinu se može izmeriti vrednost od 0V ili 5V, respektivno. Dakle na digitalnom pinu koji je konfigurisan kao izlazni je moguće očitati samo dva napona i to 0V ili 5V. Više o digitalnim pinovima se može saznati u poglavljima 4 i 5.

Ukoliko se pin proglasi ulaznim, to znači da se sa tog pina može očitati jedna od dve moguće binarne vrednosti („0“ ili „1“). Koja vrednost će biti očitana zavisi od toga kako je pin povezan na ostatak sistema. Ukoliko je napon na datom pinu (napon između tog pina i GND pina) manji od 1.5V, tada ćemo na pinu očitati „0“ dok ukoliko se na pinu nalazi vrednost veća od 3V na pinu ćemo očitati „1“. Ukoliko se napon na digitalnom pinu nalazi između 1.5V i 3V nije poznato koja će vrednost biti pročitana i ovakvo stanje se smatra nedefinisanim [7]. Neki od pinova imaju mogućnost generisanja PWM signala. Detaljna specifikacija digitalnih pinova data je u tabeli 2 - 4.

PWM (eng. Pulse-width modulation -srp. Impulsno-širinski modulisan) signal je digitalni periodični signal kod koga je za vreme jedne periode moguće menjati dužinu trajanja logičke jedinice odnosno logičke nule. Izgled PWM signala prikazan je na slici 2 - 2. PWM signali se obično koriste za kontrolu brzine DC motora, za promenu intenziteta osvetljenja diode, itd. Više detalja o širinsko-impulsnoj modulaciji dato je u poglavlju 6.



Slika 2 - 2: Impulsno širinski modulisan signal

Tabela 2 4 – Specifikacija digitalnih pinova Arduino UNO platforme

Naziv pina	Opis	Napomena
D0...D13	Digitalni ulazi izlazi	0,1 – Koriste se od strane USB porta za programiranje mikrokontrolera
		3,5,6,9,10,11 – Imaju mogućnost generisanja PWM signala
		2,3 – Imaju mogućnost generisanja prekida
		10,11,12,13 – Koriste se za SPI serijsku komunikaciju

Serijska komunikacija predstavlja razmenu podataka između Arduino platforme i nekog modula. Podaci se razmenjuju po postojećem protokolu. Najčešće se koriste sledeća 3 tipa serijske komunikacije:

- I2C komunikacija – Serijska sinhrona komunikacija koja koristi dve linije
- UART komunikacija - Serijska asinhrona komunikacija koja koristi 2 linije
- SPI komunikacija – Serijska sinhrona komunikacija koja koristi četiri linije

Detaljnija analiza serijske komunikacije biće sprovedena u poglavlju 10.

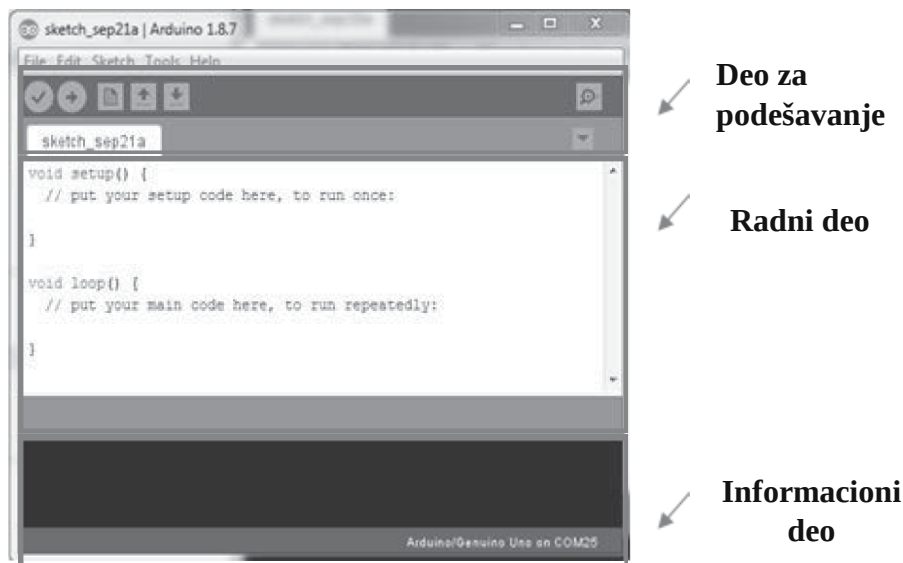
2.2.SOFTVERSKO OKRUŽENJE ZA PISANJE ARDUINO PROGRAMA

Svi programi koji se kreiraju za Arduino platforme pišu se u Arduino IDE softverskom razvojnom okruženju. Potrebni softver je moguće skinuti sa linka <https://www.arduino.cc/en/Main/Software> . Proizvođač nudi više verzija softvera koji su pisani za različite operativne sisteme. U zavisnosti od operativnog sistema, treba skinuti odgovarajući fajl. Za Windows operativni sistem postoje dva tipa fajlova koji se mogu skinuti. Prva opcija je .zip fajl u slučaju da program skidamo za računar na kome nemamo privilegije administratora. Kao druga opcija nudi se Windows Installer fajl koji skidamo u slučaju da računar na kome instaliramo ovaj softver ima privilegije administratora.

Nakon što se program preuzme sa datog linka, instalira (ili jednostavno otpakuje iz .zip arhive) i pokrene, dobija se prozor prikazan na slici 2 - 3. Radno okruženje ovog softvera se sastoji od tri celine: radni deo, informacioni deo i deo u kome se vrše podešavanja.

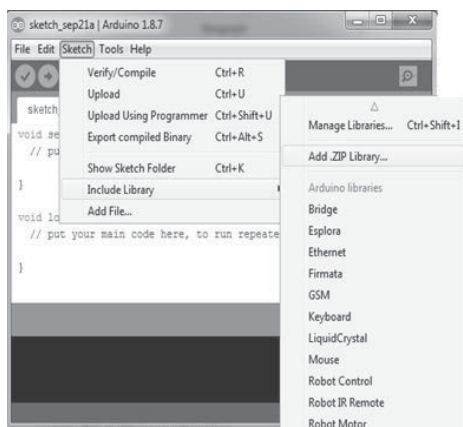
Neizostavni deo rada sa Arduino platformom predstavlja i korišćenje određenih biblioteka. Svrha biblioteka (library) jeste da korisniku olakšaju pristup određenim hardverskim elementima poput analognih pinova, digitalnih PWM portova, pinova za serijsku komunikaciju itd.

Biblioteke se mogu naći na internetu i nakon njihovog preuzimanja, najverovatnije da će tip fajla biti .zip. Da bi bilo moguće koristiti neku biblioteku, najpre je neophodno instalirati je u Arduino okruženje. Instaliranje biblioteke biće ilustrovano na primeru biblioteke za rad sa tajmerima, a detaljan opis ove biblioteke dat je u poglavlju 7.



Slika 2 - 3 - Arduino IDE softversko okruženje

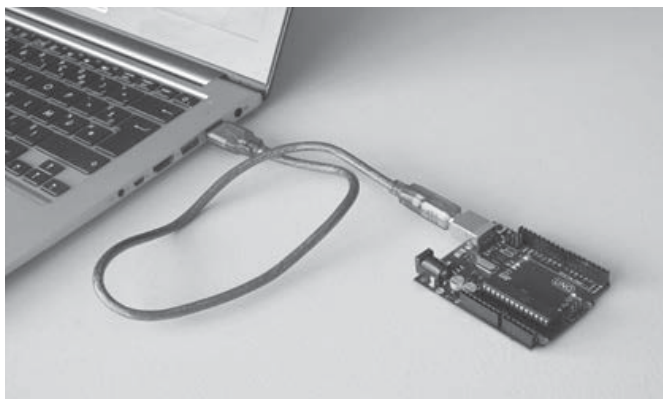
Sa link <https://github.com/PaulStoffregen/TimerOne> preuzimamo biblioteku kao .zip fajl. Nakon toga u Arduino softverskom okruženju selektuje se meni Sketch a zatim Include library, a nakon toga Add .Zip Library. Ovaj postupak je ilustrovan na slici 2 - 4. Kao rezultat otvara se prozor u kome treba navesti putanju do prethodno skinutog .zip fajla. Unošenjem putanje i potvrdom, završava se uključivanje biblioteke.



Slika 2 - 4 - Uključivanje biblioteke u projekat

2.3. POVEZIVANJE ARDUINA SA PC RAČUNAROM

Pre nego što se započne sa pisanjem programa, neophodno je pravilno povezati ceo sistem. Pod ovim se podrazumeva hardversko povezivanje elemenata i ispravna konfiguracija Arduino IDE softvera. Hardversko povezivanje podrazumeva ispravno fizičko povezivanje Arduino hardverske platforme sa PC računarom. Ovo povezivanje vrši se kablom koji sa jedne strane ima standardni (tip A) USB priključak (za PC), a sa druge USB tip B priključak (za Arduino). Primer ispravno povezanе Arduino platforme i PC računara prikazan je na slici 2 - 5.

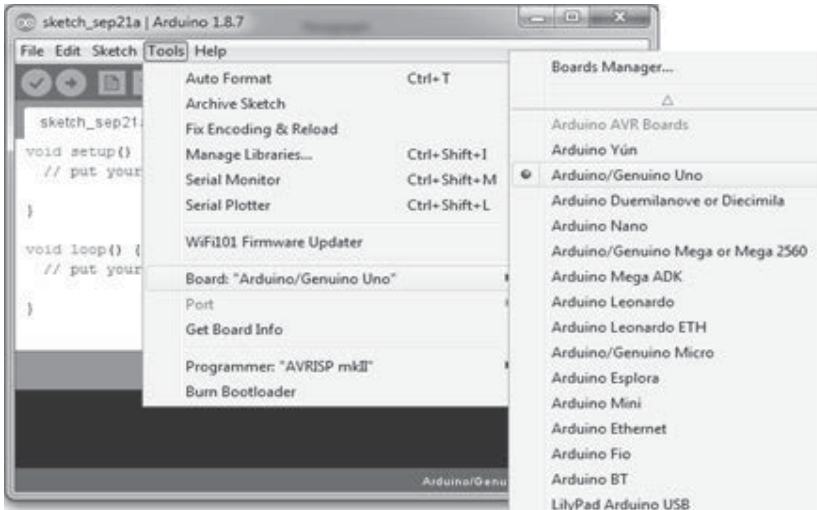


Slika 2 - 5 - Povezivanje Arduino UNO hardverske platforme sa PC računarom

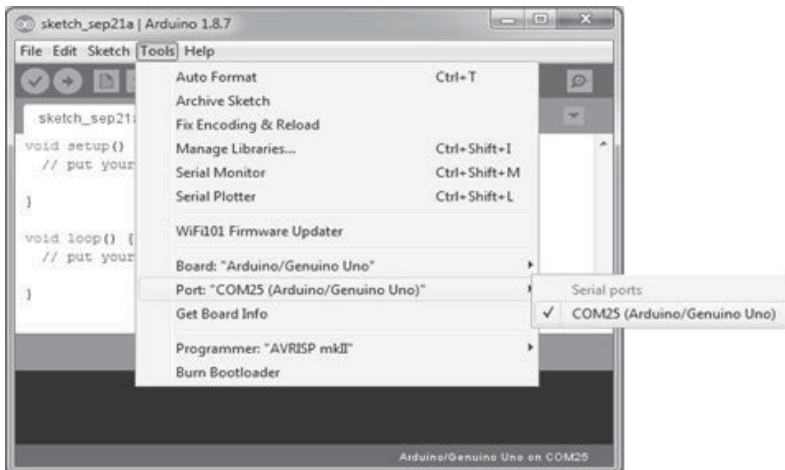
U okviru podešavanja Arduino IDE softvera, pre svega je potrebno izvršiti selekciju platforme za koju se piše kod, a nakon toga neophodno je izvršiti selekciju USB porta PC računara putem koga se odvija komunikacija sa Arduino.

Izbor platforme se vrši tako što se u meniju tools izabere sekcija Board a zatim se iz izlistanih platformi izabere platforma za koju pišemo program (u našem slučaju Arduino UNO). Ovo podešavanje je ilustrovano na slici 2 - 6.

Izbor komunikacionog porta se vrši tako što se u meniju Tools, sekciji Port, izabere onaj port koji u nazivu ima ime platforme za koju pišemo kod. Ovaj postupak je ilustrovan na slici 2 - 7.



Slika 2 - 6- Izbor odgovarajuće hardverske platforme



Slika 2 - 7 - Podešavanje porta

2.4. STRUKTURA ARDUINO PROGRAMA

Svaki program pisan za Arduino sadrži tri glavne programske celine i to:

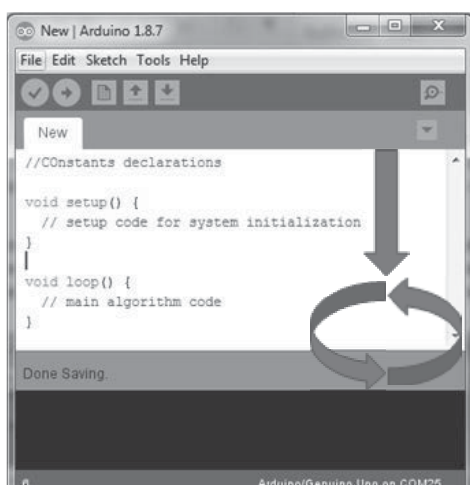
- Deklaracija promenljivih i funkcija
- Inicijalizacija sistema
- Glavni program

Deklaracija promenljivih i funkcija se vrši na samom početku programa. Pod deklaracijom promenljivih se najčešće podrazumeva deklaracija konstanti koje označavaju broj korišćenog pina, deklaracija promenljivih koje nose neku informaciju, itd. U ovom delu se deklariraju i funkcije koje se pozivaju u drugim delovima programa.

Inicijalizacija sistema podrazumeva podešavanje parametara vezanih za hardversku platformu, kako bi se ona ponašala na željeni način. To podrazumeva podešavanje direkcije pinova, početne vrednosti koje će biti na pinovima ukoliko se pinovi proglašavaju kao izlazni, konfigurisanje prekida, itd. Inicijalizacija sistema se vrši unutar funkcije setup koja se poziva samo jednom i to na samom početku izvršavanja programa.

Glavni program implementira algoritme koji realizuju željene funkcionalnosti. Ovaj deo koda se piše u glavnoj programskoj ptlji (loop). Loop funkcija se ciklično poziva. Svaki put kada se ova funkcija izvrši, ponovo se poziva. Stoga za loop funkciju možemo reći da je beskonačna petlja koja omogućava trajno ciklično izvršavanje programa na hardverskoj platformi.

Dakle, svaki program pisan za Arduino platformu mora da sadrži setup i loop funkcije. One mogu biti prazne ali neophodno je da postoje u programu. Zbog toga nakon otvaranja novog Arduino fajla dobijamo situaciju prikazanu na slici 2 - 8, gde se inicijalno nalaze dve neophodne funkcije sa praznim telima.



**Izvršava se jednom
na početku**

**Izvršava se ciklično
u beskonačnoj petlji**

Slika 2 - 8- Struktura Arduino programa

2.5.PROCES INSTALACIJE PROGRAMA NA ARDUINO PLATFORMU

Kao što je već rečeno, programi za Arduino se pišu u Arduino softverskom okruženju. Međutim, sada se postavlja pitanje kako se zapravo odvija proces programiranja Arduina, tj. kako se vrši prevođenje nekog programa koji pišemo u nešto što Arduino platforma sprovodi. Na ovo pitanje pokušaćemo da odgovorimo u nastavku ovog poglavlja.

Arduino hardverska platforma je digitalni sistem. Glavna osobina digitalnih sistema jeste da mogu da operišu samo sa dva stanja i to sa logičkom nulom („0“) i logičkom jedinicom („1“). Sa druge strane se nalazi Arduino softversko okruženje u kome se piše kod u tekstualnom formatu. Taj kod se piše po sintaksnim pravilima programskog jezika C. Ovaj programski jezik spada u više programske jezike, što znači da mikrokontroleri/računari ne mogu direktno da ga razumeju. Zbog toga je neophodno da ovaj kod bude preveden u niz nula i jedinica (mašinski kod) koji Arduino hardverska platforma može da razume i izvršava.

Unutar Arduino softverskog okruženja postoji deo koji se naziva kompajler i koji ima zadatak da kod napisan u C-u prevede na mašinski kod koji Arduino hardverska platforma može da razume i izvršava. Pokretanje procesa kompajliranja se vrši klikom na dugme u obliku znaka za tačno (štikle) koji se nalazi u gornjem levom uglu kao što je prikazano na slici 2 - 9. Nakon toga, ukoliko je kod pravilno napisan i nema sintaksnih grešaka, na sredni donjeg dela prozora se ispisiuje Done Compiling što potvrđuje da je kod uspešno kompajliran.

1. Pokretanje
kompajliranja

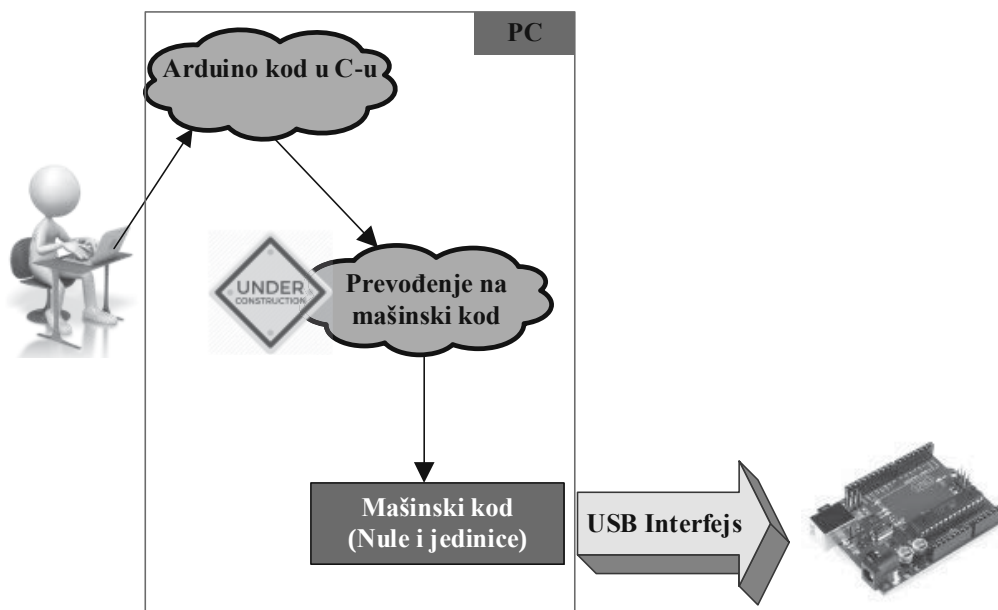
3. Prenos programa
na Arduino

2.
Kompajliranje
završeno



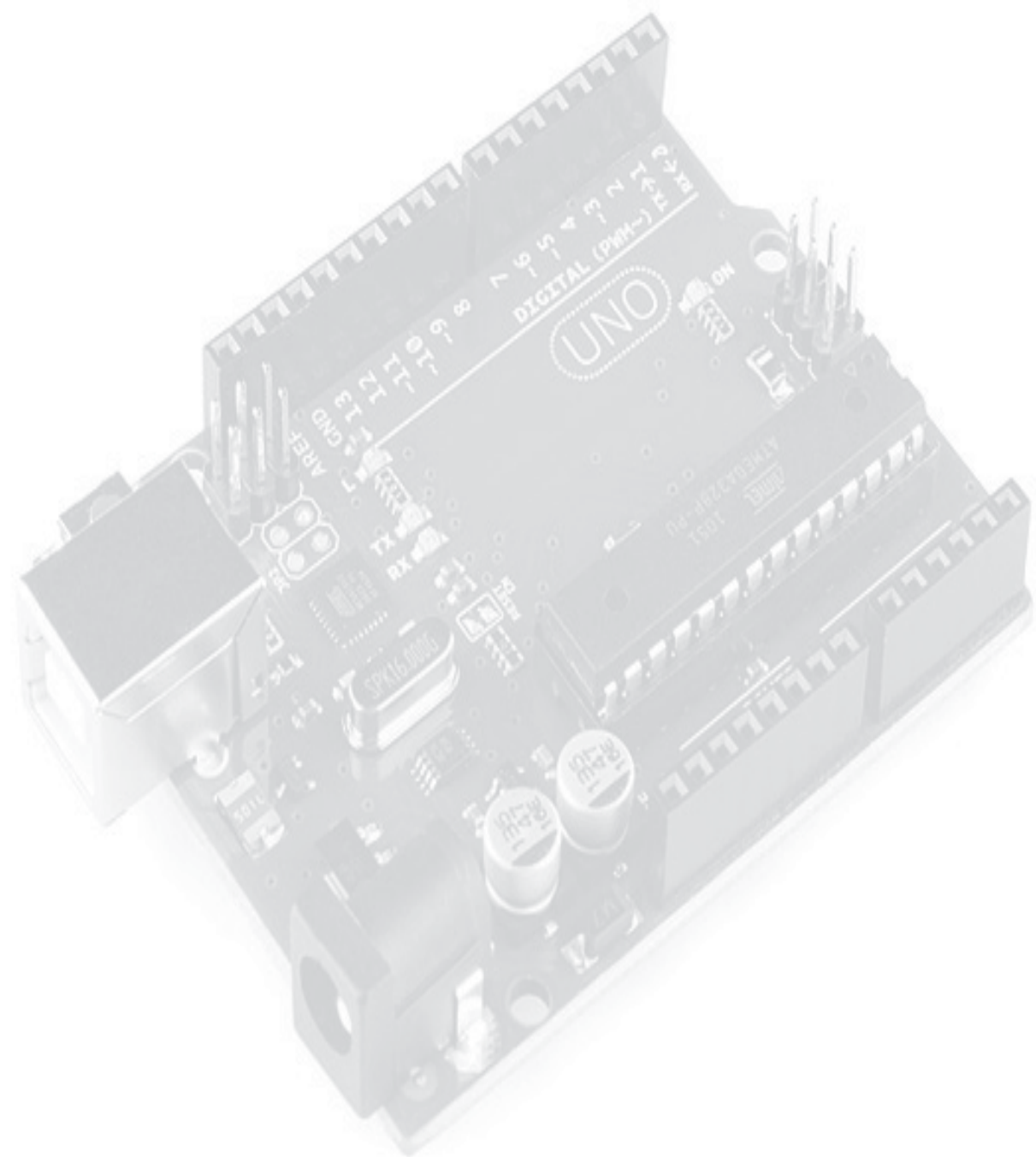
Slika 2 - 9- Primer uspešnog kompajliranja Arduino programa

Kada se kompajliranje završi, to znači da je uspešno generisan mašinski kod koji se još uvek nalazi u memoriji PC računara, pa je zatim neophodno taj kod preneti u memoriju Arduina gde će se on izvršavati. Prenosjenje ispravno kopajlirang koda vrši se klikom na dugme u obliku strelice na desno u gornjem uglu IDE prozora (slika 2 - 9). Jednom kada se kod prenese na mikrokontroler on ostaje u memoriji mikrokontrolera sve dok se ponovo ne unese neki novi kod. Opisani mehanizam programiranja mikrokontroler ilustrovan je na slici 2 - 10.



Slika 2 - 10 - Ilustracija mehanizma instaliranja koda na mikrokontroler

Treba znati da kada se jednom mašinski kod uspešno upiše u memoriju Arduina, napisani program se izvršava na Arduino hardverskoj platformi potpuno nezavisno od PC računara na kome je pisan. Pri testiranju napisanog koda, čest je slučaj da i po programiranju Arduino kontroler i PC računar ostaju povezani pomoću USB kabla. Ovakva veza PC računara i Arduina služi samo za napajanje hardverske platforme. Ukoliko bi se ova veza prekinula, a Arduino napaja iz nekog drugog izvora napajanja, program bi se na njemu izvršavao na potpuno isti način.



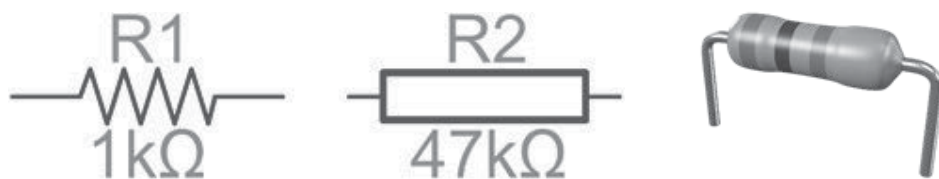
3. SKLAPANJE ELEKTRIČNIH KOLA

Tema ovog poglavlja je formiranje električnih kola i upoznavanje čitaoca sa osnovnim zakonima koji određuju njihov rad. Cilj poglavlja je da osposobi čitaoca da samostalno sklapa i proverava ispravnost električnih kola koja će biti predstavljena kroz praktične primere u ovom priručniku.

Pošto ovaj priručnik obrađuje temu Arduina, električna kola su znatno pojednostavljena jer se kao izvor napajanja u tim kolima uglavnom koriste digitalni izlazi koji imaju jednosmerni- DC napon, pa su kao takva jednostavan za rešavanje.

3.1. OSNOVNE ELEKTRIČNE KOMPONENTE

Osnovnu električnu komponentu koja se koristi u gotovo svim elektronskim kolima predstavlja otpornik. Njegov najčešći zadatak u kolima koja se sreću u ovom priručniku, jeste da ograniči protok struje. Kao posledica protoka struje kroz otpornik, na njegovim krajevima se javlja određeni napon. Ako se posmatra iz drugog ugla, tada se može reći da se struja kroz otpornik javlja kao posledica napona na njegovim krajevima. Najčešće korišćeni električni simboli otpornika i fizički izgled realnog otpornika prikazani su na slici 3 - 1.



Slika 3 - 1 - Električni simboli otpornika (levo) i fizički izgled otpornika (desno)

Napon i struja otpornika međusobno su povezani relacijom koja se naziva Omov zakon, a koja matematički ima sledeću formulu:

$$U=I \cdot R \quad (1)$$

ili

$$I=\frac{U}{R} \quad (2)$$

Sa U je označen napon na otporniku koji se izražava u voltima (V) dok je sa I označena struja kroz otpornik koja se izražava u amperima (A). Dakle napon na otporniku je proporcionalan struji koja kroz njega protiče. Konstanta te proporcionalnosti označena je sa R, naziva se otpornošću i izražava se u omima (Ω).

Prilikom rada sa otpornicima često postoji potreba da se pronađe otpornik neke specifične otpornosti ili da se pročita vrednost otpornosti nekog otpornika. Otpornost se može pročitati sa kućista otpornika na osnovu redosleda boja koje se na njemu nalaze. U tu svrhu se koristi tabela otpornosti prikazana na slici 3 - 2.



Color	Band 1	Band 2	Band 3	Multiplic.	Tolerance
Black	0	0	0	10^0 (1 Ω)	
Brown	1	1	1	10^1 (10 Ω)	$\pm 1\%$
Red	2	2	2	10^2 (100 Ω)	$\pm 2\%$
Orange	3	3	3	10^3 (1k Ω)	
Yellow	4	4	4	10^4 (10k Ω)	
Green	5	5	5	10^5 (100k Ω)	$\pm 0.5\%$
Blue	6	6	6	10^6 (1M Ω)	$\pm 0.25\%$
Purple	7	7	7	10^7 (10M Ω)	$\pm 0.1\%$
Gray	8	8	8	10^8 (100M Ω)	$\pm 0.05\%$
White	9	9	9	10^9 (1G Ω)	
Gold				10^{-1} (100m Ω)	$\pm 5\%$
Silver				10^{-2} (10m Ω)	$\pm 10\%$



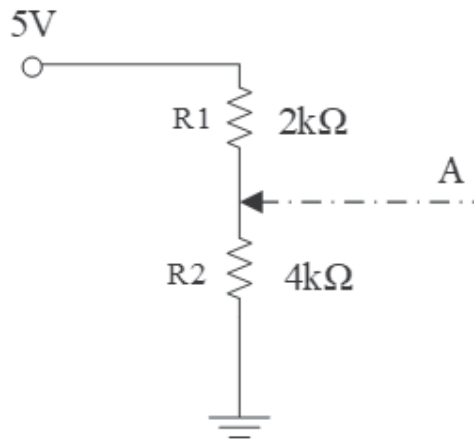
Slika 3 - 2 - Tabela za određivanje otpornosti

Postoje otpornici sa 4 ili 5 linija koje određuju njihovu otpornost. Boja svake linije predstavlja neki broj. Struktura broja koji predstavlja otpornost se može podeliti na 2 dela i to osnovni deo i množilac. U slučaju otpornika sa 5 linija (5-band-resistor) osnovni deo je trocifreni broj. Prvu cifru određuje boja prve linije, drugu cifru određuje boja druge linije i treću cifru određuje boja treće liniji. Četvrta linija predstavlja vrednost množioca. Da bi dobili tačnu otpornost neophodno je pomnožiti osnovni deo sa množiocem. U slučaju otpornosti koja ima 4 linije (4-band-resistor) osnovni deo je dvo-cifren broj koji određuju prve dve linije, dok treća linija određuje množilac. Poslednje linije u nizu kod oba tipa otpornika određuju toleranciju koja nije bitna za priremre u ovom praktikumu.

Proces određivanja otpornosti biće detaljnije objašnjen na primeru otpornika sa slike 3 - 1. Otpornik ima 4 boje. Prva je crvena a druga je crna što nakon prevođenja u brojeve na osnovu slike 3 - 2 daje osnovni broj 20. Treća linija je crvena i ona predstavlja množilac koji iznosi 100. Nakon množenja osnovnog dela i množioca dobija se vrednost $20 * 100 = 2000\Omega$ ili $2k\Omega$.

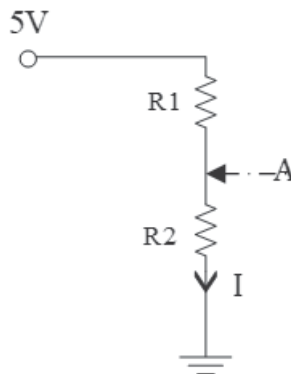
U nastavku su opisana neka tipična električna kola koja se sastoje od otpornika.

Jedno prosto električno kolo koje se sastoji od dva otpornika i jednog izvora jednosmernog napona prikazan je na slici 3 - 3.



Slika 3 - 3 - Razdelnik napona

Ovakvo kolo se često naziva razdelnik napona i služi da se u tački A dobije proizvoljna željena vrednost napona kada je napon napajanja dat i fiksna (ne može se proizvoljno birati).



Struja I koja protiče kroz granu sa otpornicima jednaka je (videti jednačinu (2)):

$$I = \frac{5V}{R_1 + R_2} \quad (3)$$

Na osnovu Omovog zakona (1) i poznatog izraza za vrednost struje može se odrediti napon na otporniku R_2 koji zapravo predstavlja traženi napon u tački A. Dakle, izraz za napon u ovoj tački glasi:

$$U_A = I \cdot R_2 \quad (4)$$

Kombinujući (3) i (4), dobijamo izraz za napon u tački A:

$$U_A = \frac{R_2}{R_1 + R_2} \cdot 5V \quad (5)$$

Iz priložene formule se vidi zašto ovo kolo nosi naziv razdelnik napona. Napona u tački A predstavlja podeljen napon od 5V koji je proporcionalan odnosu otpornosti R_1 i $R_1 + R_2$. Što je otpornost R_2 veća napon u tački A je bliži naponu od 5V dok ukoliko je veća otpornost R_1 napon je bliži naponu GND, odnosno 0V. U slučaju da su u kolu jednake otpornosti, napon od 5V se deli na pola i u tački A imamo napon od 2.5V.

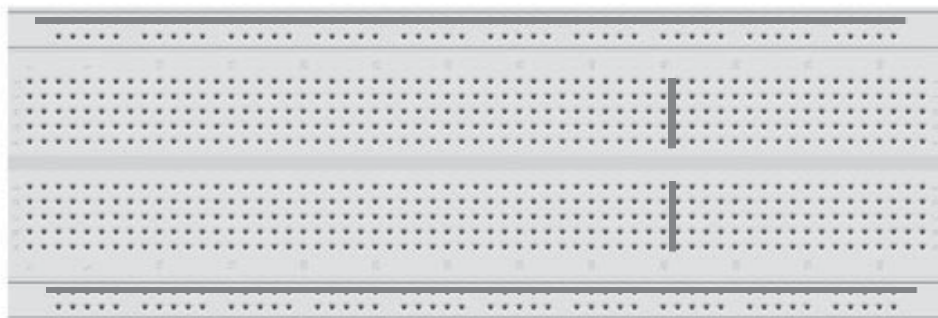
Najčešće je otpornost otpornika konstanta. Međutim, postoje otpornici čija se otpornost menja usled raznih spoljašnjih faktora (temperatura, pritisak, razni mehanički faktori). U električnim kolima se često koristi jedan takav otpornik, čija se otpornost menja mehaničkim delovanjem, tj. pomeranjem mehaničkog dela otpornika (najčešće obrtanjem). Takav otpornik se naziva potencijometar. Električni simbol kao i fizički izgled potencijometra su prikazani na slici 3 - 4.



Slika 3 - 4 - Električni simbol potencijometra (levo) i njegov fizički izgled (desno)

3.2. POVEZIVANJE KOMPONENTI POMOĆU PROTOBORDA

Protoboard predstavlja alat za kratkotrajno prototipsko sklapanje električnih kola. Njegov zadatak je da napravi trenutne veze između pojedinih elektronskih komponenti unutar sistema. Tačnije, zadatak protoborda jeste da napravi kratke spojeve između krajeva električnih komponenti i da na taj način formira električno kolo. Tipičan izgled protoborda dat je na slici 3 - 5.



Slika 3 - 5 - Protobord

Nizovi tačkica koje se nalaze paralelno crvenim linijama su kratko spojeni. Spajanje susednih tačkica koje nisu spojene na protobordu vrši se pomoći žica pod nazivom kratkospajачи. Kratkospajачи su prikazani na slici slici 3 - 6. U zavisnosti od oblika krajeva postoje tri tipa kratkospajача:

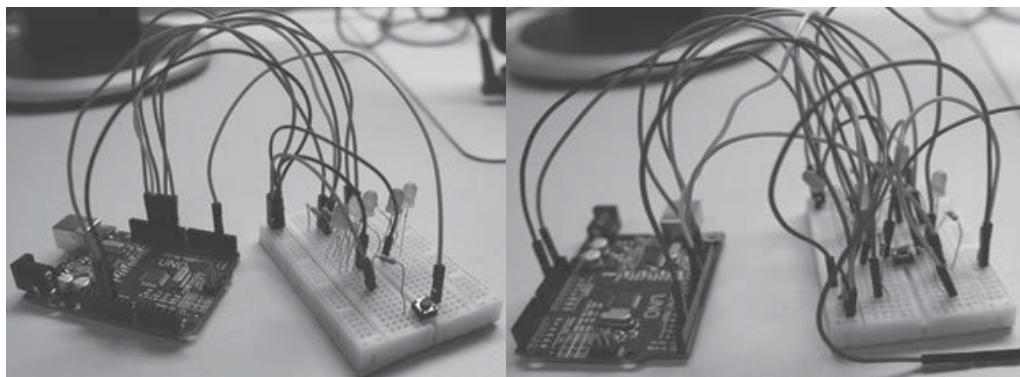
1. Muško-muški
2. Muško-ženski
3. Žensko-ženski



Slika 3 - 6- Kratkospajачи (muško-ženski)

Kratkospajajući mogu imati različite dužine. Dobra praksa jeste da se kraći kratkospajajući koriste za povezivanje komponenti na protobordu dok se duži kratkospajajući koriste za dovođenje napajanja i signala sa drugih platformi. Takođe, treba smisleno koristiti boje kratkospajача. Najbolje je usvojiti neki standard korišćenja boja i njime se voditi kroz projektovanje sistema. Na primer crvena boja se koristi za napajanje, crna za masu, zelena za zelene diode, itd.

Korišćenjem protoborda i kratkospajача, jedno isto električno kolo može se fizički realizovati na veći broj različitih načina. Međutim, pri povezivanju treba voditi računa o preglednosti kola. Realizacija kola treba da je takva da kolo bude pregledno i da se lako mogu uočiti eventualne greške i napraviti neophodna merenja. U ovoj knjizi, za sve primere kola date su realizacije sa protobordom koje su pregledne i vizuelno jasne. Slika 3 - 7 daje uporedni pregled istog električnog kola koje je pregledno i nepregledno realizovano.



Slika 3 - 7 - jasno, smisleno i pregledno realizovano kolo (levo), nepregledna realizacija (desno)

3.3. MERENJE NAPONA I STRUJA U ELEKTRIČNOM KOLU

Pored matematičke analize električnih kola često je neophodno izvršiti određene provere električnim merenjima. U tu svrhu se koristi multimeter koji je prikazan na slici 3 - 8. U ovoj knjizi su obrađene osnovne metode rada sa ovim mernim instrumentom, dok se detaljnije uputstvo za kroišćenje može naći na internetu [21].

Displej

Tasteri za konfiguraciju

Selektor modova rada

Priključci za kablove



Slika 3 - 8 - Prednja strana multimetra

Ovaj multimeter pruža različite mogućnosti merenja i testiranja. Obrtnim selektorom koji se nalazi u centralnom delu ovog mernog instrumenta biraju se različiti modovi rada u zavisnosti od toga šta se meri ili proverava. Neki od mogućih modova rada su:

- Merenje naizmeničnih napona
- Merenje jednosmernih napona
- Merenje otpornosti

- Testiranje dioda
- Testiranje kratkog spoja

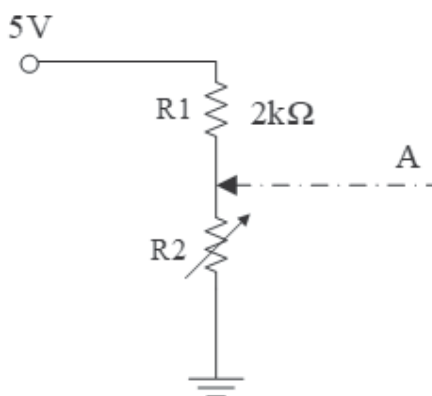
U zavisnosti od odabranog režima rada vrši se povezivanje kablova multimetra na odgovarajuće ulaze tako da simbol pored izabranog režima rada odgovara simbolu na koji povezujemo kablove.



Usled duže pauze u korišćenju instrumenta za vreme merenja, može doći do njegovog isključenja. U tom slučaju je neophodno obrtni selektor vratiti u poziciju OFF, a nakon toga ga ponovo vratiti u željeni režim rada.

Primer 3.1. - Merenje jednosmernog napona

- *Merenje jednosmernih napona u kolu biće objašnjeno na primeru prikazanom na slici 3 - 9. Zadatak je da se izmeri napon u tački A dok se pomera položaj potencijometra (menja se otpornost R_2). Potrebno je uraditi sledeće:*
1. Podesiti mod rada (Slika 3 - 10)
 2. Priključiti kablove (Slika 3 - 11)
 3. Meriti napon (Slika 3 - 12)



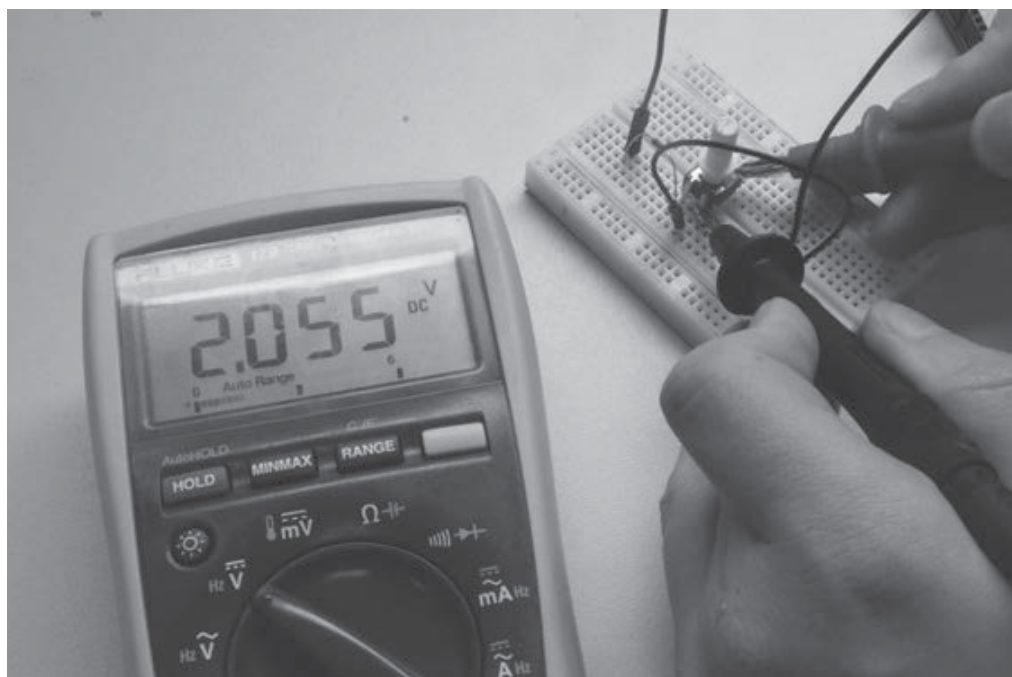
Slika 3 - 9 - Razdelnik napona sa potencijometrom



Slika 3 - 10 – Multimetar u odgovarajućem režimu rada



Slika 3 - 11 – Način povezivanja kablova sa multimetrom



Slika 3 - 12 - Merenje traženog napona

Primer 3.2. - Testiranje kratkog spoja

Ovaj primer ima za cilj da demonstrira mod testiranja kratkog spoja. Ovaj mod je veoma koristan u fazi sklapanja i testiranja električnog kola kada je neophodno proveriti da li se određene tačke, ili komponente, u kolu ispravno spojene ili da li je došlo do kratkog spajanja tačaka (ili komponenti) koje ne bi smele da budu kratko spojene. Koraci kojima se realizuje ova provera su sledeći:

1. Podesiti multimeter u mod koji proverava kratke spojeve (Slika 3 - 13).
2. Priljučiti kablove (Slika 3 - 11).
3. Međusobno spojiti krajeve kablova za merenje. Trebalo bi da se čuje pištanje (Slika 3 - 14).
4. Instrument je spreman. Proveriti da li postoji kratak spoj (Slika 3 - 15).



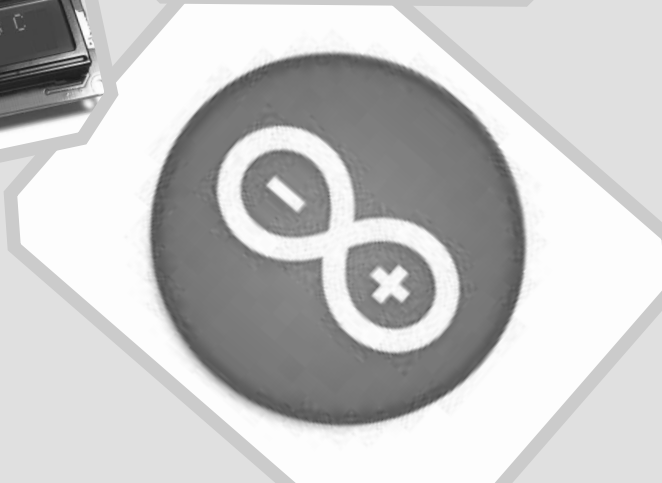
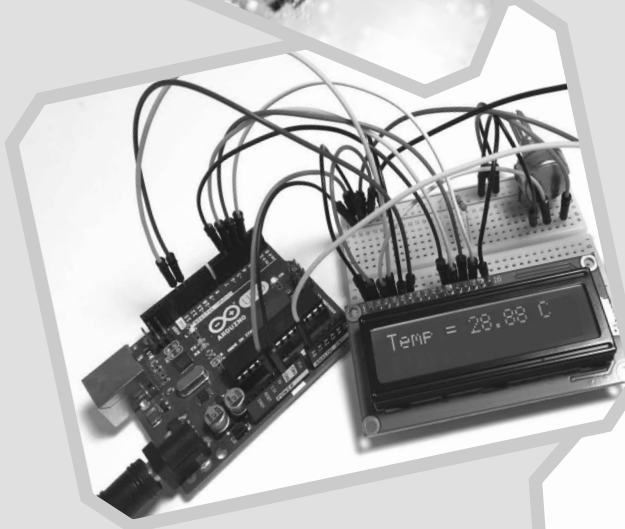
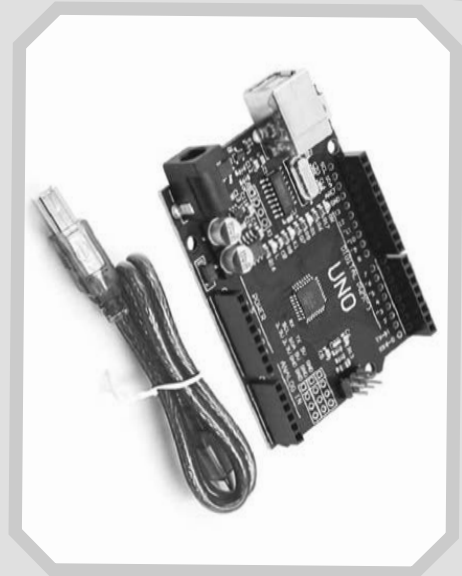
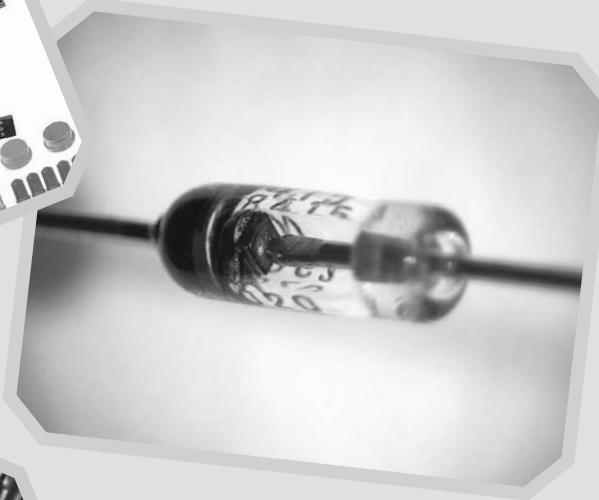
Slika 3 - 13 - Unimer u traženom režimu rada

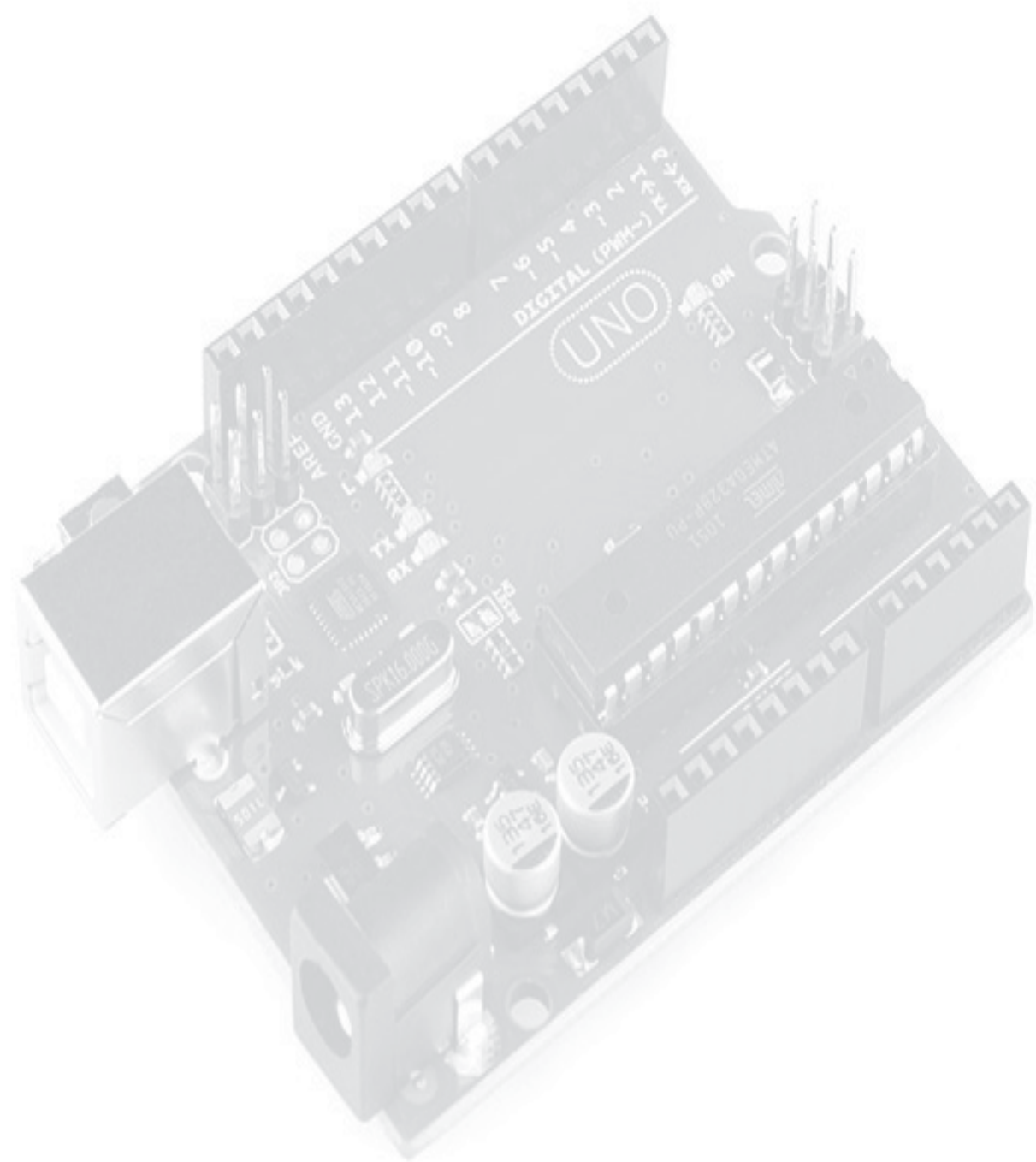


Slika 3 - 14 - Provera da li se čuje pištanje



Slika 3 - 15 - Ispitivanje kratkog spoja





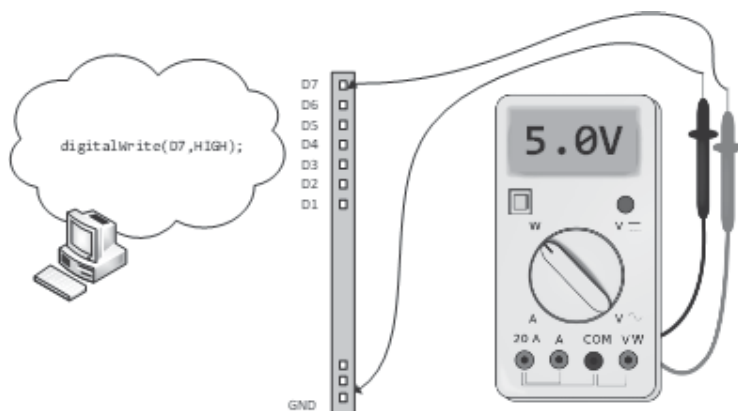
4.DIGITALNI IZLAZI I RAD SA DIODAMA

Digitalni sistemi koriste binarne brojeve (najčešće označene sa 0 i 1) za obradu, prenos skladištenje ili prikazivanje informacija. Digitalni sistemi uglavnom rade na principu prepoznavanja logičkih nivoa. Najčešće se koriste dva logička nivoa i to **visok naponski nivo ili nivo logičke jedinice (HIGH)** i **nizak naponski nivo ili nivo logičke nule (LOW)**. Kod Arduino mikrokontrolera pod nivoom logičke jedinice često se podrazumeva napod od 5V dok ponekad pri komunikaciji sa nekim modulima taj napon iznosi 3.3V. Nivo logičke nule podrazumeva napon od 0V ili, vrlo retko, neki negativan napon.

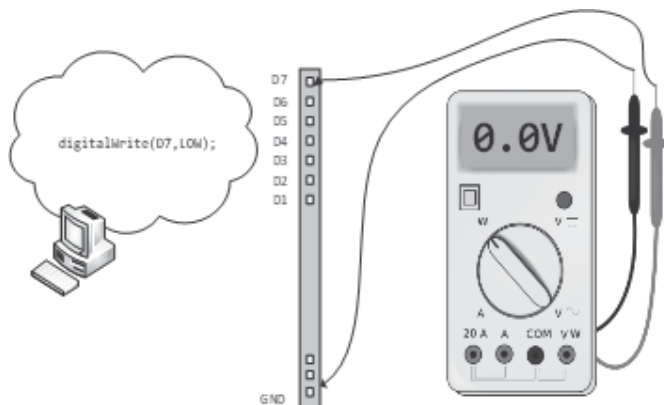
4.1. DIGITALNI IZLAZI

Digitalni izlazi se koriste u cilju kontrole uređaja koji imaju dva stanja: uključen ili isključen. Pri radu sa Arduino hardverskom platformom često se vrši ispis logičke nule ili jedinice na odgovarajuće digitalne portove. Logičke nule i jedinice iz softvera u fizičkom svetu predstavljaju određene naponske nivoe. To znači da ukoliko se softverski na neki port upiše logička jedinica („1“), a potom izmeri napon na tom portu, vrednost tog napona će biti 5V, dok ukoliko se na neki port upiše logička nula („0“), tada će napon na tom portu biti 0V.

Slika 4 - 1 Napon na digitalnom pinu u slučaju softverski upisane „1



Slika 4 - 1 Napon na digitalnom pinu u slučaju softverski upisane „1



Slika 4 - 1a - Napon na digitalnom pinu u slučaju softverski upisane „0“

4.2. DIODA

Diode spadaju u osnovne elektronske komponente i često se sreću u električnim kolima. Njihovo glavno svojstvo jeste da provode struju samo u jednom smeru. Dioda provodi struju kada se na nju dovede odgovarajući napon. Način predstavljanja dioda u elektronskim šemama, kao i fizički izgled jedne diode dati su na slici 4 - 2. Smer strelice na električnoj šemi označava smer u kome diode provode struju. Iz tela diode izlaze dva provodnika. Jedan se zove anoda a drugi katoda. Fizičke realizacije dioda su takve da se *anoda* i *katoda* mogu lako međusobno razlikovati. Najčešće je žica koja je povezana na anodu duža od žice povezane na katodu (slika 4 - 2).

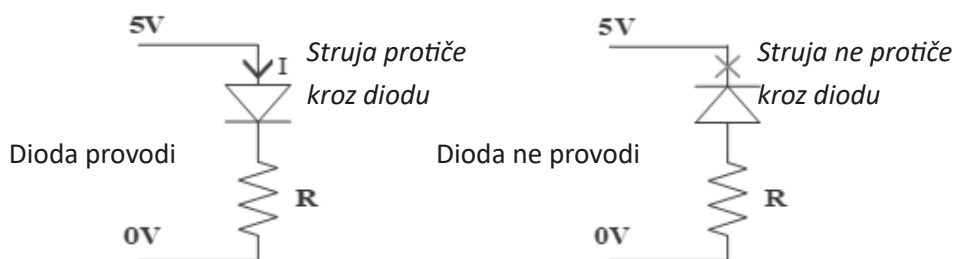


Slika 4 - 2 - Simbol diode (levo) i fizički izgled diode (desno)

Da bi dioda provela struju u naznačenom smeru potrebno je da napon na anodi bude veći od napona na katodi za određenu razliku (tipično 0.7V). Ukoliko je napon između katode i anode suviše mali ili negativan, dioda neće provoditi struju. Ovo svojstvo diode ilustrovano je na slici 4 - 3.

Ako se dioda u ostatak sistema uključi tako da provodi struju, struja koja teče u smeru strelice može biti izuzetno velika (jer je interna otpornost diode jako mala) i može dovesti do sagorevanja diode. Zbog toga se redno sa diodom često vezuje otpornik koji ima zadatak da ograniči struju diode. Drugim rečima, veoma retko se dioda u ostatak sistema povezuje samostalno a da zajedno sa njom ne ide otpornik.

Dioda sa otpornikom se u kolu najlakše može posmatrati kao prekidač koji se uključuje kada se na taj prekidač dovodi odgovarajući napon. U Arduino realizacijama taj napon obično predstavlja izlaz digitalnog porta na Arduino platformi.



Slika 4 - 3 - Uslov provođenja diode

4.2.1. Light Emitting (LE) dioda

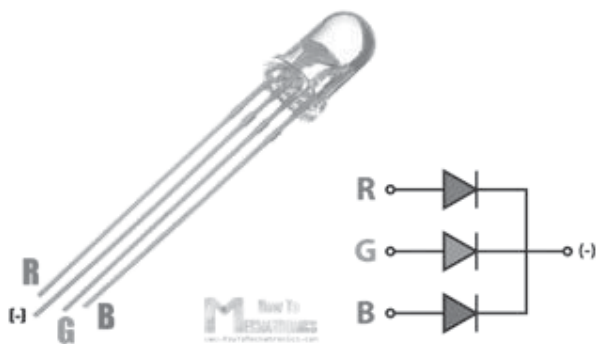
Postoje različiti tipovi dioda, ali kroz primere u ovom priručniku najviše se koristi Light Emitting (LE) dioda. Ona ima svojstvo da kada provodi struju emituje svetlost i kao takva našla je primenu u raznim aplikacijama. Različitim kombinacijama hemijskih elemenata koji čine strukturu LE diode ostvaruje se da ona emituje svetlost određenje boje. Električni simbol LE diode, kao i izgled LE dioda različitih veličina i boja prikazani su na slici 4 - 4.



Slika 4 - 4 - Električni simbol LE diode (levo) i LE diode različitih boja (desno)

4.2.2. Red Green Blue (RGB) dioda

Ovu diodu zapravo čine tri diode različitih boja i to : crvene, plave i zelene, koje su integrisane u jednom kućištu. Svetlost koju emituje ova dioda može biti posledica različitih kombinacija trenutno aktivnih dioda unutar RGB diode. Izgled, kao i raspored pinova ove diode prikazani su na slici 4 - 5.



Slika 4 - 5 - RGB Dioda

Postoje dva tipa RGB dioda: sa zajedničkom katodom i sa zajedničkom anodom. Razlikuju se po tome da li diode dele katodu ili anodu. Na slici 4 - 5 je prikazana varijanta RGB diode sa zajedničkom katodom koja se češće sreće u praksi.

Crvena, plava i zelena boja se mogu dobiti aktiviranjem samo jednog od odgovarajućih pinova, dok se ostale boje mogu postići istovremenim aktiviranjem različitih boja. Kombinujući intenzitet i različite osnovne boje možemo dobiti širok spektar boja.

Pored LE i RGB diode postoje još i Zener diode, obične prekidačke diode, šotki diode. One služe za građenje složenijih električnih kola i nisu predmet ovog priručnika.

4.3. PRIMERI

Primeri koji slede imaju zadatak da čitaocu približe i objasne upotrebu LE i RGB dioda, kao i da ilustruju rad sa digitalnim izlazima Arduina.

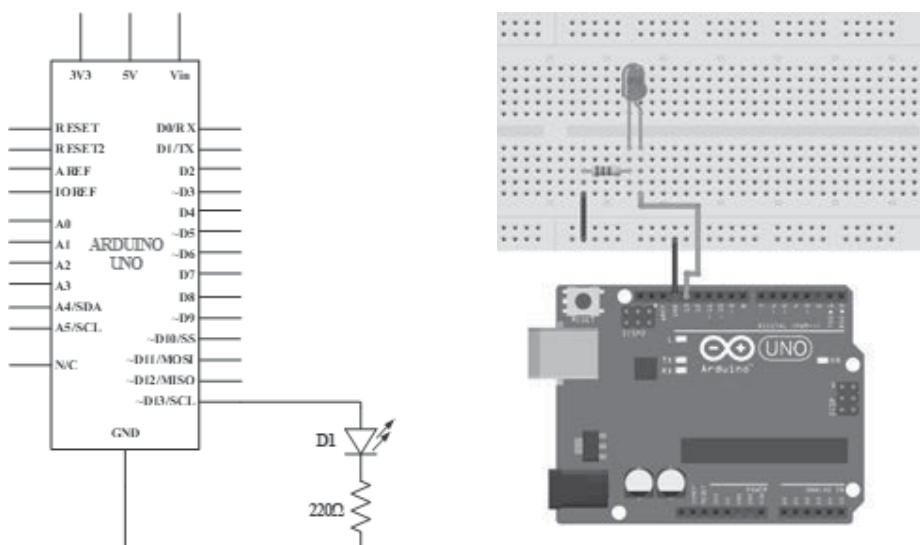
Primer 4.1. - Treperenje diode

- *Realizovati Arduino sistem koji koristi jednu diodu. Dioda treba da treperi sa periodom od 2 sekunde. Napisati program koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Da bi realizovali ovaj sistem neophodni su sledeći elementi:

- Arduino UNO
- Otpornik 220Ω
- Dioda
- Protobord
- 3 MM kratkospajача



Slika 4 - 6 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 4.1.

Električna šema povezivanja, kao i izgled fizičke realizacije sistema prikazani su na slici 4 - 6. Dioda je povezana tako da je njena anoda (duža nogica diode) spojena sa digitalnim pinom Arduina broj 13 (D13), dok je katoda (kraća nogica) preko otpornika od 220Ω spojena na GND. Dioda se periodično uključuje i isključuje naizmeničnim upisom logičke jedinice i nule na digitalni pin 13. Nakon upisa logičke jedinice, na pinu 13 se javlja napon od 5V. Usled dovoljne razlike potencijala na krajevima diode, kroz diodu i otpornik će početi da teče struja usled čega će se dioda uključiti (počeće da emituje svetlost). Nakon isteka zahtevanog vremenskog intervala, upisuje se logička nula na pin 13, što za posledicu ima dovođenje napona od 0V na taj pin. Usled male razlike napona na krajevima diode, struja prestaje da teče kroz nju, te se ona gasi (ne emituje više svetlost).

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
int PinNo = 13;
int DelayTime = 500;
int DiodeStatus = LOW;

void setup() {
    // Initialize diode pin as output pin
    pinMode(PinNo, OUTPUT);
    // Set initial diode state
    digitalWrite(PinNo, DiodeStatus);
}

void loop() {
    // Change diode status
    DiodeStatus = !DiodeStatus;
    // Write desired output value to desired pin
    digitalWrite(PinNo, DiodeStatus);
    // Delay program execution for DelayTime ms
    delay(DelayTime);
}
```

Digitalni pinovi mogu da se koriste kao ulazni i kao izlazni. Inicijalizacija smera u kome će da rade pinovi vrši se pomoću funkcije

pinMode(PinNo, Direction);

- | | |
|-----------|----------------------------------|
| PinNo | – Broj pina |
| Direction | – Postoje dve mogućnosti : |
| - INPUT | – ako se pin koristi kao ulazni |
| - OUTPUT | – ako se pin koristi kao izlazni |

Digitalna vrednost pina se upisuje pozivom funkcije

digitalWrite(PinNo, Value);

- | | |
|-----------|-------------------------|
| PinNo | – Broj pina |
| Direction | – Vrednost HIGH ili LOW |

Izvršavanje programa (naredne linije koda) se može odložiti na određeno vreme pozivom funkcije

delay(delayTimeInms);

delayTimeInms – Koliko vremena (izraženog u milisekundama) želimo da odložimo izvršavanje sledeće komande

Promenljiva *DiodeStatus* sadrži trenutno stanje pina na koji je priključena dioda koji može biti **LOW** (pin ima nizak napon 0V), što znači da je dioda isključena ili **HIGH** (pin ima visok napon 5V), što znači da je dioda uključena. Inicijalno je dioda isključena. U funkciji *setup*, digitalni pin na koji je priključena dioda sa otpornikom je konfigurisan kao izlazni pin. Zatim je ovaj pin inicijalizovan na logičku 0. To je urađeno pozivom funkcije *digitalWrite* koja kao prvi argument prima broj pina dok je drugi argument vrednost koja treba da bude upisana na dati pin. Ovo je urađeno da bi pri pokretanju Arduina dioda uvek bila isključena.

Funkcija *loop* se ponavlja sve dok Arduino ima napajanje. U prvoj naredbi unutar glavne programske petlje *loop* vrši se promena trenutnog stanja diode primenom operacije inverzije na promenljivu ***DiodeStatus***. Operator inverzije u Arduino kodu je operator „!” i on izaziva promenu stanja promenljive (ukoliko je vrednost promenljive bila HIGH usled primene operatora „!” kao rezultat dobijamo vrednost LOW). Ova operacija je moguća samo nad promenljivim koje imaju dve vrednosti (*true/false*, *HIGH/LOW*).

Nakon što se invertuje trenutni status diode i zapamti novo stanje diode u promenljivoj *DiodeStatus*, neophodno je upisati tu vrednost na digitalni port na koji je povezana dioda. To se postiže pozivom funkcije *digitalWrite*.

Na kraju koda poziva se funkciju *delay* koja ima zadatak da odloži izvršavanje sledeće linije koda (što je u ovom slučaju prva linija koda u *loop* funkciji) za *DelayTime* milisekundi. Naravno, promenu periode je moguće regulisati promenom vrednosti promenljive *DelayTime*.



Ukoliko posle programiranja Arduino hardverske platforme dioda ne treperi, verovatno je došlo do greške u povezivanju komponenti. U ovom slučaju treba ukloniti USB kabl sa Arduina i vizuelno ispitati kolo. Treba proveriti da li su sve komponente povezane i da li je dioda povezana tako da je duži pin diode povezan sa pinom broj 13 Arduino platforme. Često se dešava da se u brzini dioda pogrešno okrene i da zbog toga kolo ne funkcioniše kako treba.

Primer 4.2. – Dve diode koje se naizмениčno pale

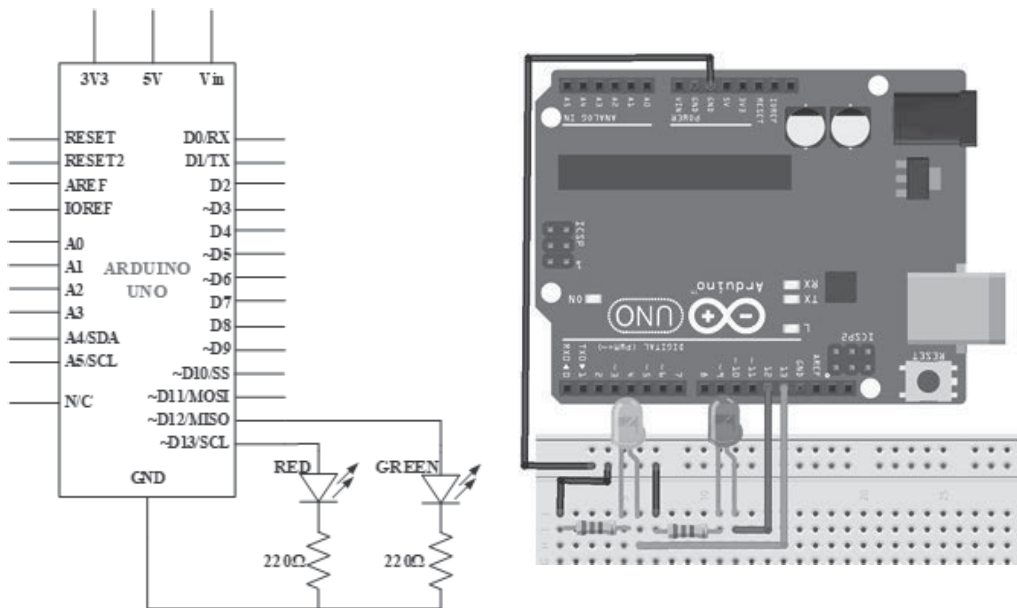
- Realizovati Arduino sistem koji koristi jednu crvenu i jednu zelenu diodu. Napisati program koji naizмениčno uključuje svaku od dioda na 0.5 sekundi. Napisati kod i praktično potvrditi uspešnost realizovanog sistema.

Rešenje:

Za realizaciju ovog sistema potrebno je:

- Arduino UNO
- Dva otpornika 220Ω
- Dve diode (crvena i zelena)
- Protobord
- 5 MM kratkospajča

Električna šema na osnovu koje treba povezati sistem kao i fizički izgled sistema u Arduino okruženju su prikazani na slici 4 - 7. Na osnovu prikazanih šema možemo zaključiti da se one znatno ne razlikuju od šema iz prethodnog zadatka izuzev što je dodata još jedna dioda.



Slika 4 - 7 - Električna šema (levo) i fizički izgled (desno) sistema iz primera 4.2.

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
int RedDiodePinNo = 13;
int GreenDiodePinNo = 12;
int DelayTime = 500;
int RedDiodeStatus = LOW;
int GreenDiodeStatus = HIGH;
void setup(){
    // Initialize diode pins as output pins
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT);
    // Set initial states of the digital outputs
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
}
void loop() {
    // Change diode pin state values
    RedDiodeStatus = !RedDiodeStatus;
    GreenDiodeStatus = !GreenDiodeStatus;
    // Delay program execution for DelayTime ms
    delay(DelayTime);
}
```

Na samom početku (van funkcija *setup* i *loop*) definisane su promenljive koje se koriste u ostatku programa. Najpre su definisane promenljive *RedDiodePinNo* i *GreenDiodePinNo* koje označavaju brojeve pinova na koje su priključene crvena i zelena dioda respektivno. Nakon toga definisana je promenljiva *DelayTime* koja čuva informaciju o periodu promene aktivne diode. Promenljive *RedDiodeStatus* i *GreenDiodeStatus* čuvaju informaciju o trenutnoj vrednosti na pinu na koji je povezana odgovarajuća dioda. Inicijalno crvena dioda je isključena, dok je zelena uključena.

Nakon deklaracije promenljivih ulazi se u funkciju *setup* koja ima zadatak da inicijalizuje sistem. Funkcionalnosti koje implementira ova funkcija su iste kao iz prethodnog zadatka samo su sada primenjene na obe diode.

Unutar glavne programske petlje implementirane su iste funkcionalnosti kao u prethodnom primeru ali na obe diode.

Primer 4.3. – RGB Dioda

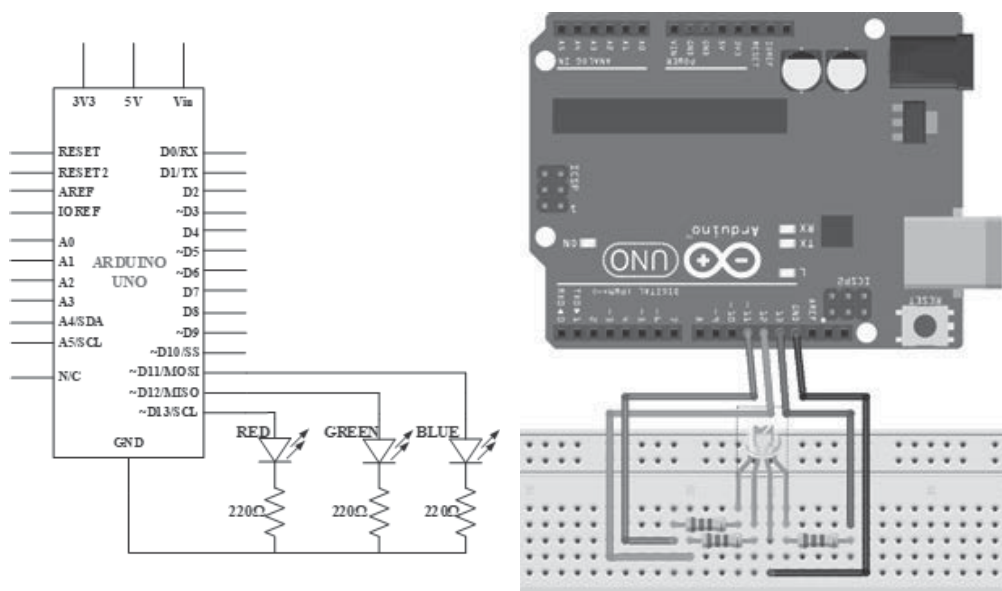
- Realizovati Arduino sistem koji koristi jednu RGB diodu. Napisati program koji realizuje sledeću sekvencu: 1 sekund dioda svetli crveno, 0.5 sekundi zeleno i 1 sekund plavo. Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizovanog sistema.

Rešenje:

Za realizaciju ovog sistema potrebne su sledeće komponente:

- Arduino UNO
- 3 otpornika od 220Ω
- RGB dioda
- Protobord
- 4 MM Kratkospajča

Električna šema i fizička realizacija sistema prikazani su na slici 4 - 8.



Slika 4 - 8 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 4.3.

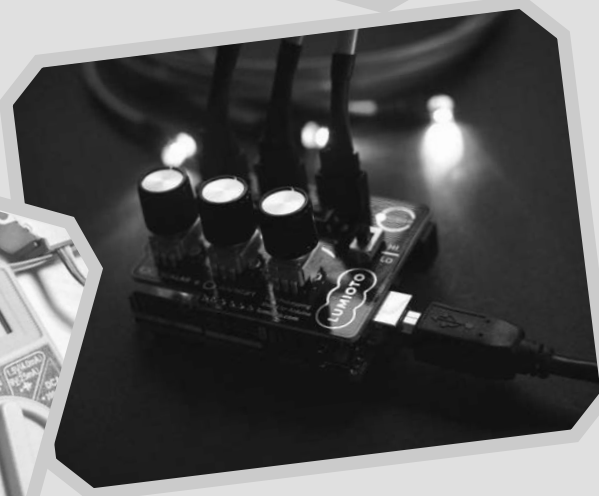
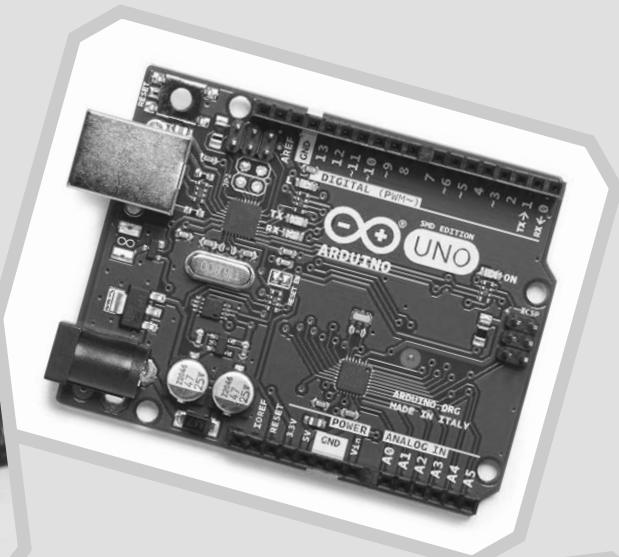
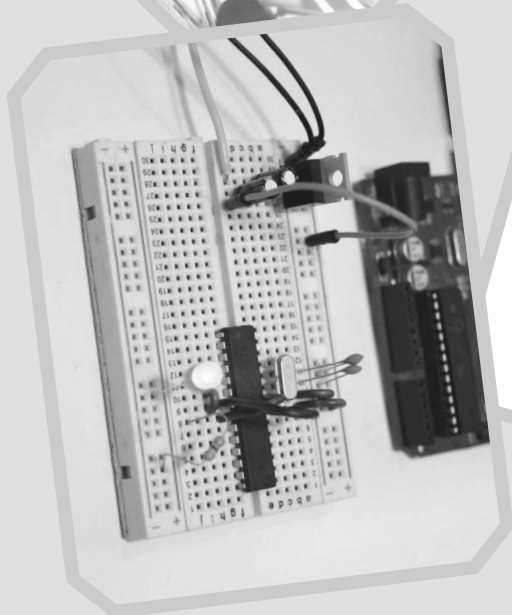
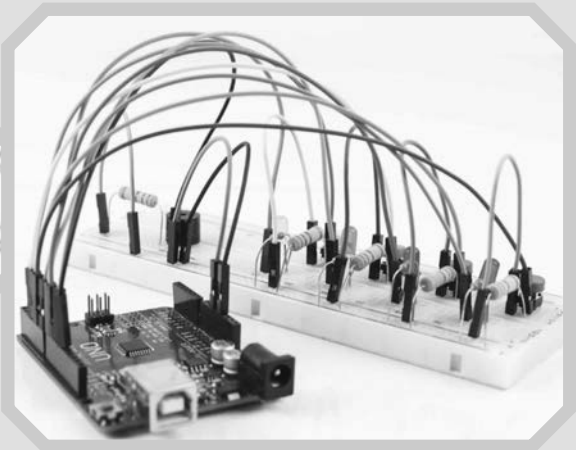
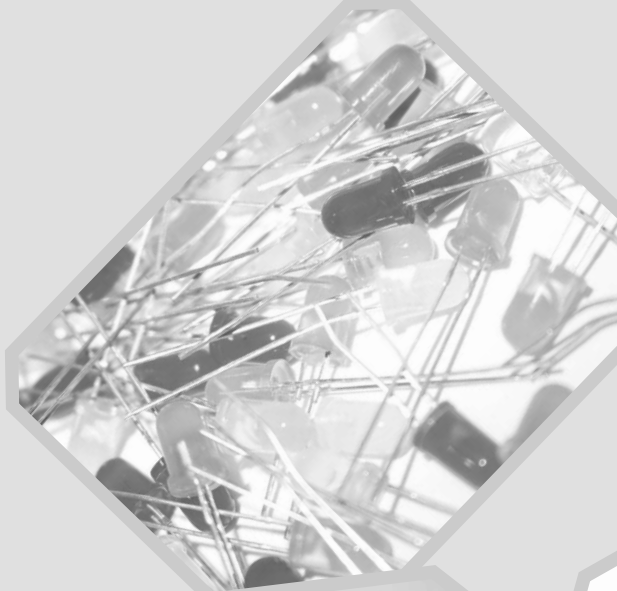
Rad sa RGB diodom se suštinski može posmatrati kao rad sa tri odvojene diode: crvenom, zelenom i plavom. Zbog toga su tri kraja RGB diode preko otpornika od 220Ω povezana na tri različita pina 13, 12 i 11 tako da odgovaraju crvenoj, zelenoj i plavoj diodi.

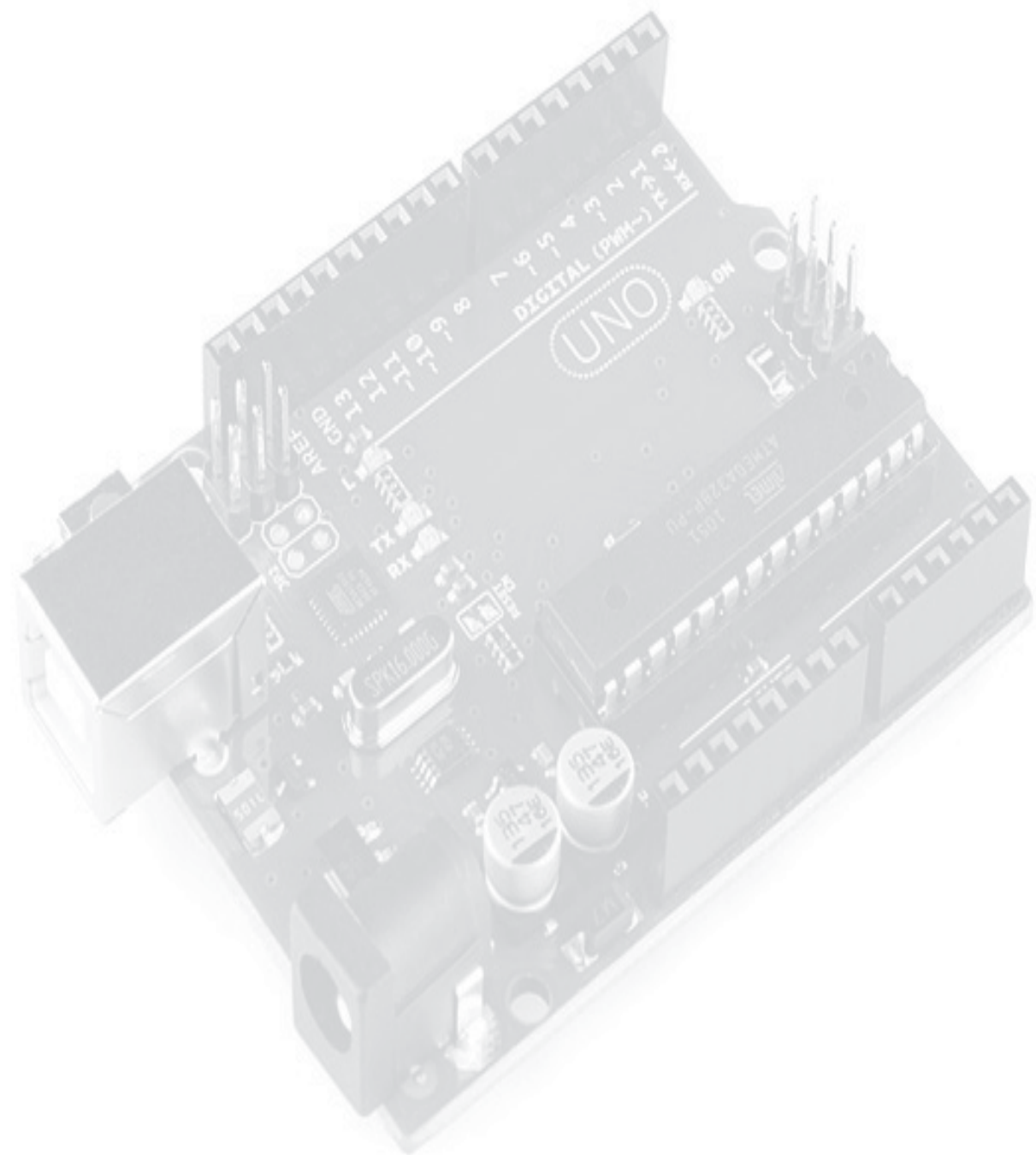
Na početku su svi pinovi postavljeni na vrednost „0“, što za posledicu ima isključenje svih dioda. Zatim se uključi pin 13 na koji je priključena crvena dioda i drži se uključenim zahtevano vreme, a zatim nakon isteka ovog vremena, dioda se isključuje upisom logičke nule na pin 13. Nakon što se isključi crvena dioda, uključuje se zelena dioda upisom logičke jedinice na pin 12. Ova vrednost ostaje na pinu 12 zahtevano vreme, nakon čega se pin postavlja na vrednost „0“. Slična ideja se primenjuje i za zelenu diodu. Kod koji realizuje ovu ideju je sledeći:

```
int RedDiodePinNo = 13;
int GreenDiodePinNo = 12;
int BlueDiodePinNo = 11;
int RedDelayTime = 1000;
int GreenDelayTime = 500;
int BlueDelayTime = 1000;
void setup(){
    // Initialize diode pins as output pins
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT);
    pinMode(BlueDiodePinNo, OUTPUT);
    // Disable RGB diode light
    digitalWrite(RedDiodePinNo, LOW);
    digitalWrite(GreenDiodePinNo, LOW);
    digitalWrite(BlueDiodePinNo, LOW);
}
void loop() {
    //-----RED DIODE-----
    //Turn on red diode
    digitalWrite(RedDiodePinNo, HIGH);
    //Wait for RedDelayTime
    delay(RedDelayTime);
    //Turn off red diode
    digitalWrite(RedDiodePinNo, LOW);
    //-----GREEN DIODE-----
    //Turn on green diode
    digitalWrite(GreenDiodePinNo, HIGH);
    //Wait for GreenDelayTime
    delay(GreenDelayTime);
    //Turn off green diode
    digitalWrite(GreenDiodePinNo, LOW);
    //-----BLUE DIODE-----
    //Turn on blue diode
    digitalWrite(BlueDiodePinNo, HIGH);
    //Wait for BlueDelayTime
    delay(BlueDelayTime);
    //Turn off blue diode
    digitalWrite(BlueDiodePinNo, LOW);
}
```

4.4. ZADACI ZA SAMOSTALAN RAD

1. Modifikovati kod u primeu 4.1, tako da dioda treperi sa 1,2 i 3 sekunde.
2. Modifikovati program u primeru 4.2, tako da se ostvari sledeća sekvenca: crvena dioda svetli 0.5 sekundi a zatim obe diode budu ugašene 0.5 sekundi, zatim zelena dioda svetli 0.5 sekundi a onda obe diode budu ugašene 0.5 sekundi, nakon čega se cela sekvenca ponavlja.
3. Realizovati primer 4.2 uz pomoć RGB diode, a primer 4.3 uz pomoć crvene, plave i zelene LE diode.
4. Modifikovati kod iz primera 4.3, tako da svaka od tri boje (crvena, zelena i plava) budu aktivne po 6 sekundi, pri čemu dok je boja aktivna dioda treba da trepće u toj boji sa periodom 0.5 sekundi. Dakle, 6 sekundi treperi crvena pa 6 sekundi treperi zelena pa 6 sekundi treperi plava nakon čega se sekvenca ponavlja.





5. DIGITALNI ULAZI I RAD SA TASTERIMA

Cilj ovog poglavlja je da uvede osnove u radu sa digitalnim pinovima kada se oni koriste kao ulazni. U ovom slučaju se vrednost na pinu ne postavlja softverski (unutar koda) pozivom odgovarajućih funkcija, već se vrednost na tom pinu postavlja hardverski, tj. postoji kolo izvan mikrokontrolera koje definiše naponski nivo na pinu. Dati naponski nivo se tumači od strane Arduino kontrolera kao logička „0“ ili logička „1“.

Digitalni pinovi se najčešće koriste kao ulazni u kombinaciji sa tasterima, pa su oni i glavna tema ovog poglavlja.

5.1. TASTERI

Tasteri su elektromehaničke komponente koje služe za trenutno uspostavljanje ili prekidanje strujnog kola. Prekidanje ili uspostavljanje električnog kola se odvija mehaničkim pritiskom na površinu tastera. Električni simbol tastera i njegov fizički izgled prikazani su na slici 5 - 1.



Slika 5 - 1 - Električni simboli (levo i u sredini) i fizički izgled (desno) tastera

Tasteri imaju dva stanja: otvoren i zatvoren. Pod otvorenim tasterom se podrazumeva taster koji nije pritisnut, tj taster koji nije uspostavio protok struje na svojim krajevima (ponaša se kao otvorena veza). Pod zatvorenim tasterom se podrazumeva taster koji je pritisnut i kroz čije krajeve teče struja (ponaša se kao kratak spoj).

Tasteri se obično sa Arduino platformom povezuju putem digitalnog porta koji je prethodno inicijalizovan kao ulazni. Da li je taster pritisnut ili ne određuje napon na datom digitalnom pinu. Zbog toga je, pre pisanja koda, neophodno upoznati se sa hardverskom realizacijom sistema. Postoje dva moguća načina povezivanja tastera u kolo i oni su prikazani na slikama 5 - 2 i 5 - 3.

Na slici 5 - 2 levo usled pritiskanja tastera dolazi do kratkog spajanja pina DIN sa GDN naponom od 0V. Usled pritiska tastera koji je povezan kao na slici 5 - 2 desno, dolazi do spajanja pina DIN sa naponom od VCC odnosno 5V.

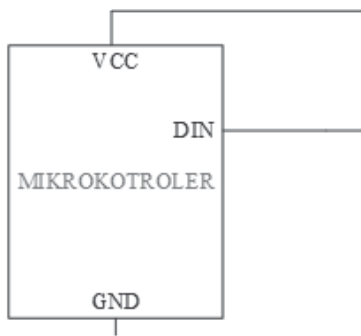
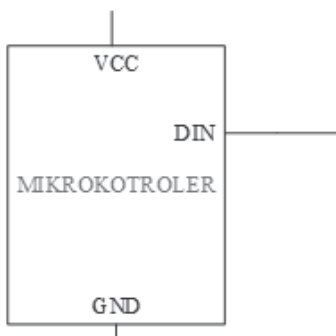
Kao posledicu ovake dve konfiguracije moguće je da kontroler čitanjem digitalnog pina DIN pročitati dve vrednosti i to :

- „0“ ako je konfiguracija kao na levoj slici
- „1“ ako je konfiguracija kao na desnoj slici

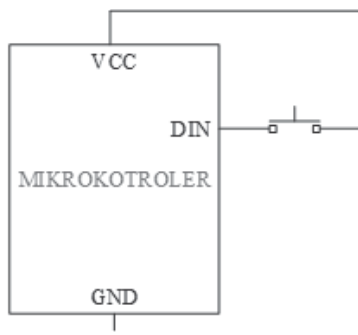
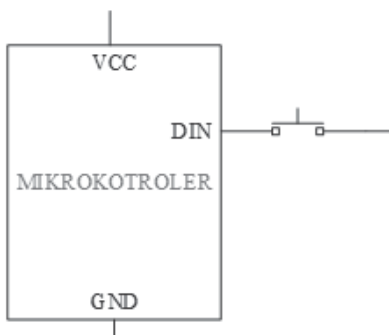
Kada taster nije pritisnut (slika 5 - 3) stanje na pinu nije definisano. To znači da ukoliko bi kontroler pokušao da pročita vrednost sa pina DIN kada taster nije pritisnut, rezultat tog čitanja ne bi bio poznat, što bi moglo dovesti do problema u funkcionisanju sistema.



Pri projektovanju sistema treba izbegavati povezivanje komponenti koje nemaju predvidljivo ponašanje jer mogu izazvati probleme. Zbog povezivanja sa drugim sistemima, prilikom projektovanja je neophodno voditi računa o tome da sistem koji se projektuje ima predvidivo ponašanje

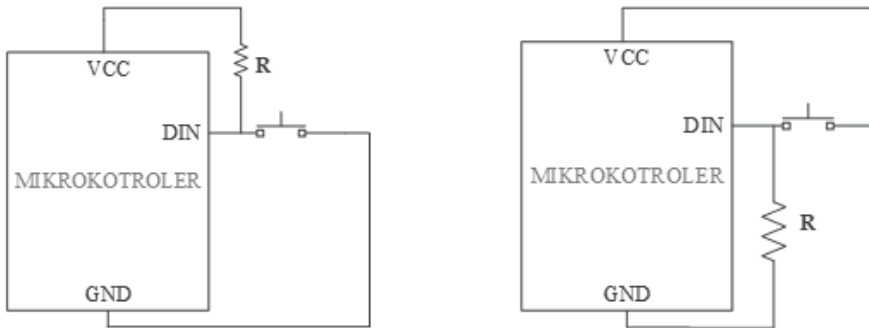


Slika 5 - 2 - Načini povezivanja tastera (taster pritisnut)



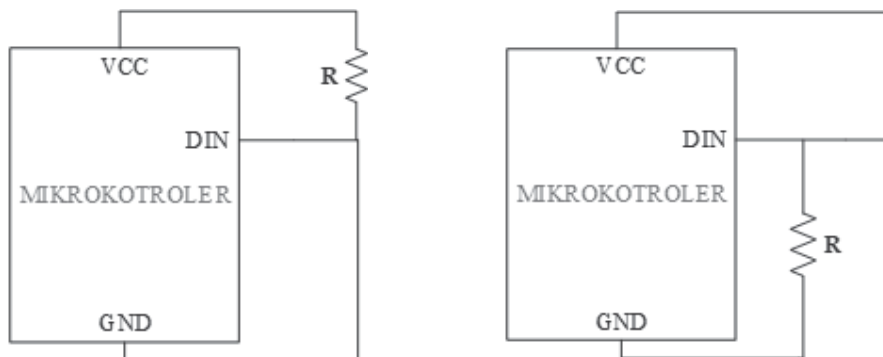
Slika 5 - 3 - Način povezivanja tastera (taster nije pritisnut)

Da bi se izbeglo nepredvidivo ponašanje sistema kada taster nije pritisnut, često se vrši hardverska inicijalizacija stanja na pinu na koji je povezan taster. Inicijalizacija se vrši pomoću otpornika koji se povezuje na suprotan naponski nivo od onog na koji je povezan taster. To znači da će u slučaju povezivanja tastera kao na slici 5 - 3 levo, otpornik biti povezan između digitalnog pina DIN i napona VCC, dok u slučaju povezivanja tastera kao na slici 5 - 3 desno, otpornik se povezuje između digitalnog pina DIN i napona GND. Ovakav način pvezivanja prikazan je na slici 5 - 4.



Slika 5 - 4 -Povezivanje tastera sa pull-up i pull-down otpornicima (taster nije pritisnut)

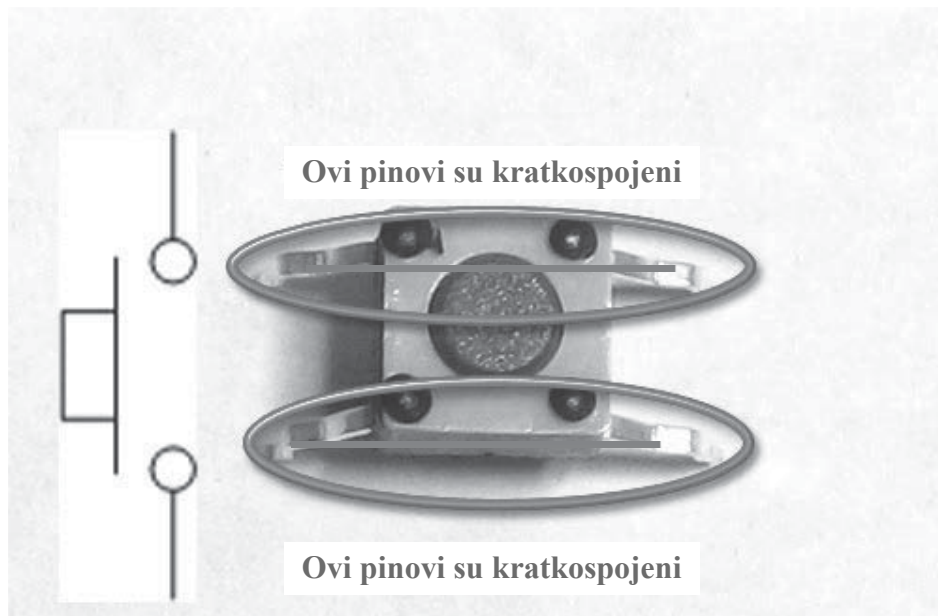
Načinom povezivanja kao na slici 5 - 4 omogućava se da sistem ima predvidivo ponašanje čak i kada taster nije pritisnut. Otpornik R na levoj slici se naziva *Pull-up* otpornik dok se otpornik na slici desno naziva *Pull-down*. *Pull-down* otpornik definiše „0“ kada taster nije pritisnut dok pull-up definiše „1“ . Kada je taster pritisnut situacija ostaje kao da otpornik nije dodat u kolo. Električna šema koja ilustruje Pull-up i Pull-down mehanizam u trenutku kada su tasteri pritisnuti prikazana je na slici 5 - 5.



Slika 5 - 5 - Povezivanje tastera sa pull-up i pull-down otpornicima (taster pritisnut)

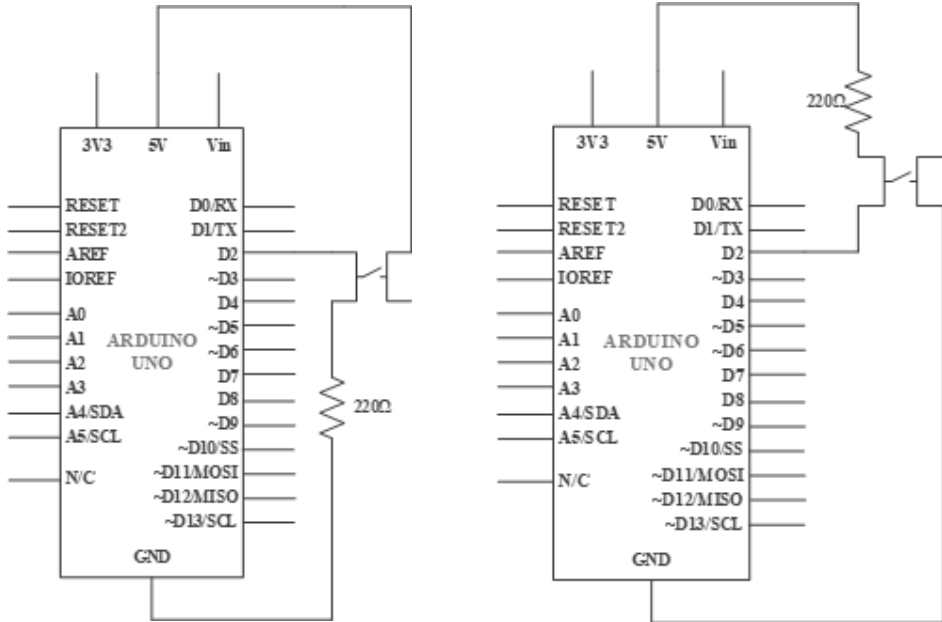


Taster sa slike 5 - 1 ima 4 nožice. Sa električnog simbola tastera jasno se vidi koje nožice su kratkospojene. Međutim, često prilikom prvih korišćenja tastera postoji problem prilikom određivanja koje nožice tastera su kratkospojene. Slika 5 - 6 prikazuje kratak spoj između nožica u fizičkoj realizaciji tastera.



Slika 5 - 6 - Ilustracija kratkog spoja među nožicama tastera

Na slici 5 - 7 prikazane su *pull-up* i *pull-down* realizacije tastera na Arduino hardverskoj platformi. Kroz ovaj priručnik biće korišćenja samo *pull-up* realizacija. Moguće je korišćenje i prikazanih ideja sa *pull-down* realizacijom, ali u tom slučaju treba zanti da pritisnuti taster rezultuje logičkom „1“, umesto logičkom „0“ koja se javlja u svim primerima koji su nadalje predstavljeni u ovom priručniku. To podrazumeva da kodovi koji su dati u ovom priručniku ne mogu direktno da funkcionišu u realizacijama sa *pull-down* otpornikom. Obrtanjem logike za čitanje vrednosti digitalnih ulaza, kodovi se mogu prilagoditi tako da rade u konfiguraciji sa *pull-down* otpornikom.



Slika 5 - 7 – Šema povezivanja tastera i Arduina sa Pull - Down (levo) i Pull - Up (desno) otpornikom

5.2. PRIMERI

Primeri koji slede imaju za cilj da detaljnije objasne načine korišćenja tastera i njihove upotrebe. Primeri takođe ilustruju rad sa digitalnim ulazima Arduina.

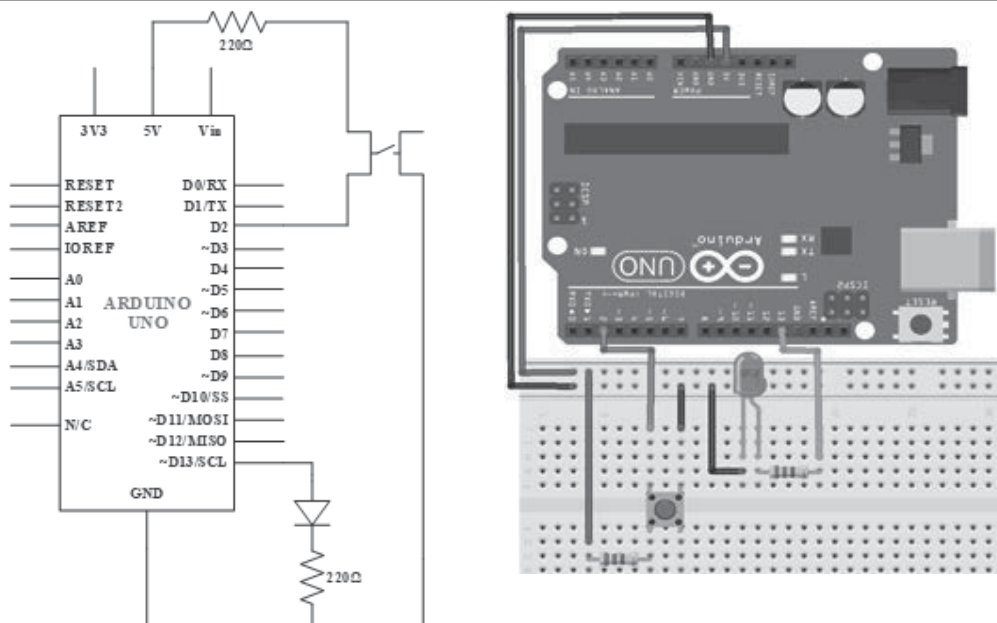
Primer 5.1. – Upotreba tastera

- *Realizovati Arduino sistem koji uključuje diodu ukoliko je taster pritisnut dok je u suprotnom dioda neaktivna. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Dva otpornika od 220Ω
- Dioda
- Protobord
- 7 MM kratkospajача
- Taster



Slika 5 - 8 - Električna šema (levo) i hardverska realizacija (desno) sistema iz primera 5.1.

Električna šema koja ilustruje način povezivanja navedenih komponenti, kao i fizička realizacija sistema prikazani su na slici 5 - 8.

Ideja koju treba realizovati je da se stalno čita vrednost sa pina broj 2 i da se pročitana vrednost invertuje i upiše kao nova vrednost na pinu broj 13 koji je povezan sa diodom. Pročitana vrednost sa pina broj 2 je potrebno invertovati jer se koristi *pull-up* otpornik, tako da kada je taster pritisnut, na pinu broj 2 se pojavljuje nizak naponski nivo odnosno softverski se očitava „0“. Kod koji realizuje ovu ideju je sledeći:

```
const int DiodePinNo = 13;
const int ButtonPinNo = 2;
int DiodeStatus = LOW;
int ButtonStatus;
void setup() {
    //Set diode pin directions as output
    pinMode(DiodePinNo, OUTPUT);
    //Set button pin as input
    pinMode(ButtonPinNo, INPUT);
    //Initialize diode pin as LOW
    digitalWrite(DiodePinNo, DiodeStatus);
void loop() {
    //Read value from button pin
    ButtonStatus = digitalRead(ButtonPinNo);
    //Write value from button pin as diode pin value
    digitalWrite(DiodePinNo, !ButtonStatus);
}
```

Očitavanje vrednosti sa pina koji je inicijalizovan kao ulazni vrši se pozivom funkcije

Value = digitalRead(PinNo);

PinNo – Broj pina sa koga očitavamo vrednost

Pre funkcije *setup* vrši se definicija promenljivih koje će se koristiti u kodu. Promenljive *DiodePinNo* i *DiodeStatus* su poznate iz prethodnog poglavlja. Promenljiva *ButtonPinNo* definiše pin na koji je povezan taster, dok promenljiva *ButtonStatus* nosi informaciju o tome da li je taster pritisnut.

Unutar funkcije *setup* vrši se inicijalizacija sistema. Prvo se podese direkcije pinova. Pin na koji je povezana dioda podešava se kao izlazni, dok se pin na koji je povezan taster podešavamo kao ulazni. Nakon toga vrši se inicijalizacija vrednosti na pinu na koji je povezana dioda. Inicijalna vrednost na ovom pinu je „0“. Unutar glavne programske petlje *loop* vrši se stalno očitavanje vrednosti sa pina broj 2 na koji je prikačen taster i upisivanje inverzije pročitane vrednosti na digitani pin broj 13 za koji je zakačena dioda.



Ukoliko nakon spuštanja koda na Arduino hardversku platformu sistem i dalje ne funkcioniše na očekivani način, proveriti da li je taster dobro orijentisan. Ovo se može uraditi tako što se još jednom detaljno razmotri slika 5 - 6 i proveriti da li njegova funkcionalnost odgovara onoj sa slike.

Primer 5.2. – Uključenje diode pritiskom na taster

- *Realizovati sistem sa jednom diodom i jednim tasterom. Pritisak na taster treba da uzrokuje uključenje/isključenje diode. Napisati programski kod za Arduino i praktično potvrditi uspešnost realizovanog sistema.*

Rešenje:

Potrebni elementi:

- Arduino UNO
- Dva otpornika od 220 Ω
- Dioda
- Protobord
- 7 MM kratkospajča

Električna šema povezivanja kao i šema u Arduino okruženju su prikazani na slici 5 - 8.

U ovom primeru neophodno je detektovati pritisak tastera da bi se uključila dioda. Sa softverskog stanovišta, iza detekcije pritiska tastera stoji detekcija promene vrednosti na pinu na koji je povezan taster. Pod promenom se podrazumeva prelazak sa „1“ na „0“. Kada se detektuje pritisak tastera neophodno je promeniti stanje pina na koji je povezana dioda. Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
const int DiodePinNo = 13;
const int ButtonPinNo = 2;
const int BouncingTime = 200;
int DiodeStatus = LOW;
int ButtonPrevStatus;
int ButtonCurrentStatus;
void setup() {
    // Initialize diode pin direction as output
    pinMode(DiodePinNo, OUTPUT);
    // Initialize button pin direction as input
    pinMode(ButtonPinNo, INPUT);
    // Diode pin is initialized as HIGH
    digitalWrite(DiodePinNo, DiodeStatus);
    // Read current status on the button pin
```

```

    ButtonPrevStatus = digitalRead(ButtonPinNo);
    ButtonCurrentStatus = digitalRead(ButtonPinNo);
}
void loop() {
    // Read button pin status
    ButtonCurrentStatus = digitalRead(ButtonPinNo);
    // If the button current status is different from the
    // previous one
    if(ButtonCurrentStatus != ButtonPrevStatus){
        // Check if falling edge
        if(ButtonCurrentStatus == LOW){
            //Change diode status
            DiodeStatus = !DiodeStatus;
            //Write new diode status to the pin
            digitalWrite(DiodePinNo, DiodeStatus);
        }
        ButtonPrevStatus = ButtonCurrentStatus;
    }
}

```

Pre inicijalizacije sistema unutar funkcije *setup*, vrši se definisanje promenljivih koje će se koristiti u programu. Promenljive *DiodePinNo* i *ButtonPinNo* čuvaju informaciju o tome na koje digitalne pinove Arduino hardverske platforme su povezani dioda i taster. Promenljiva *DiodeStatus* čuva informaciju o trenutnom stanju pina na koji je povezana dioda.

Unutar funkcije *setup* vrši se već dobro pozanata inicijalizacija sistema tako što se pin na koji je povezana dioda podešava kao izlazni i dodeljuje mu se inicijalna vrednost, dok se pinovi koji se koriste kao pinovi tastera proglašavaju kao ulazni.

Unutar glavne programske petlje *loop* vrši se detekcija silazne ivice i kao rezultat te detekcije se uključuje dioda. U ovom primeru su uvedene dve nove promenljive koje se koriste unutar algoritma za detekciju silazne ivice. Te promenljive su: *ButtonCurrentStatus* i *ButtonPrevStatus*. Promenljiva *ButtonCurrentStatus* čuva informaciju o trenutnoj vrednosti na pinu na koji je povezan taster, dok promenljiva *ButtonPrevStatus* čuva informaciju o prethodnom stanju na tom pinu. Na početku, unutar funkcije *setup*, obe promenljive imaju iste vrednosti.

Prva naredba unutar funkcije *loop* vrši čitanje trenutne vrednosti stanja na pinu na koji je povezan taster. Zatim se vrši provera da li se pročitana vrednost razlikuje od vrednosti pročitane sa ovog pina u prethodnom izvršavanju *loop* funkcije. Ukoliko razlika postoji, tada smo detektovali promenu stanja na pinu za koji je povezan taster. Ova promena može se javiti iz dva razloga: usled puštanja tastera ili usled pritiskanja tastera. Pošto je zadatak da se detektuje stanje „pritisnut taster“, odnosno silazna naponska ivica na datom pinu, neophodno je odmah posle detekcije promene stanja na pinu detektovati i koja se promena desila. Ono po čemu se promena „pritisnut taster“ razlikuje od promene „pušten taster“ je to što nakon pritisnutog tastera stanje na pinu ostaje *LOW* za neko kratko vreme.

Dakle, nakon što se prvim *if* uslovom detektuje promena neophodno je drugim *if* uslovom utvrditi da li je stanje na tom pinu LOW što znači da je detektovana silazna ivica, odnosno pritisak tastera.

Nakon drugog *if* uslova i detekcije pritiska tastera neophodno je izvršiti promenu stanja pina na koji je povezana dioda pozivom funkcije *digitalWrite*.

Na kraju prvog *if* uslova, odnosno detekcije promene stanja na pinu, neophodno je promenljivoj *ButtonPrevStatus* dodeliti vrednost *ButtonCurrentStatus* jer na taj način sistem će u narednom izvršavanju *loop* funkcije imati informaciju o stanju ovog pina u toku trenutnog izvršavanja ove funkcije.



Nakon što pokrenemo gore navedeni kod i sistem počne sa radom, uočićemo da se on ne ponaša uvek na predviđen način, tj. dioda neće uvek menjati stanje na pritisak tastera onako kako bi to bilo očekivano. Razlog tome jeste takozvano odskakivanje tastera, u svetu elektronike poznat pod nazivom *bauncing*.

Tasteri su mehanički delovi električnog sistema. Kao takvi su veoma nepouzdati za korišćenje ali su istovremeneo i veoma neophodni. Kao mehanički delovi sadrže opruge koje su zadužene za vraćanje tastera u prvobitni položaj. Ove opruge stvaraju male oscilacije koje dovode do odskakivanja tastera. Odskakivanje tastera za posledicu ima oscilacije u signalu, što može da izazove veliki broj lažnih detekcija pritiska na taster.

Najjednostavniji način da se ovaj problem reši, tj. da se izvrši *debaunsiranje* tastera, je da se nakon detekcije promene stanja na pinu sačeka neko vreme, a zatim, proveriti da li je promena bila „pritisnut taster“ odnosno silazna ivica. Kod koji realizuje opisanu funkcionalnost je sledeći:

```
const int DiodePinNo = 13;
const int ButtonPinNo = 2;
const int BouncingTime = 200;
int DiodeStatus = LOW;
int ButtonPrevStatus;
int ButtonCurrentStatus;
void setup() {
    // Initialize diode pin direction as output
    pinMode(DiodePinNo, OUTPUT);
    // Initialize button pin direction as input
    pinMode(ButtonPinNo, INPUT);
    // Diode pin is initialized as HIGH
    digitalWrite(DiodePinNo, DiodeStatus);
    // Read current status on the button pin
    ButtonPrevStatus = digitalRead(ButtonPinNo);
    ButtonCurrentStatus = digitalRead(ButtonPinNo);
}
```

```
void loop() {
  // Read button pin status
  ButtonCurrentStatus = digitalRead(ButtonPinNo);
  // If the button current status is different from the
  // previous one
  if(ButtonCurrentStatus != ButtonPrevStatus){
    delay(BouncingTime);
    ButtonCurrentStatus = digitalRead(ButtonPinNo);
    // Check if falling edge
    if(ButtonCurrentStatus == LOW){
      //Change diode status
      DiodeStatus = !DiodeStatus;
      //Write new diode status to the pin
      digitalWrite(DiodePinNo,DiodeStatus);
    }
    ButtonPrevStatus = ButtonCurrentStatus;
  }
}
```

Za razliku od prethodnog rešenja u kome nije korišćeno debaunsiranje tastera, sada se nakon detekcije promene stanja na pinu ubacuje *delay* a zatim se vrši ponovno čitanje stanja na pinu na koji je povezan taster. Ovako se smanjuje verovatnoća da se pogrešno detektuje pritisak tastera koji je moguć usled odskakivanja tastera.

Primer 5.3. – Dve diode i taster

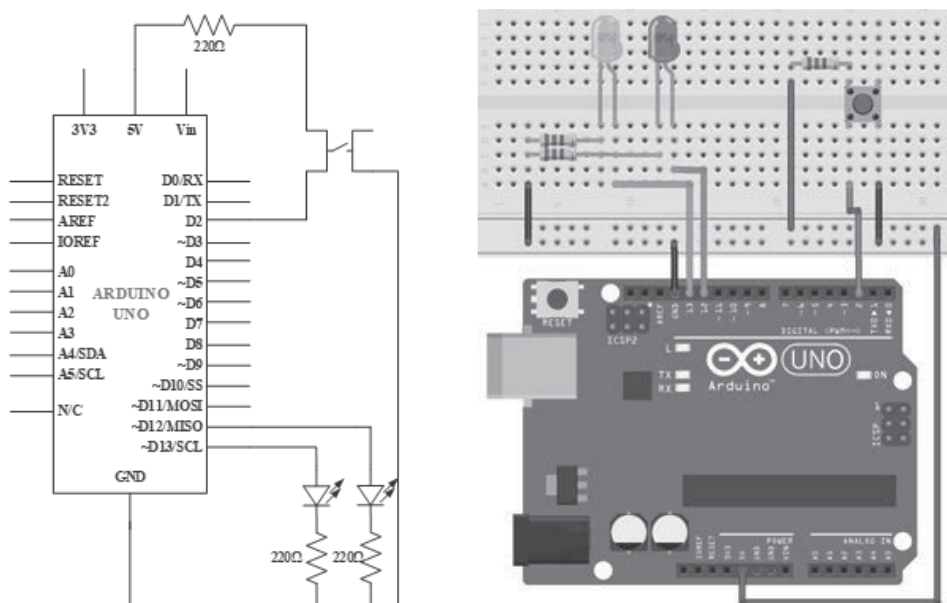
- *Realizovati Arduino sistem koji se sastoji od 2 diode (crvene i zelene) i tastera. Pritiskom na taster vrši se promena aktivne diode. Početno stanje je takvo da je crvena dioda uključena a zelena isključena. Napisati kod za Arduino platformu koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju sistema potrebni su sledeći elementi:

- Arduino UNO
- 3 otpornika od 220Ω
- 2 diode (crvena i zelena)
- Protobord
- 8 MM kratkospajča
- Taster

Električna šema koja predstavlja realizaciju sistema kao i fizička realizacija sistema u Arduino okruženju prikazani su na slici 5 - 9.



Slika 5 - 9 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 5.3.

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
const int GreenDiodePinNo = 13;
const int RedDiodePinNo = 12;
const int ButtonPinNo = 2;
const int BouncingTime = 200;
int DiodeStatus = LOW;
int ButtonPrevStatus;
int ButtonCurrentStatus;
int GreenDiodeStatus = LOW;
int RedDiodeStatus = HIGH;

void setup() {
    // Initialize diodes pins direction as output
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT);
    // Initialize button pin direction as input
    pinMode(ButtonPinNo, INPUT);
    // Write diodes pins init values
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    // Read current status on the button pin
    ButtonPrevStatus = digitalRead(ButtonPinNo);
    ButtonCurrentStatus = digitalRead(ButtonPinNo);
}

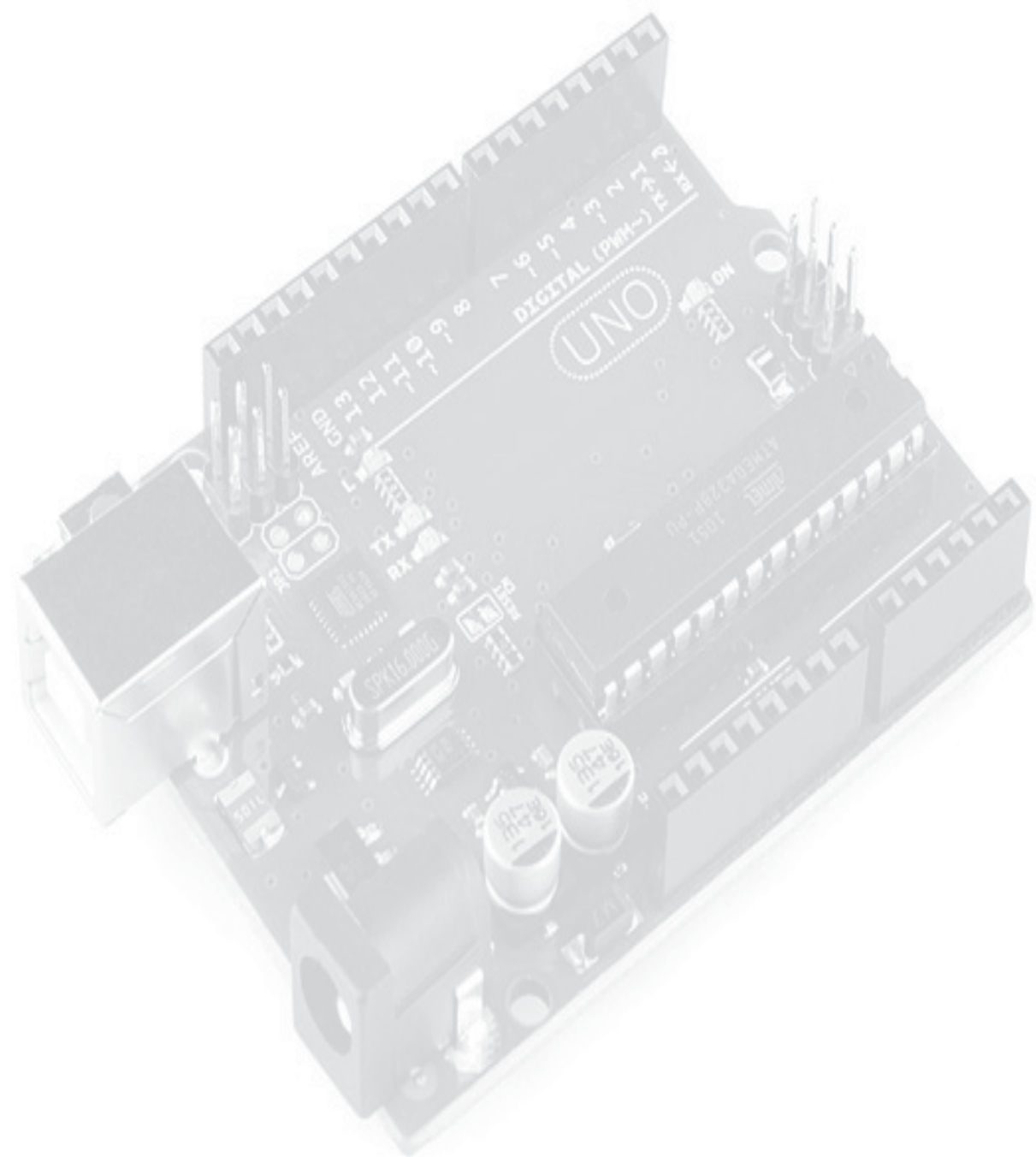
void loop() {
```

```
// Read button pin status
ButtonCurrentStatus = digitalRead(ButtonPinNo);
// If the button current status is different from the
// previous one
if(ButtonCurrentStatus != ButtonPrevStatus){
    delay(BouncingTime);
    ButtonCurrentStatus = digitalRead(ButtonPinNo);
    // Check if falling edge
    if(ButtonCurrentStatus == LOW){
        //Change diodes status
        RedDiodeStatus = !RedDiodeStatus;
        GreenDiodeStatus = !GreenDiodeStatus;
        //Write new diodes status to the pins
        digitalWrite(RedDiodePinNo, RedDiodeStatus);
        digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    }
    ButtonPrevStatus = ButtonCurrentStatus;
}
}
```

Ideja koju realizuje gore prikazani kod je slična kao u prethodnom zadatku, samo što sada umesto jedne, radimo sa sve diode.

5.3. ZADACI ZA SAMOSTALAN RAD

1. Realizovati Arduino sistem koji se sastoji od dve diode i jednog tastera. Kada taster nije pritisnut treba da sija samo jedna dioda. Kada je taster pritisnut, i drži se, diode treba naizmenično da se uključuju i isključuju na svakih 0.5 sekundi. U trenutku kada se otpusti taster ostaje uključena ona dioda koja je svetlela u trenutku otpuštanja tastera. Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.
2. Realizovati Arduino sistem koji se sastoji od 2 diode i jednog tastera. Dioda trepere sa periodom od 3 sekunde. U jednom trenutku treperi samo jedna dioda i ona se naziva aktivnom. Promena aktivne diode se vrši pritiskom na taster. Napisati kod za Arduino platformu koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.
3. Realizovati sistem koji se sastoji od dva tastera i jedne diode. Pritiskom na jedan taster se vrši povećanje periode treperenja dok se pritiskom na drugi taster perioda treperenja diode smanjuje. Perioda treperenja diode uzima vrednosti iz konačnog skupa čiji su elementi 0.5s, 1s, 1.5s, 2s i 2.5s. Na početku je perioda treperenja diode 1.5s. Napisati kod za Arduino hardversku platformu koji realizuje traženu funkcionalnost i potvrditi uspešnost realizacije hardverskom implementacijom sistema.



6. NAPREDNE OPERACIJE SA DIGITALNIM ULAZIMA I IZLAZIMA: MEHANIZAM PREKIDA I IMPULSNO ŠIRINSKA MODULACIJA

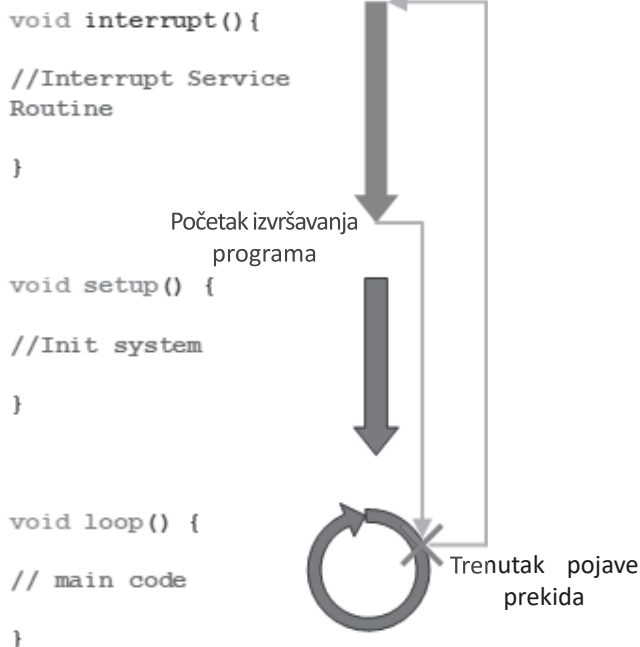
Ovo poglavlje ima za cilj da čitaoca upozna sa naprednim načinima korišćenja digitalnih ulaza i izlaza koji se dosta često koriste u praksi. Koncepti koji se uvode u ovom poglavlju su mehanizam prekida i impulsno širinska modulacija (PWM).

6.1. MEHANIZAM PREKIDA

U svim primerima sa digitalnim ulazima iz poglavlja 5 proveru stanja tastera vrši se neprekidnim čitanjem digitalne vrednosti pina na koji je taster povezan. Ovakvo neprekidno očitavanje naziva se *polling* i nije previše pouzdano. Naime, u primerima iz poglavlja 5. u okviru *loop* petlje, pored komandi za očitavanje stanja odgovarajućih digitalnih pinova nema velikog broja drugih programskih komandi te se svaki pritisak na taster detektuje. Međutim, ukoliko bi kod u *loop* petlji sadržao jako veliki broj programskih komandi za čije izvršavanje je potrebno nezanemarljivo vreme, postojao bi rizik da dođe do pritiskanja i otpuštanja tastera dok se ove komande izvršavaju. U tom slučaju promena stanja digitalnog pina ne bi na vreme bila detektovana i pritisak na taster ne bi imao nikakav efekat.

Ovaj problem, kao i potreba za polingom mogu se eliminisati korišćenjem mehanizma prekida (eng. *interrupt*). Prekid se definiše kao signal procesoru mikrokontrolera da se dogodio neki događaj kome treba da se posveti pažnja. Kada procesor primi ovaj signal, on prekida izvršavanje programskog koda koji je do tada izvršavao i izvršava posebnu funkciju koja vrši servisiranje prekida. Ova funkcija se naziva servisna rutina prekida (eng. *interrupt service routine*). Kada procesor završi sa izvršavanjem servisne rutine prekida, vraća se na izvršavanje originalnog programskog koda od tačke u kojoj je bio prekinut pojavom prekida. Ilustracija toka Arduino programa sa prekidom data je na slici 6 - 1.

Funkcija za servisiranje prekida se automatski poziva po prijemu signala prekida i nema smisla eksplicitno pozivati ovu funkciju bilo gde u kodu. Stoga ona nikada ne vraća rezultat (uvek se deklarise kao *void*).



Slika 6 - 1- Mehanizam poziva prekidne rutine

Signal prekida može biti generisan interno od strane mikrokontrolera ili eksterno od strane komponenti i uređaja koji su povezani sa mikrokontrolerom. Spoljašnji prekidi se mikrokontroleru signaliziraju putem digitalnih pinova koji su opremljeni posebnim hardverskim mehanizmom za podršku radu sa prekidima. Da bi ovakav digitalni pin generisao prekid, potrebno je da bude deklarisan kao ulazni. Naponska stanja na pinu koja mogu dovesti do prekida su:

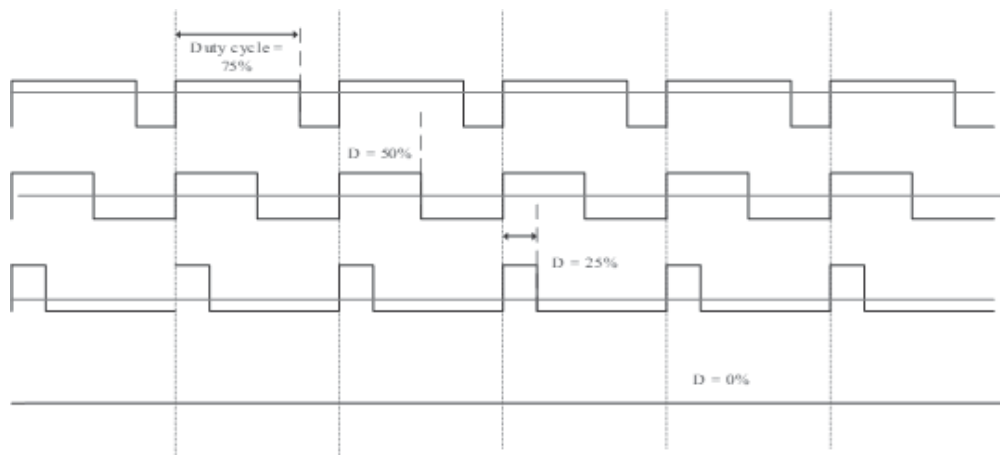
- vrednost naponskog nivoa na pinu je na vrednosti logičke jedinice
- vrednost naponskog nivoa na pinu je na vrednosti logičke nule
- vrednost naponskog nivoa na pinu se menja sa logičke nule na logičku jedinicu
- vrednost naponskog nivoa na pinu se menja sa logičke jedinice na logičku nulu
- vrednost naponskog nivoa na pinu se promenio

Na Arduino UNO platformi digitalni pinovi koji omogućavaju rad sa prekidima su pinovi 2 i 3.

6.2. IMPULSNO ŠIRINSKA MODULACIJA

Impulsno širinska modulacija je tehnika modulisanja digitalnog signala koja za cilj ima utiskivanje određene informacije u taj signal. Kod širinsko impulsne modulacije informacije se enkoduju uz pomoć širine impulsa [11]. Impulsno širinski modulisani signal je digitalni signal sa dva nivoa (nivoi logičke „1“ i „0“). Ovaj signal je periodičan i karakteriše ga perioda, kao i procenat trajanja pozitivnog nivoa u toku jedne periode, koji se naziva *Duty Cycle* i najčešće se izražava u procentima. Promenom *Duty Cycle*-a moguće je kontrolisati srednju vrednost signala što je veoma značajno u nekim primenama kao što su intezitet osvetljaja diode, brzina okretanja motora, itd.

Na slici 6 - 2 prikazana je zavisnost srednje vrednosti signala od promene *Duty Cycle*-a. Horizontalne pune linije na slici označavaju srednju vrednost, koja direktno zavisi od *Duty Cycle*-a signala.



Slika 6 - 2 - PWM signal

6.3. PRIMERI

Primeri koji slede imaju za cilj da ilustruju primenu naprednog korišćenja digitalnih ulaza i izlaza. Neki primeri iz prethodnih poglavlja ovde su elegantnije rešeni korišćenjem mehanizma prekida.

Primer 6.1. - Dve diode i taster (uz pomoć prekida)

- Rešiti primer 5.3 primenom mehanizma prekida.

Rešenje:

Potrebne elektronske komponente i načini vezivanja su isti kao u primeru 5.3. Ono po čemu se ova dva rešenja istog zadatka razlikuju jeste što je u primeru 5.3 korišćen mehanizam polinga, dok se ovde koristi mehanizam prekida. Kod za ovu realizaciju je sledeći:

```
int GreenDiodePinNo = 13;
int RedDiodePinNo   = 12;
int ButtonPinNo     = 2;
int GreenDiodeStatus = LOW;
int RedDiodeStatus   = HIGH;
void setup() {
    //Initialize diode pins direction as output
    pinMode(GreenDiodePinNo, OUTPUT);
    pinMode(RedDiodePinNo, OUTPUT);
    //Button pin direction as input pin
    pinMode(ButtonPinNo, INPUT);
    //Initialize diode's pins value
    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
    //Attach interrupt to the pin
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
}
void ButtonISR()
{
    GreenDiodeStatus = !GreenDiodeStatus;
    RedDiodeStatus   = !RedDiodeStatus;
    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
}
void loop() {
    //Processor is free
}
```

Dodeljivanje prekidne rutine nekom od pinova koji podržavaju prekidne zahteve vrši se pozivom funkcije:

`attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);`

pin - Broj pina kome dodeljujemo prekid

ISR - Naziv funkcije koju će program pozvati nakon pojave prekida

mode - Koja pojava na pinu uzrokuje prekid. Postoje sledeći modovi:

- HIGH – pin ima vrednost logičke jedinice
- LOW - pin ima vrednost logičke nule
- RISING – vrednost pina se promenila sa „0“ na „1“
- FALLING – vrednost pina se promenila sa „1“ na „0“
- CHANGE - vrednost pina se promenila (bilo kako)

Prikazano rešenje se od prethodnog razlikuje po tome što se kod koji se prethodno nalazio u glavnom programu i koji se izvršavao kada za to nije bilo potrebe, sada nalazi u prekidnoj rutini koja je dodeljena pinu broj 2. Zapravo, pinu broj 2 je dodeljena prekidna rutina pod nazivom *ButtonISR* u kojoj se nalazi procedura za obradu prekida. U ovu prekidnu rutinu se ulazi kada se pojavi silazna ivica signala na pinu broj 2, tj. usled promene vrednosti na pinu broj 2 sa „1“ na „0“. Obzirom da je taster u kolo vezan uz pomoć pull up otpornika, silazna naponska ivica na pinu 2 će se javiti prilikom pritiska tastera.

Takođe, iz priloženog koda se vidi da je zbog korišćenja prekida, sada glavna programska petlja *loop* ostala prazna, jer je kompletna funkcionalnost prebačena u prekidnu rutinu.



Prilikom pisanja funkcija za servisiranje prekida, treba voditi računa o tome da prekidne procedure budu što kraće i što efikasnije, tj. da se što manje posla obavlja u prekidnoj proceduri.

Programski kod koji sledi ima zadatak da demonstrira najefikasnije rešenje primera 5.3. U ovom programskom kodu smo izbegli pisanje dugačkih prekidnih procedura i podelili smo funkcionalnost između prekidne rutine i glavne programske petlje.

```

int GreenDiodePinNo = 13;
int RedDiodePinNo   = 12;
int ButtonPinNo     = 2;
int GreenDiodeStatus = LOW;
int RedDiodeStatus   = HIGH
volatile boolean ButtonPressed;
void setup() {
    //Initialize diodes pins direction as output
    pinMode(GreenDiodePinNo, OUTPUT);
    pinMode(RedDiodePinNo, OUTPUT);
    //Initialize button pin direction as input pin
    pinMode(ButtonPinNo, INPUT);
    //Initialize diode's pins value
    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
    //Attach interrupt to the pin
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
    //Button isn't pressed
    ButtonPressed = false;
}
void ButtonISR(){
    // Remember that button is pressed
    ButtonPressed = true;
}
void loop() {
    //check if button is pressed
    if(ButtonPressed == true )
    {
        // if button is pressed change diodes status
        GreenDiodeStatus = !GreenDiodeStatus;
        RedDiodeStatus = !RedDiodeStatus;
        digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
        digitalWrite(RedDiodePinNo, RedDiodeStatus);
        ButtonPressed = false;
    }
}

```

Iz priloženog rešenja se vidi da je uvedena pomoćna promenljiva tipa boolean koja predstavlja vezu između glavnog programa i prekidne rutine, a naziva se *ButtonPressed*. Ova promenljiva nosi informaciju o tome da li je pritisnut taster. Postavlja se na vrednost *true* u prekidnoj rutini ukoliko se detektuje pritisak tastera. U glavnoj programskoj petlji *loop* se proverava da li je postavljena ova vrednost i ukoliko jeste menjaju se stanja dioda.



Kada se definiše globalna promenljiva i kada se vrednost te promenljive menja iz više funkcija neophodno je da ta promenljiva bude definisana kao *volatile* promenljiva, inače se može narušiti funkcionalnost sistema.

Primer 6.2. – Promena intenziteta osvetljenosti diode

- *Realizovati Arduino sistem koji se sastoji od dva tastera i diode gde se tasterima podešava intenzitet diode. Inicijalno dioda treba da svetli sa 50% intenziteta. Pritisak na jedan taster smanjuje intenzitet za 10% dok pritisak na drugi taster povećava intenzitet osvetljaja diode za 10%. Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.*

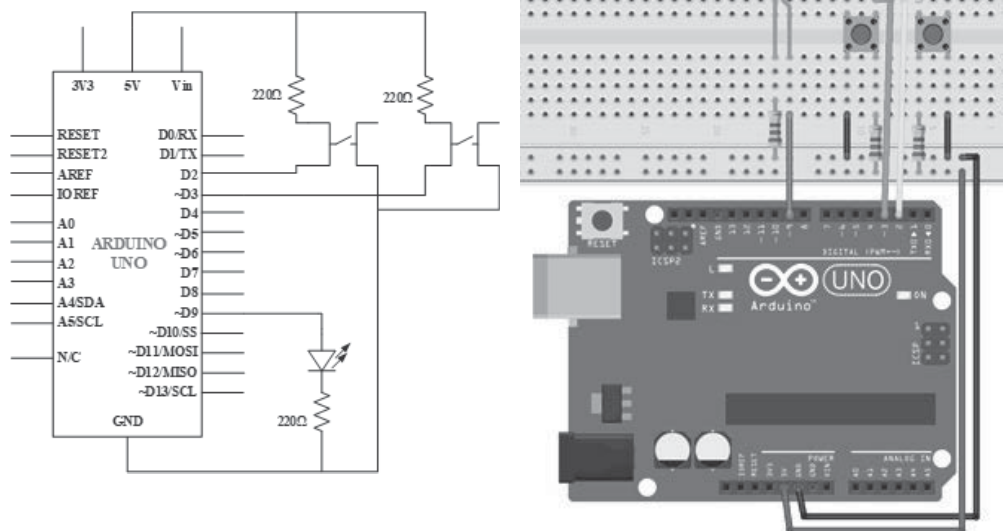
Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino Uno
- 3 otpornika od 220Ω
- Dioda
- 7 MM Kratkospajča
- Protobord
- 2 tastera

Električna šema koja ilustruje način povezivanja komponenti kao i izgled fizičke realizacije sistem prikazani su na slici 6 - 3. Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
const int DiodePinNo      = 9;
const int ButtonUpPinNo   = 2;
const int ButtonDownPinNo = 3;
volatile int Intensity     = 50; //%
void DownISR(){
    //Decrease intensity value
    if(Intensity>0){
        Intensity -= 10;
    }
}
void UpISR(){
    //Increase intensity value
    if(Intensity<100){
        Intensity +=10;
    }
}
```



Slika 6 - 3 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 6.2.

```
void setup() {
    //Initialize system
    pinMode(DiodePinNo, OUTPUT);
    pinMode(ButtonUpPinNo, INPUT);
    pinMode(ButtonDownPinNo, INPUT);
    attachInterrupt(digitalPinToInterrupt(ButtonUpPinNo), UpISR, FALLING);
    attachInterrupt(digitalPinToInterrupt(ButtonDownPinNo), DownISR, FALLING);
}
void loop() {
    analogWrite(DiodePinNo, 255*Intensity/100);
}
```

Na Arduino platformi postoje pinovi koji omogućavaju generisanje PWM signala. Ti pinovi su digitalni pinovi označeni sa ~ ispred naziva pina. Učestanost signala na ovim pinovima je veća od 490Hz. Promena Duty Cycle-a se vrši pozivom funkcije

`analogWrite(PinNo, Value);`

PinNo – Broj PWM pina

Value – Vrednost od 0 do 255 gde vrednost 255 označava Duty Cycle 100% , 127 označava Duty Cycle 50% a 0 označava Duty cycle 0%

Dioda je povezana na pin broj 9 Arduino hardverske platforme, koji ima mogućnost generisanja PWM signala. Tasteri su povezani na pinove 2 i 3 koji imaju mogućnost generisanja prekida. Kada se pritisne taster koji je povezan na pin broj 2 usled silazne ivice signala odlazi se u prekidnu rutinu *UpISR* u kojoj povećavamo vrednost promenljive *Intensity* za 10. Suprotno ovoj funkciji za servisiranje prekida imamo funkciju *DownISR* u koju se ulazi usled silazne ivice signala na pinu broj 3 na koji je priključen drugi taster.

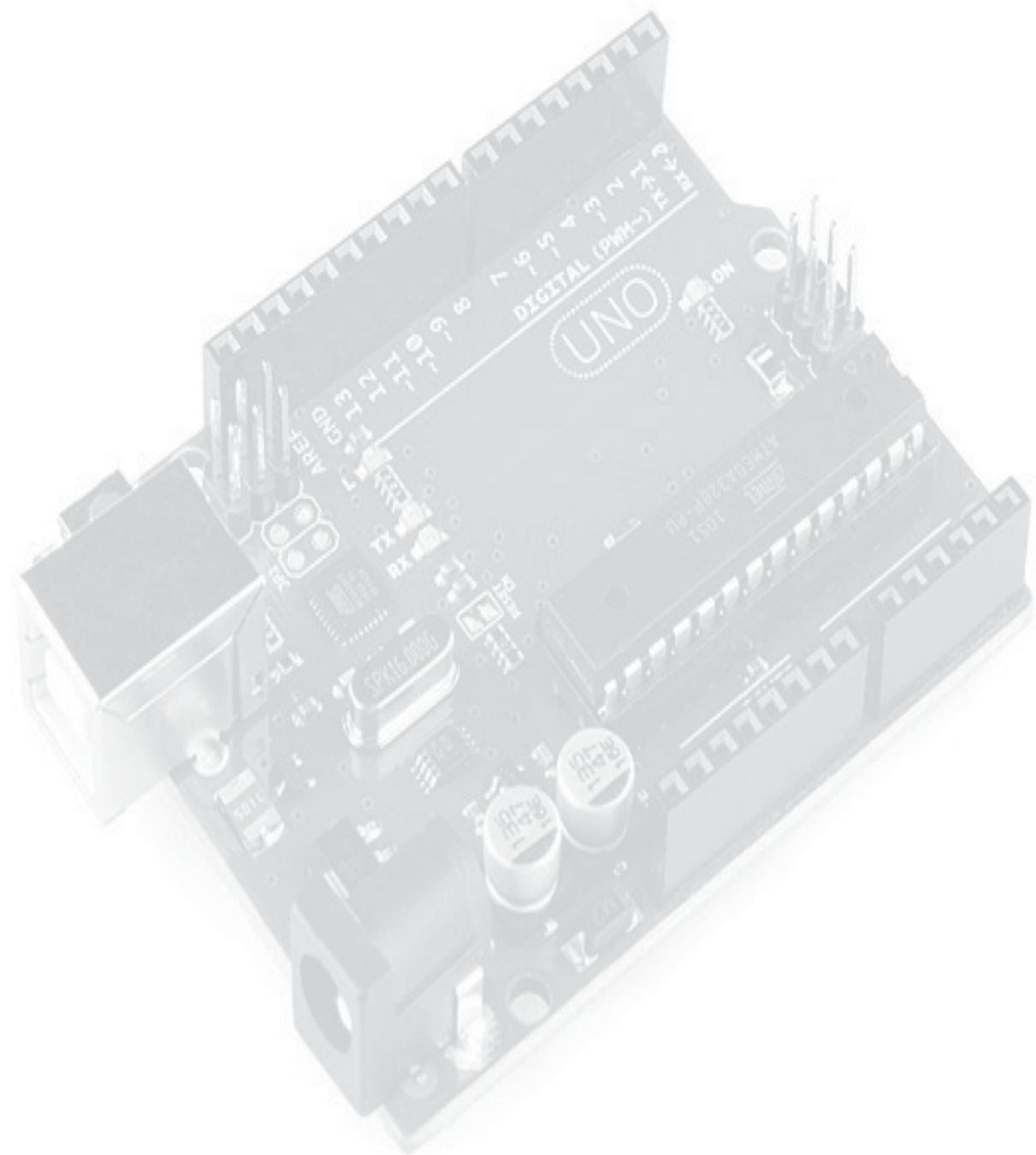
U glavnoj programskoj petlji pozivom funkcije *analogWrite* vršimo stalno generisanje signala na pinu broj 9. *DutyCycle* ovog signala izračunat u procentima jednak je vrednosti smeštenoj u promenljivi *Intensity*.

Pitanje: Koja je veza između PWM signala i intenziteta osvetljenosti diode?

Odgovor: Intenzitet osvetljenosti diode u direktnoj je vezi sa karakteristikama ljudskog vida. Naime, teorijski je utvrđeno da ljudsko oko ne može da primeti promenu u osvetljenosti diode ukoliko ona treperi sa periodom manjom od 20ms. Dakle, perioda PWM signala mora biti manja od 20ms, odnosno njegova frekvencija mora biti veća od 50Hz da se ne bi videlo treperenje diode. Intenzitet osvetljenosti LE diode direktno je srazmeran srednjoj vrednosti struje koja prolazi kroz diodu. Srednja vrednost PWM signala opada sa smanjenjem *Duty Cycle*-a diode. Dakle, ako želimo da smanjimo osvetljenje diode, neophodno je smanjiti *Duty Cycle* PWM signala.

6.4. ZADACI ZA SAMOSTALAN RAD

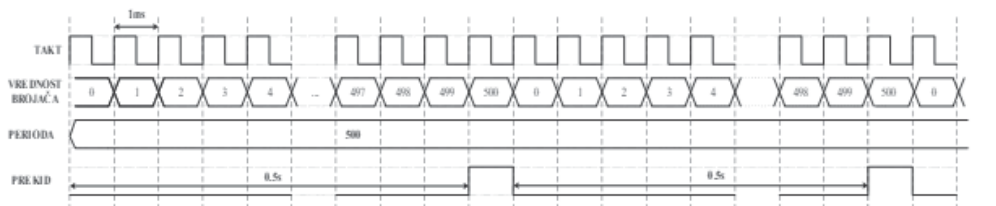
1. Realizovati Arduino sistem koji se sastoji od 2 tastera i 3 diode (crvena, zelena i žuta). Pritiskom na tastere sekvencijalno se menja aktivna dioda. Inicijalno je uključena crvena dioda, dok su žuta i zelena isključene. Pritiskom na jedan taster treba menjati aktivnu diodu po sekvenci crvena-žuta-zelena dok pritiskom na drugi taster treba menjati aktivnu diodu u suprotnom smeru, tj. zelena-crvena-žuta. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije
2. Realizovati Arduino sistem koga čine dve diode (zelena i crvena) i tri tastera. Pritiskom na prvi taster se vrši selekcija aktivne diode, dok drugi i treći taster služe za povećanje, odnosno smanjenje, intenziteta svetlosti diode. Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.



7. TAJMERI

Tajmeri spadaju u periferni uređaji mikrokontrolera čija je osnovna uloga da mere protok vremena. Tajmeri se sastoje iz brojača i registra. Tajmer koristi *clock* signal koji predstavlja povorku naponskih četvrtki konstantne periode. Ovaj signal generiše oscilator. Brojač tajmera se uvećava za jedan (inkrementira) na svaku uzlaznu ivicu *clock* signala. Vrednost brojača indirektno ukazuje na proteklo vreme, pošto ono odgovara proizvodu trenutne vrednosti brojača i poznate periode clock signala. Na primer, ukoliko tajmer koristi *clock* signal sa periodom od 1ms, vrednost brojača od 1000 ukazuje da je proteklo vreme od 1 sekunde.

Registar sadrži maksimalno dozvoljenu vrednost brojača. Posle svakog inkrementiranja brojača, vrši se njegovo poređenje sa registrom. Kada vrednost brojača dostigne vrednost upisanu u registar brojač se resetuje, a procesoru se šalje interni prekidni signal. Ispravnim postavljanjem vrednosti tajmerskog registra postiže se periodična pojava prekida sa tačno željenom periodom. Tako na primer, ukoliko clock signal ima periodu od 1ms, postavljanjem vrednosti tajmerskog registra na 500 ostvaruje se periodična pojava prekida na svakih 0.5s, dok se postavljanjem ove vrednosti na 1000 postiže periodična pojava prekida na 1s. Opisani mehanizam rada tajmera ilustrovan je na slici 7 - 1.



Slika 7 - 1 - Generisanje signala prekida od 0.5 sekundi prikazan na dužem vremenskom intervalu

Tajmeri su važan deo svakog ugrađenog sistema [12]. Oni se najčešće koriste za izvršavanje periodičnih programskih zadataka, kao što su čitanje vrednosti sa senzora, osvežavanje displeja, čitanje sa tastature i slično. Upotrebom tajmera izbegava se potreba za korišćenjem komande *delay()* koja blokira procesor za zadati vremenski period. Tajmeri za razliku od ove komande obezbeđuju periodičnost sa velikom tačnošću i dozvoljavaju konstantnu upotrebu procesora za izvršavanje korisnih programskih funkcija.

Arduino UNO hardverska platforma ima takt od 16MHz i na njoj postoje 3 tajmera: *Timer0*, *Timer1* i *Timer2*. *Timer0* je osmobitni tajmer i njega koriste funkcije kao što su *delay*, *milis* i *micros* dok se *Timer2* koristi u nekim drugim Arduino bibliotekama.

Timer1 predstavlja šesnaestobitni tajmer i pristup ovom tajmeru je abstrahovan određenim funkcijama koje se nalaze u biblioteci *TimerOne*. Instalacija ove biblioteke je objašnjena u poglavlju 2.2. dok će u nastavku biti urađeni primeri koji ilustruju način upotrebe funkcija iz ove biblioteke. Opis pojedinih funkcija ove biblioteke može se naći na internetu [22].

7.1. PRIMERI

Primeri koji slede imaju za cilj da čitaocu demnostriraju rad sa tajmerima.

Primer 7.1. – Treperenje diode

- *Realizovati Arduino sistem koji se sastoji od jedne diode koja treperi sa peridom od 0.5s. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente

- Arduino UNO
- Otpornik 220Ω
- Dioda
- 3 MM kratkospajča
- Protobord

Električna šema koja ilustruje način povezivanja komponenti i fizička realizacija sistema prikazani su na slici 4 - 6.

Ovaj primer je idejno isti kao i primer 4.1. Razlika je u tome što se sada glavna funkcionalnost programa implementira u funkciji koja obrađuje prekid tajmera. Kod koji realizuje ovu funkcionalnost je sledeći:

```
#include <TimerOne.h>
const int DiodePinNo = 13;
const unsigned long TimerPeriod = 500000; //microseconds

int DiodeStatus = LOW;
```

```
void TimerISR(){
    //Change diode status
    DiodeStatus = !DiodeStatus;
    //Write new value to the diode pin
    digitalWrite(DiodePinNo, DiodeStatus);
}
void TimerISR(){
    //Change diode status
void setup() {
    //Initialize diode pin as output
    pinMode(DiodePinNo, OUTPUT);
    //Write value to the diode pin
    digitalWrite(DiodePinNo, DiodeStatus);
    //Initialize timer period
    Timer1.initialize(TimerPeriod);
    //Assign timer interrupt function
    Timer1.attachInterrupt(TimerISR);
}
void loop() {
    // put your main code here, to run repeatedly:
}
```

Prilikom inicijalizacije sistema neophodno je podesiti periodu tajmera što se radi pozivom funkcije *initialize* nad tajmerom 1. Sintaksa ove instrukcije je:

Timer1.initialize(TimerPeriod);

TimerPeriod – perioda tajmera u mikrosekundama (μ s)

Funkcija koja će obrađivati prekid tajmera dodeljuje se pozivom funkcije

Timer1.attachInterrupt(FunctionName);

FunctionName – naziv funkcije koja obrađuje prekid tajmera

Pre funkcije *setup* vrši se definisanje promenljivih koje će se koristiti u nastavku. Najpre se definiše promenljiva *DiodePinNo* koja označava pin na koji je povezana dioda. Nakon toga se promenljivom *TimerPeriod* definiše perioda tajmera.

Unutar funkcije *setup* vrši se konfiguracija pina diode na već poznat način. Nakon toga se inicijalizuje tajmer tako što mu se podesi perioda, a zatim se dodeli funkcija koja će se pozivati kada se desi prekid tajmera. U ovom primeru to je funkcija *TimerISR* unutar koje se vrši promena stanja diode.

Prednost ovakvog pristupa, za razliku od načina na koji je rešen primer 4.1, je što se više ne koristi funkcija *delay* koja odlaže izvršenje sledeće komande na određeno vreme u toku koga je procesor neupotrebljiv. Ovakvom realizacijom omogućava se da procesor bude iskorišćen sve vreme rada.

Primer 7.2. – Promena periode treperenja diode

- *Realizovati Arduino sistem koji se sastoji od jedne diode i dva tastera. Pritiskom na jedan taster se vrši uvećanje periode treperenja diode za 0.01 sekundu dok se pritiskom na drugi taster vrši smanjenje periode treperenja diode za 0.01 sekundu. Inicijalno je perioda treperenja diode jednaka 0.1 sekundi. Perioda treperenja diode treba da se nalazi u opsegu od 0.01 do 0.5 sekundi. Ukoliko se trenutna vrednost periode treperenja diode nalazi na granici opsega, a korisnik pritisne neki od tastera koji može dovesti do prekoračenja opsega, taj pritisak tastera treba ignorisati. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 3 Otpornika 220Ω
- Dioda
- 7 MM kratkospajča
- Protobord
- 2 Tastera

Električna šema koja ilustruje način povezivanja navedenih električnih komponenti, kao i fizička realizacija sistema prikazani su na slici 6 - 3.

```
#include <TimerOne.h>
const int DiodePinNo      = 13;
const int ButtonUpPinNo   = 2;
const int ButtonDownPinNo = 3;
volatile int TimerPeriod   = 100000; //microseconds
volatile int DiodeStatus   = LOW;

void TimerISR(){
    DiodeStatus = !DiodeStatus;
    digitalWrite(DiodePinNo,DiodeStatus);
}

void DownISR(){
    TimerPeriod-=10000;
    Timer1.setPeriod(TimerPeriod);
}

void UpISR(){
    TimerPeriod+=10000;
    Timer1.setPeriod(TimerPeriod);
}

void setup() {
```

```
//Init diode pin as output
pinMode(DiodePinNo, OUTPUT);
//Init button pins as input
pinMode(ButtonUpPinNo, INPUT);
pinMode(ButtonDownPinNo, INPUT);
//Attach interrupts to the buttons pins
attachInterrupt(digitalPinToInterrupt(ButtonUpPinNo), UpISR, FALLING);
attachInterrupt(digitalPinToInterrupt(ButtonDownPinNo), DownISR, FALLING);
//Set timer
Timer1.initialize(TimerPeriod);
Timer1.attachInterrupt(TimerISR);
}
void loop() {
  //free
}
```

Promena periode tajmera tokom izvršavanja programa se vrši pozivom funkcije
Timer1.setPeriod(NewPeriod);
 NewPeriod – Nova perioda tajmera izražena u mikrosekundama

Pritiskom na svaki od tastera se poziva funkcija koja obrađuje prekid generisan od strane tog tastera. Unutar te funkcije vrši se uvećanje ili smanjenje vrednosti promenljive *TimerPeriod* koja predstavlja periodu tajmera. Nakon promene ove vrednosti neophodno je podesiti periodu tajmera na novu vrednost, što se postiže pozivom funkcije *Timer1.setPeriod()*. Paljenje i gašenje diode odvija se u funkciji *TimerISR()* koja se poziva sa periodom koja je jednaka *TimerPeriod*.

Primer 7.3. - Dve diode koje trepere i taster

- *Realizovati Arduino sistem koji se sastoji od 2 diode (crvena i zelena) i jednog tastera. Dioda trepere sa periodom od 0.5 sekundi. U jednom trenutku treperi samo jedna dioda i naziva se aktivna dioda. Promena aktivne diode se vrši pritiskom na taster. Napisati kod za Arduino platformu koji realizuje ovaj sistem korišćenjem tajmera i praktično potvrditi uspešnost napisanog koda.*

Rešenje:

Komponente potrebne za realizaciju ovog sistema su sledeće:

- Arduino UNO
- 3 Otpornika od 220Ω
- 2 Dioda (crvena i zelena)

- Protobord
- 8 MM kratkospajči
- Taster

Električna šema i fizička realizacija sistema prikazani su na slici 5 - 9.

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
#include <TimerOne.h>
const int GreenDiodePinNo = 13;
const int RedDiodePinNo   = 12;
const int ButtonPinNo     = 2;
volatile int ButtonStatus;
volatile int GreenDiodeStatus = LOW;
volatile int RedDiodeStatus   = LOW;
//Combination == true => Red diode active
//Combination == false => Green diode active
volatile boolean Combination;
const unsigned long DiodePeriod = 500000;
//Diode period blink in microseconds

void setup() {
    //Initialize all port's pins direction and variable
    //Diode pin directions as output pin
    pinMode(GreenDiodePinNo, OUTPUT);
    pinMode(RedDiodePinNo, OUTPUT);
    //Button pin direction as input pin
    pinMode(ButtonPinNo, INPUT);
    //Initialize diode pin pins value
    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
    //Red active
    Combination = true;
    //Attach interrupt to the pin
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
    //Initialize timer
    Timer1.initialize(DiodePeriod);
    Timer1.attachInterrupt(TimerISR);
}
void ButtonISR()
{
    Combination = !Combination;
}
void TimerISR()
{
    if(Combination == false)
    {
        GreenDiodeStatus = !GreenDiodeStatus;
        digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
    }
}
```

```

else
{
    RedDiodeStatus = !RedDiodeStatus;
    digitalWrite(RedDiodePinNo, RedDiodeStatus);
}
}
void loop() {
    //Free
}

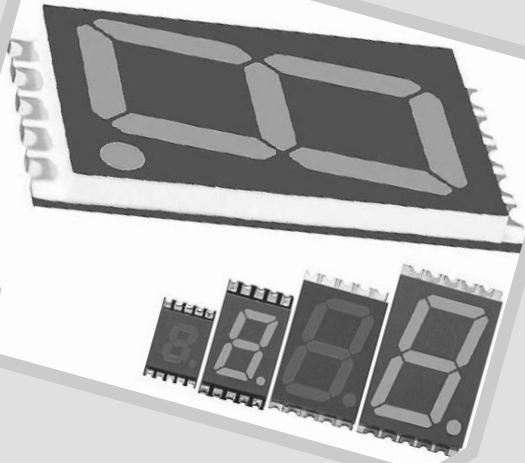
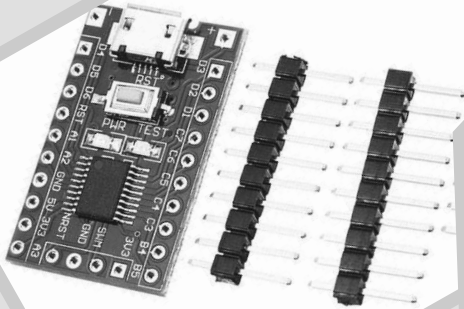
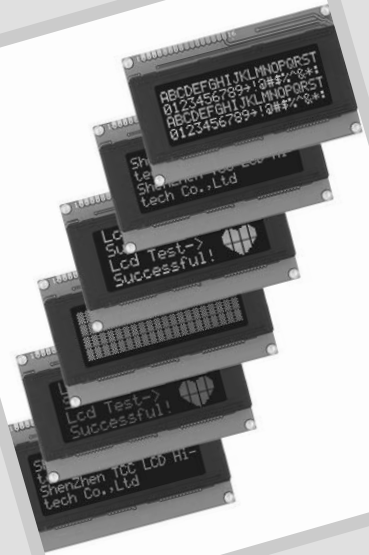
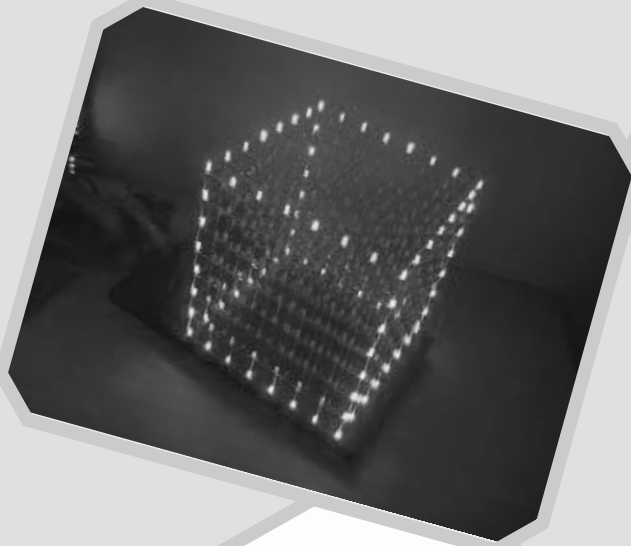
```

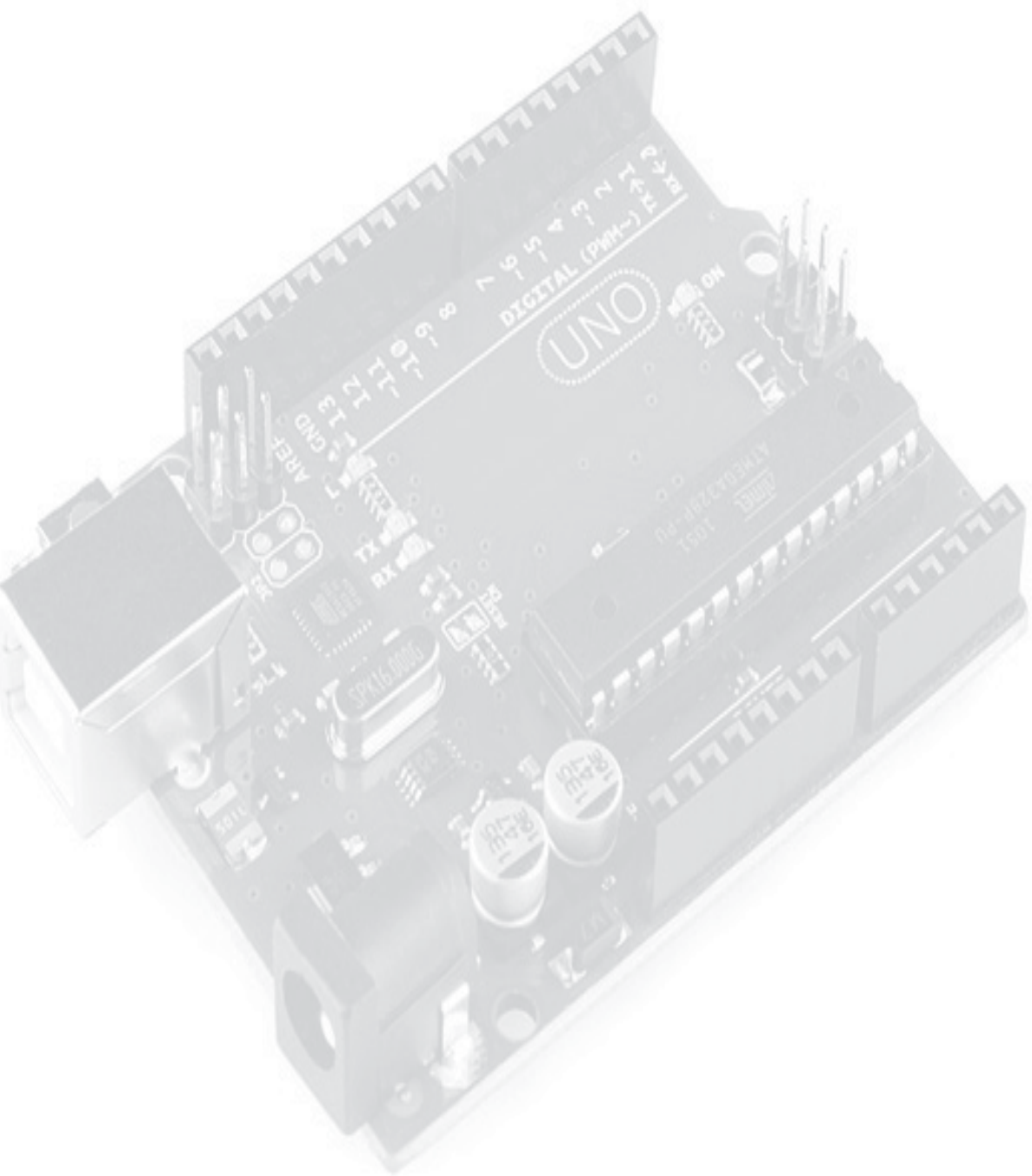
Da bi softversko korišćenje tajmera bilo moguće, najpre je na početku programa neophodno uključiti biblioteku *TimerOne.h*. Nakon toga sledi definicija promenljivih koje određuju pinove koji će se koristiti, a zatim se definišu početna stanja na pinovima. Uvedena je promenljiva *Combination* koja ima dve moguće vrednosti *true* i *false*. Njen zadatak je da pamti trenutno aktivnu diodu. Ukoliko je vrednost ove promenljive *true*, to znači da je aktivna crvena dioda, dok ukoliko je vrednost ove promenljive *false*, to znači da je trenutno aktivna zelena dioda. Ova promenljiva menja vrednost u prekidnoj rutini *ButtonISR*, koja se poziva usled pritiska tastera. Dakle, pritiskom na taster menjamo stanje ove diode.

Prekidni zahtev tajmera javlja se periodično sa definisanom periodom (za ovaj primer na 3 sekunde). Sa istekom periode tajmera poziva se prekidna procedura *TimerISR* koja ima zadatak da promeni stanje trenutno aktivne diode iz aktivnog u neaktivno i obrnuto.

7.2. ZADACI ZA SAMOSTALNI RAD

1. Realizovati Arduino sistem sa četiri diode i jednim tasterom. Dioda su grupisane u dve grupe od po dve diode. Pritiskom na taster se menja aktivna grupa. Unutar aktivne grupe se diode koje čine tu grupu naizmenično uključuju i isključuju sa periodom od 0.5s. Napisati kod za Arduino platformu koristeći mehanizam prekida i praktično potvrditi uspešnost realizacije .
2. Realizovati sistem sa dve diode (crvena i zelena) i jednim tasterom gde se pritiskom na taster selektuje dioda koja treperi. Selektovana dioda treperi sa periodom od 1s. Prilikom pokretanja sistema crvena dioda je selektovana. Promena selektovane diode se vrši samo ukoliko je u trenutku pritiska tastera selektovana dioda uključena. U suprotnom treba ignorisati pritisak tastera. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost koristeći mehanizam prekida i tajmera i praktično potvrditi uspešnost realizacije.
3. Realizovati sistem sa dva tastera i dve diode. Upotrebom prvog tastera se vrši selekcija trenutno aktivne diode, dok drugi taster uvećava periodu treperenja diode. Perioda treperanja diode uzima vrednosti iz niza vrednosti 0.5s, 1s, 1.5s, 2s. Kada se uzme poslednji element iz niza, nova perioda se uzima sa početka niza. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost koristeći mehanizam prekida tajmera i praktično potvrditi uspešnost realizacije.

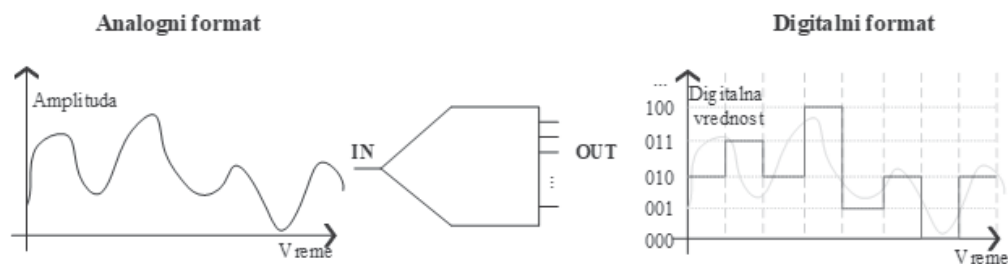




8. ANALOGNI ULAZI I SENZORI

Mikrokontroleri najčešće služe za implementaciju upravljačkih algoritama koji regulišu rad različitih električnih i mehaničkih sistema. Oni pomoću senzora prikupljaju informacije o sistemu kojim upravljaju. Senzori su uređaji koji različite fizičke veličine (kao što su temperatura, vlažnost, pritisak, sila, masa, brzina, itd.) pretvaraju u električni napon.

Fizičke veličine koje se mere, a samim tim i električni napon na izlazu senzora su analogne veličine, što znači da ih odlikuje kontinualna promena vrednosti. U konačnom vremenskom intervalu analogna fizička veličina ima beskonačno mnogo različitih vrednosti. Sa druge strane mikrokontroler je diskretna mašina koja može da operiše sa konačnim setom vrednosti (odnosno brojeva). Zbog toga se, radi obrade u mikrokontroleru, analogni signali prevode u diskretni oblik. Ovo se radi uz pomoć uređaja koji se naziva analogno-digitalni konverter. Ovaj uređaj prvo vrši vremensku diskretizaciju analognog signala tako što uzima njegove vrednosti samo u određenim vremenski ekvidistantnim trenucima. Drugi korak analogno digitalne konverzije je kvantizacija u okviru koje se vremenski diskretizovane analogne vrednosti zaokružuju na najbliže digitalne vrednosti iz konačnog skupa mogućih vrednosti. Na ovaj način, analogni signal se prevodi u konačan skup brojeva sa kojima mikrokontroler može da manipuliše. Opisani postupak diskretizacije analognih signala ilustrovan je na slici 8 - 1.



Slika 8 - 1 - Proces AD konverzije

Jedna od najbitnijih karakteristika AD konvertora jeste broj bita (rezolucija konvertora) koji se koriste za mapiranje ulaznih analognih vrednosti. Drugim rečima, ovaj broj određuje broj nivoa na koje će ulazna analogna veličina biti zaokruživana u toku analogno-digitalne konverzije.

Pored broja bita, bitnu karakteristiku AD konvertora predstavlja i naponski opseg konvertora. Opseg konvertora određuje koliki opseg analognih vrednosti je moguće dovesti na ulaz AD konvertora.

Ovaj opseg se najčešće poklapa sa naponom napajanja mikrokontrolera. Ukoliko poznajemo opseg i broj bita idealnog AD konvertora možemo izračunati veličinu koja se naziva korak AD konvertora (K), a koja se izračunava kao:

$$K = \frac{\text{Opseg}}{2^{\text{BROJ BITA KOVERTORA}} - 1} \quad (6)$$

Kada izračunamo korak konvertora možemo odrediti koja će digitalna vrednost biti na izlazu konvertora (D) ako se na njega dovede određeni napon (U):

$$D = \frac{U}{K} = \frac{U}{\text{Opseg}} \cdot (2^{\text{BROJ BITA KOVERTORA}} - 1) \quad (7)$$

Vrednost D se zaokružuje na manji ceo broj. U svrhu ilustracije načina rada konvertora razmotrimo sledeći primer:

AD konvertor konvertuje analogne vrednosti od 0V do 5V i ima rezoluciju od 10 bita. Izračunati korak AD konvertora kao i vrednosti digitalnih ekvivalenata na izlazu ovog konvertora kada se na ulaz dovedu analogne vrednosti od 5V, 2.5V i 0V.

Na osnovu formule (6) dobijamo da je u ovom slučaju korak AD konverzije

$$K = \frac{5V - 0V}{2^{10} - 1} = \frac{5}{1023} [V]$$

Prema formuli (7) odgovarajući digitalni ekvivalent ako na ulaz dovedemo 5V je:

$$D = \frac{U}{K} = \frac{5}{\frac{5}{1023}} = 1023$$

U slučaju da na ulaz dovedemo napon od 2.5V na osnovu formule (7) dobijamo:

$$D = \frac{U}{K} = \frac{2.5}{\frac{5}{1023}} = \frac{1023}{2} = 511$$

Primenom iste formule dobija se da je za ulazni napon od 0V digitalni ekvivalent na izlazu AD konvertora jednak 0.

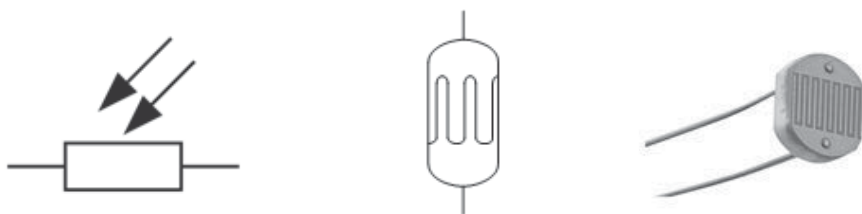
8.1. ANALOGNI ULAZI

Pinovi mikrokontrolera koji omogućavaju povezivanje AD konvertora sa eksternim električnim kolima nazivaju se analogni ulazi. Na Arduino UNO platformi postoji 6 pinova koji se mogu koristiti kao analogni ulazi (A0..A5). Na ove pinove može da se dovodi analogni naponski signal u opsegu od 0-5V. Analogno-digitalni konvertori povezani sa ovim pinovima su desetobitni konvertori koji ulazni napon u opsegu od 0-5V pretvaraju u celobrojne vrednosti u opsegu od 0-1023. Najčešće se na analogne ulaze dovode izlazi sa analognih senzora.

U nastavku ćemo detaljnije razmotriti tipične primere ovakvih senzora kao što su fotootpornik i senzor temperature.

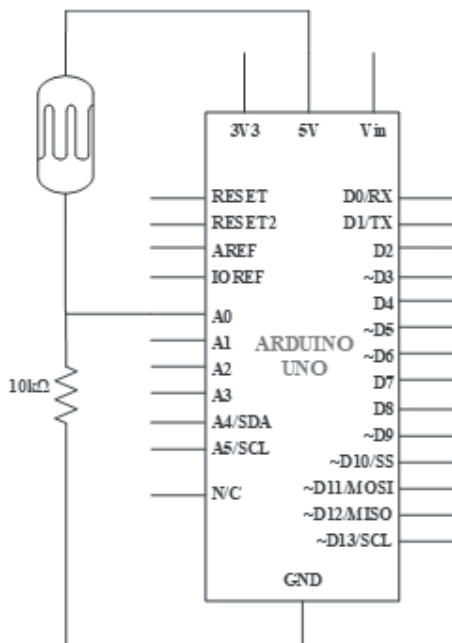
8.2. FOTOOTPORNIK

Fotootpornik predstavlja otpornik čija otpornost zavisi od intenzita svetlosti koja pada na njegovu površinu. Što je intenzitet svetlosti veći, otpornost fotootpornika je manja i obrnuto. Kada ima svetlosti otpornost ovog otpornika se kreće u granicama od 5-10k Ω dok u slučaju odsustva svetlosti, vrednost otpornosti iznosi oko 200k Ω . Tipični električni simboli fotootpornika i njegov fizički izgled prikazani su na slici 8 - 2.



Slika 8 - 2 - Električni simboli (levo i u sredini) i fizički izgled (desno) fotootpornika

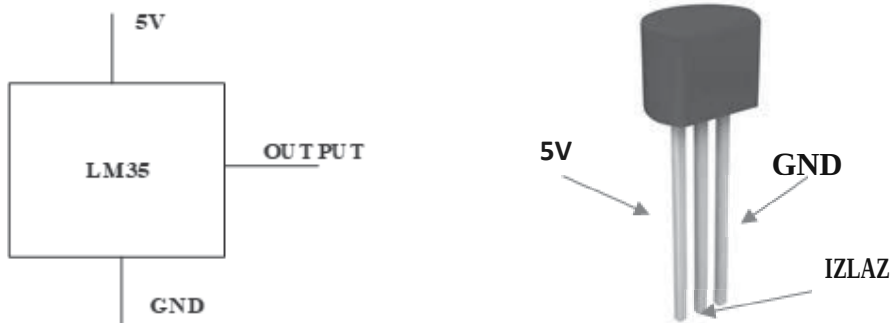
Fotootpornik se najčešće u ostatak kola vezuje redno sa otpornikom poznate i fiksne otpornosti. Najčešće je to otpornik od 10k Ω . Na ovaj način se obezbeđuje da se na mestu spajanja dva otpornika, u slučaju napajanja sistem naponom od 5V obezbedi 2.5V kada je osvetljenje jako ili skoro 0V u slučaju odsustva svetlosti. Način vezivanja fotootpornika u kolo mikrokontrolera prikazan je na slici 8 - 3.



Slika 8 - 3 - Način vezivanja fotootpornika u kolo

8.3. SENZOR TEMPERATURE

Kao senzor temperature u ovom priručniku korist ćemo LM35 [8]. Električni simbol ovog senzora kao i njegov fizički izgled prikazani su na slici 8 - 4.



Slika 8 - 4 - Električni simbol (levo) i fizički izgled LM35 senzora temperature

Ovaj senzor ima tri pina: GND – pin koji povezujemo sa masom sistema, 5V – pin koji povezujemo na napajanje sistema i izlazni pin senzora. Na izlaznom pinu se generiše napon V_0 , srazmeran temperaturi:

$$V_0 = 10 \frac{\text{mV}}{^{\circ}\text{C}} \cdot T \quad (8)$$

$$T = \frac{V_0}{10 \text{ mV}} \cdot ^{\circ}\text{C} \quad (9)$$

Ukoliko se ovaj senzor koristi sa Arduino UNO hardverskom platformom koja ima AD konvertore rezolucije 10 bita koji konvertuju napon u opsegu od 0V do 5V odgovarajući digitalni ekvivalent ulaznog analognog napona V_0 iznosi:

$$D = \frac{V_0}{5 \text{ V}} \cdot 1023$$

Kada se iz ove formule izrazi napon V_0 dobijamo:

$$V_0 = \frac{D}{1023} \cdot 5 \text{ V}$$

Nakon zamene ovog izraza za V_0 u jednačinu (9), dobija se izraz:

$$T = \frac{D \cdot 5 \text{ V}}{1023 \cdot 10 \text{ mV}} \cdot ^{\circ}\text{C}$$

A nakon pretvaranja mV u V i sređivanja izraza dobijamo:

$$T = \frac{D \cdot 5 \cdot 1000 \text{ mV}}{1023 \cdot 10 \text{ mV}} \cdot ^{\circ}\text{C}$$

$$T = \frac{D}{1023} \cdot 500 ^{\circ}\text{C} \quad (10)$$

Poslednja formula za temperaturu transformiše digitalnu vrednost dobijenu sa izlaza AD konvertora Arduino UNO platforme u vrednost temperature na Celzijusovoj skali i kao takva se koristi kroz primere koji se rešavaju u ovom priručniku.

8.4. PRIMERI

Primer 8.1. – Kontrola intenziteta osvetljenosti diode putem potenciometra

- Realizovati Arduino sistem koji se sastoji od potenciometra i diode. Pomeranje (okretanje) potenciometra dovodi do promene intenziteta osvetljenosti diode. Napisati kod koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

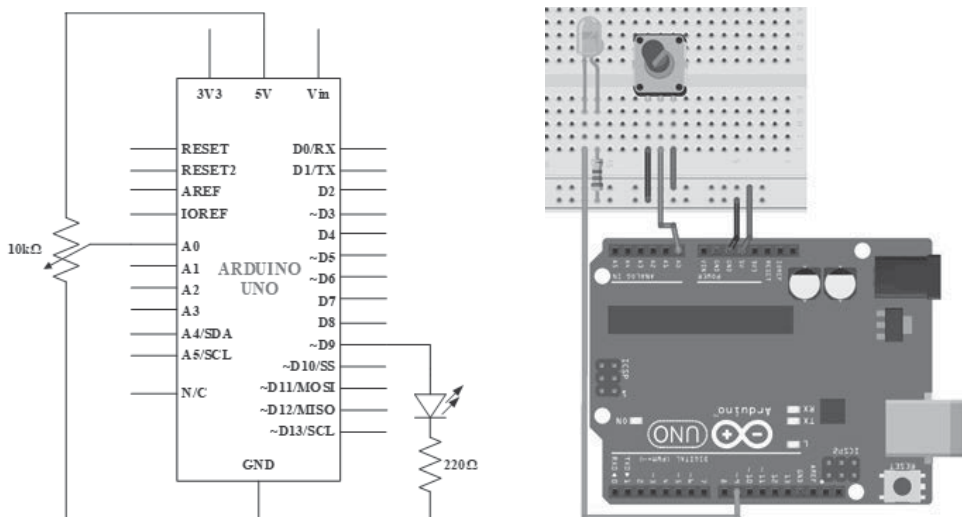
Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Dioda
- 6 MM kratkospajача
- Protobord
- Potenciometar

Električna šema koja ilustruje način povezivanja komponenti, kao i izgled fizičke realizacije sistema prikazani su na slici 8 - 5.

Kao što se sa priloženih električnih šema vidi, potenciometar je priključen na analogni pin A0. Zadatak potenciometra je da promenom otpornosti između pinova A0 i GND menja napon na pinu A0. Odgovarajućom konverzijom ulaznog analognog signala dobijaju se odgovarajući digitalni ekvivalenti (brojevi). Ti digitalni ekvivalenti treba da predstavljaju širinu impulsa signala kojim se menja intenzitet osvetljenosti diode.



Slika 8 - 5 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 8.1

Kod koji realizuje izloženu ideju je sledeći:

```
#include <TimerOne.h>

#define PotentPin A0
const int DiodePinNo = 9;
//With this period we read potentiometer
//analog pin value
const unsigned long SensorReadPeriod = 10000;
//Variable for storing the value we read
volatile unsigned int Value;

void TimerISR(){
    Value = analogRead(PotentPin);
    Value /=4;
}

void setup() {
    pinMode(DiodePinNo, OUTPUT);
    Timer1.initialize(SensorReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}

void loop() {
    analogWrite(DiodePinNo, Value);
}
```

Za čitanje vrednosti sa analognog porta koristi se funkcija

Value = analogRead(PinNo);

PinNo – Broj analognog pina sa koga se očitava vrednost

Value – Digitalni ekvivalent analogne veličine na pinu PinNo

Tajmer generiše prekid u ekvidistantnim vremenskim intervalima. Unutar prekidne rutine tajmera *TimerISR* se čita vrednost sa analognog pina A0 i smešta se u promenljivu *Value*. Vrednost ove promenljive za 10-bitni AD konvertor Arduino UNO hardverske platforme je u opsegu 0-1023 u zavisnosti od položaja potenciometra. Tu vrednost je neophodno skalirati u opseg od 0-255, što je opseg u kome se zadaje duty-cycle PWM signala kojim se određuje intenzitet osvetljenosti diode. Ovo skaliranje se ostvaruje jednostavnim deljenjem sa 4.

Unutar funkcije *setup* vrši se inicijalizacija sistema i tajmera dok se u glavnoj programskoj petlji vrši generisanje PWM signala određenog *DutyCycle*-a što za posledicu ima promenu intenziteta diode.

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
#include <TimerOne.h>
#define ResistorPin      A0
const int DiodePinNo     = 9;
//Here is located readed value from Photoresistor
volatile unsigned int Value;
const int SensorReadPeriod = 20000;
void TimerISR(){
    //Variable for storing the analog value
    Value = analogRead(ResistorPin);
    Value /= 2;
    //constrain value
    Value = constrain(Value,0,255);
}
void setup() {
    //Init pins direction
    pinMode(DiodePinNo,OUTPUT);
    Timer1.initialize(SensorReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}
void loop() {
    //Write Duty cicle
    analogWrite(DiodePinNo,255-Value);
}
```

Za ograničavanje vrednosti promenljivih koristi se funkcija

constrain(value, min, max);

value – vrednost koja se ograničava

min, max – minimalna i maksimalna granica opsega na koji se ograničava vrednost promenljive.

Prikazani kod realizuje očitavanje napona sa razdelnika napona, a zatim tako očitani analogni napon pretvara u digitalnu vrednost. Kao što je pomenuto u uvodu koji se tiče fotootpornika, očekivani napon u toj tački je 2.5V kada ima osvetljenja, odnosno 0V kada nema osvetljenja. Nakon što pročitamo analognu vrednost, neophodno je generisati takav signal na pinu broj 9 koji će omogućiti promenu intenziteta diode u zavisnosti od intenziteta svetlosti. Intenzitet diode je moguće menjati na osnovu generisanog PWM (*Pulse-width modulation*) signala.

Pre *setup* funkcije vrši se definisanje pinova koji će se koristiti. Kao analogni ulazni pin koristi se port A0 dok se kao izlazni pin koristi pin 9 digitalnog porta koji podržava PWM signale. Funkcija *TimerISR* predstavlja servisnu funkciju prekida tajmera i u njoj se očitava vrednost sa analognog pina. Unutar funkcije *setup* vrši se inicijalizacija sistema na već poznat način dok se unutar *loop* funkcije generiše PWM signal.

Primer 8.3. - Interaktivna indikacija stanja osvetljenosti

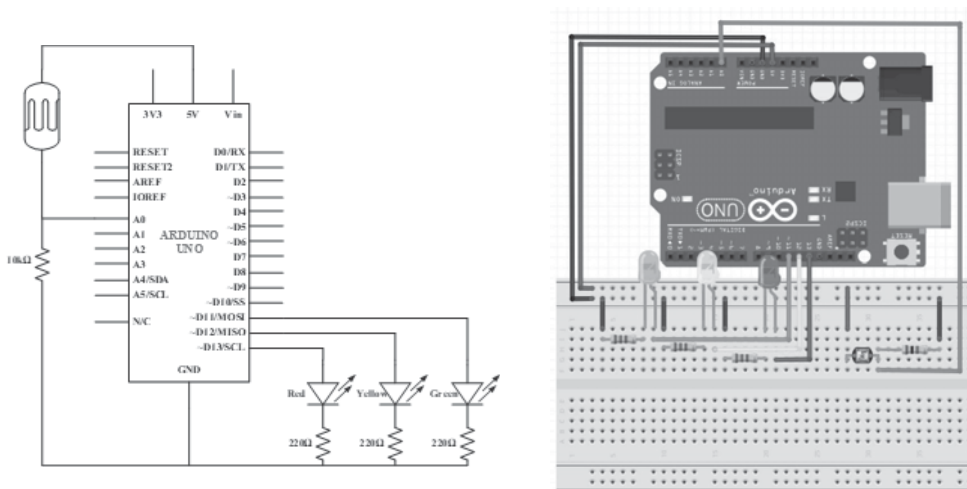
- *Napraviti sistem koji se sastoji od fotootpornika i tri LE diode (crvene, žute i zelene) kojima se signalizira nivo osvetljenosti u prostoriji. Ukoliko je prostorija mračna, treba da sija samo zelena dioda, ukoliko je prostorija srednje osvetljena zelena i žuta, a ukoliko je prostorija jako osvetljena treba da sijaju sve tri diode. Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju sistema su neophodne sledeće komponente:

- Arduino UNO
- 3 otpornika od 220Ω i 1 otpornik od $10k\Omega$
- Cvena, zelena i žuta LE dioda
- Protobord
- 8 MM Kratkospajča
- Fotootpornik

Električna šema sistema, kao i izgled njegove fizičke realizacije prikazani su na slici 8 - 7.



Slika 8 - 7 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 8.3

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
#include <TimerOne.h>
#define ResistorPin A0
const int RedPinNo = 13;
const int YellowPinNo = 12;
const int GreenPinNo = 11;
volatile unsigned int
const unsigned int SensorReadPeriod = 20000;
void TimerISR(){
    Value = analogRead(ResistorPin);
    Value /= 2;
    //constrain value
    Value = constrain(Value,0,255);
}
void setup() {
    //Set pins directions
    pinMode(RedPinNo,OUTPUT);
    pinMode(YellowPinNo,OUTPUT);
    pinMode(GreenPinNo,OUTPUT);
    //Set pins init values
    digitalWrite(RedPinNo,LOW);
    digitalWrite(YellowPinNo,LOW);
    digitalWrite(GreenPinNo,LOW);
    //Init timer
    Timer1.initialize(SensorReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}
void loop() {
    //Always green beacouse Value is
    //always greather than 0
    digitalWrite(GreenPinNo,HIGH);
    if(Value > 255/3){
        digitalWrite(YellowPinNo,HIGH);
        if(Value > 255*2/3){
            digitalWrite(RedPinNo,HIGH);
        }
        else{
            digitalWrite(RedPinNo,LOW);
        }
    }
    else{
        digitalWrite(YellowPinNo,LOW);
    }
}
```

Ideja koja se krije iza prikazanog koda je zapravo poređenje konvertovane vrednosti sa pragovima i ukoliko je vrednost veća od određenih pragova, koji se nalaze na trećinama maksimalne vrednosti, uključuju se odgovarajuće diode. Ukoliko je konvertovana vrednost veća od $\frac{1}{3} * 255$ uključuje se žuta dioda, dok ukoliko je vrednost veća i od $\frac{2}{3} * 255$ uključuje se i crvena dioda. Zelena dioda je uvek uključena pod postavkom da konvertovana vrednost ne može biti manja od 0.

Primer 8.4. – Indikacija temperature prostorije

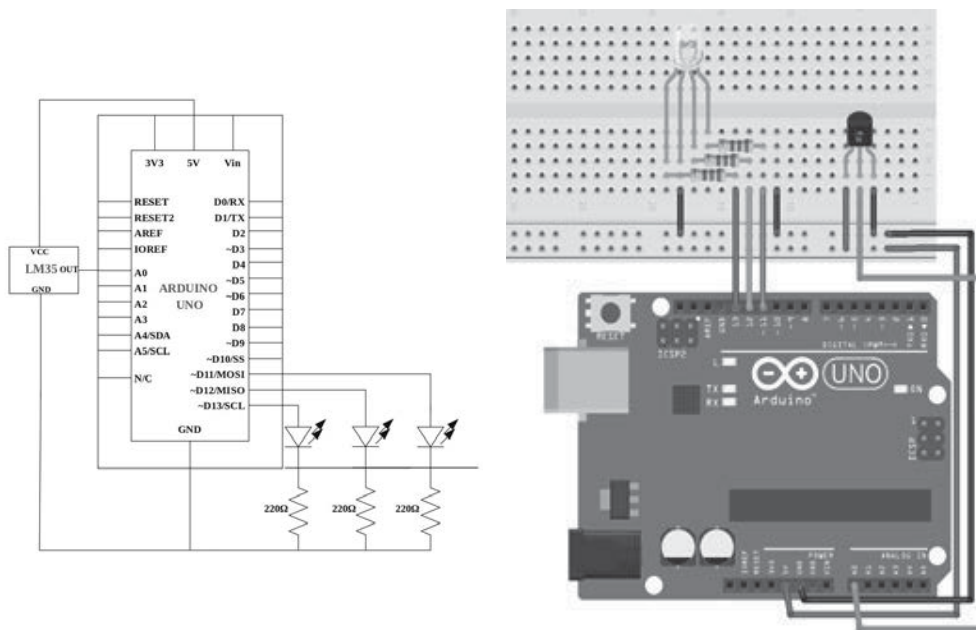
- *Realizovati Arduino sistem koji se sastoji od senzora temperature LM35 i RGB diode koja služi za indikaciju temperature u prostoriji. Ukoliko je temperature ambijenta ispod 18°C, RGB dioda treba da bude plava. Ukoliko je temperatura prostorije između 18°C i 23°C RGB dioda treba da bude zelena, a za temperature ambijenta preko 23°C, RGB dioda treba da bude crvena. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 3 otpornika 220Ω
- RGB dioda
- 10 MM kratkospajача
- Protobord
- LM35 temperaturni senzor

Električna šema sistema kao i izgled njegove fizičke realizacije prikazani su na slici 8 - 8.



Slika 8 - 8 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 8.4.



Prilikom povezivanja senzora temperature treba voditi računa da se pin senzora za napajanje i za GND ne obrnu prilikom povezivanja. Ukoliko do ovoga dođe kućište senzora će se jako zagrejati i osetiće se miris topljenja plastike. Najčešća posledica ovakvog događaja je da senzor temperature više neće ispravno funkcionisati (neće davati realno izmerene vrednosti temperature) .

```
#include <TimerOne.h>
#define SensorPin A0
//Timer variable
const unsigned long SensorReadPeriod = 500000;
//microseconds
//Temperature variable
volatile unsigned int Temp;
//RGB Diode pins
const int RedDiodePinNo = 13;
const int GreenDiodePinNo = 12;
const int BlueDiodePinNo = 11;
```

```

void setup() {
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT);
    pinMode(BlueDiodePinNo, OUTPUT);
    digitalWrite(RedDiodePinNo, LOW);
    digitalWrite(GreenDiodePinNo, LOW);
    digitalWrite(BlueDiodePinNo, LOW);
    Timer1.initialize(SensorReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}
void TimerISR()
{
    Temp=(analogRead(SensorPin)*500.0)/1023;
}
void loop() {
    if(Temp < 18){
        digitalWrite(RedDiodePinNo, LOW);
        digitalWrite(GreenDiodePinNo, LOW);
        digitalWrite(BlueDiodePinNo, HIGH);
    }
    else{
        if(Temp <23){
            digitalWrite(RedDiodePinNo, LOW);
            digitalWrite(GreenDiodePinNo, HIGH);
            digitalWrite(BlueDiodePinNo, LOW);
        }
        else{
            digitalWrite(RedDiodePinNo, HIGH);
            digitalWrite(GreenDiodePinNo, LOW);
            digitalWrite(BlueDiodePinNo, LOW);
        }
    }
}
}

```

Pre funkcije *setup* vrši se definisanje promenljivih koje su neophodne za pravilan rad sistema. Promenljiva *SensorReadPeriod* predstavlja periodu pozivanja funkcije tajmera u kojoj se očitava vrednost sa senzora po formuli (10). Nakon toga se definiše promenljiva *Temp* čija vrednost predstavlja trenutnu temperaturu ambijenta. Na kraju se definišu promenljive čije vrednosti predstavljaju brojeve pinova na koje je povezana RGB dioda.

U funkciji *setup* se vrši inicijalizacija sistema tako što se najpre definišu direktije pinova, te se datim izlaznim pinovima dodele inicijalne vrednosti. Potom se izvrši inicijalizacija tajmera i pridruživanje tajmerske prekidne rutine.

U glavnoj programskoj petlji *loop* se vrši određivanje opsega vrednosti u kome se nalazi trenutna temperatura i na osnovu toga se na određene pinove RGB dioda upisuje „1“ ili „0“.



Ako se za realizaciju sistema primeni gore navedeni kod, sistem koji se dobija može postati nestabilan. Razlog tome je osetljivost senzora koja je velika, tj. vrednosti temperature mogu varirati za neki opseg što može predstavljati značajan problem. U cilju stabilizacije rezultata merenja temperature može se primeniti metod usrednjavanja. Nakon primene ovog metoda, vrednost koja predstavlja temperaturu nije direktna vrednost sa senzora već predstavlja srednju vrednost na nekom broju uzoraka.

Kod kojim se implementira usrednjavanje je sledeći:

```
#include <TimerOne.h>
#define SensorPin A0
//Timer variable
const unsigned long SensorReadPeriod = 10000;
//microseconds
//Temperature variable
volatile unsigned int CurrentTemp;
volatile unsigned int TempSum;
volatile unsigned int Temp;
volatile int counter =0;
volatile const int SamplesNo = 100;
//RGB Diode pins
const int RedDiodePinNo = 13;
const int GreenDiodePinNo = 12;
const int BlueDiodePinNo = 11;
void setup() {
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT);
    pinMode(BlueDiodePinNo, OUTPUT);
    digitalWrite(RedDiodePinNo, LOW);
    digitalWrite(GreenDiodePinNo, LOW);
    digitalWrite(BlueDiodePinNo, LOW);
    Timer1.initialize(SensorReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}
void TimerISR()
```

```
{
    CurrentTemp=(5.0*analogRead(SensorPin)*100.0)/1024;
    //Add to the temperature sum;
    TempSum+=CurrentTemp;
    counter++;
    if(counter == SamplesNo){
        //Count average
        Temp = TempSum/SamplesNo;
        //Reset counter and temperature
        TempSum = 0;
        counter = 0;
    }
}

void loop() {
    if(Temp < 18){
        digitalWrite(RedDiodePinNo, LOW);
        digitalWrite(GreenDiodePinNo, LOW);
        digitalWrite(BlueDiodePinNo, HIGH);
    }
    else{
        if(Temp <23){
            digitalWrite(RedDiodePinNo, LOW);
            digitalWrite(GreenDiodePinNo, HIGH);
            digitalWrite(BlueDiodePinNo, LOW);
        }
        else{
            digitalWrite(RedDiodePinNo, HIGH);
            digitalWrite(GreenDiodePinNo, LOW);
            digitalWrite(BlueDiodePinNo, LOW);
        }
    }
}
```

Ovim kodom usrednjava se očitana temperatura, čime se dobija stabilniji rezultat merenja. Da bi primenili ovaj algoritam najpre je neophodno deklarirati sledeće promenljive:

CurrentTemp – Trenutna vrednost temperature (jedan uzorak)

TempSum – Trenutna suma uzoraka

Temp – Srednja vrednost temperature

Counter – Koliko uzoraka je trenutno prikupljeno

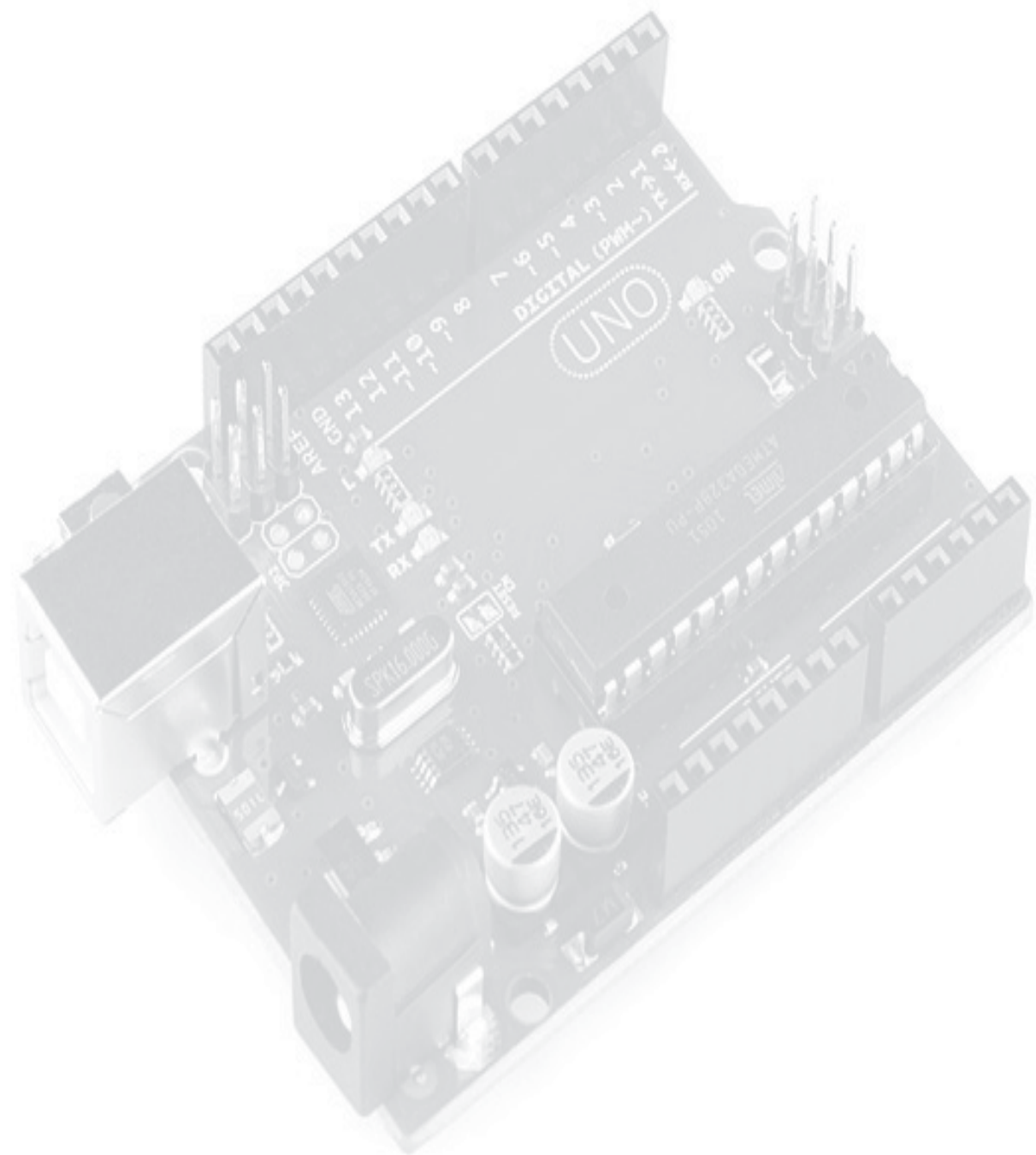
SamplesNo – Koliko uzoraka temperature prikupljamo

Algoritam usrednjavanja je implementiran u funkciji koja servisira prekid tajmera. Ostatak koda je isti kao i u realizaciji bez usrednjavanja. Algoritam funkcioniše tako što

se prikupi određeni broj odbiraka temperature (*SamplesNo*). Ovi odbirci se sabiraju i broje. Kada vrednost brojača (*Counter*) postane jednaka željenom broju odbiraka, očitana temperatura se računa kao prosečna vrednost prikupljenih odbiraka, tako što se suma podeli sa brojem odbiraka. Potom se suma i brojač resetuju na nulu i ceo postupak se ponavlja.

8.5. ZADACI ZA SAMOSTALAN RAD

1. Realizovati Ardino sistem sastavljen od fotootpornika i diode. Potrebno je da realizovani sistem očitava vrednost sa fotootpornika, poredi sa nekim predefinisanim pragom i ukoliko je očitana vrednost veća od praga, dioda se uključuje. U suprotnom, kada očitana vrednost padne ispod datog praga, dioda se isključuje. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizovanog sistema.
2. Realizovati Arduino sistem sačinjen od fotootpornika, 2 tastera i LE diode. Napisati program za Arduino platformu koji očitava vrednost sa analognog porta na koji je priključen fotootpornik, poredi vrednost sa pragom čija se vrednost može menjati (smanjivati ili uvećavati) pritiskom na tastere. Ukoliko je očitana vrednost veća od praga uključiti diodu u suprotnom dioda ostaje isključena. Praktično potvrditi ispravnost realizovanog sistema.
3. Realizovati Arduino sistem sastavljen od temperaturnog senzora LM35 i LE diode. Neophodno je implementirati funkcionalnost sistema gde se dioda uključuje ukoliko temperatura u prostoriji pređe vrednost od 20°C dok je u suprotnom dioda isključena. Praktično potvrditi uspešnost realizovanog sistema.

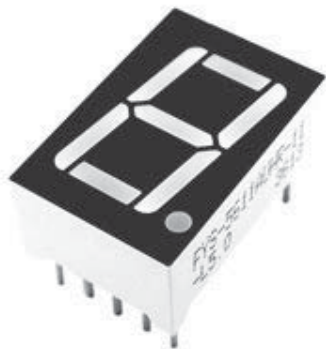


9. DISPLEJI

Ovo poglavlje upoznaje čitaoca sa osnovnim konceptima rada sa displejima. Displeji su periferni hardverski uređaji koji služe za vizuelno prikazivanje informacija korisnicima. U upotrebi je veliki broj različitih vrsta displeja koji se razlikuju po tehnologiji koju koriste za ispis. Ovaj priručnik se bavi displejima sastavljenim od LE dioda (LED displeji), kao i displejima koji rade na bazi svojstava tečnih kristala (LCD displeji), a koji se u praksi najčešće i sreću.

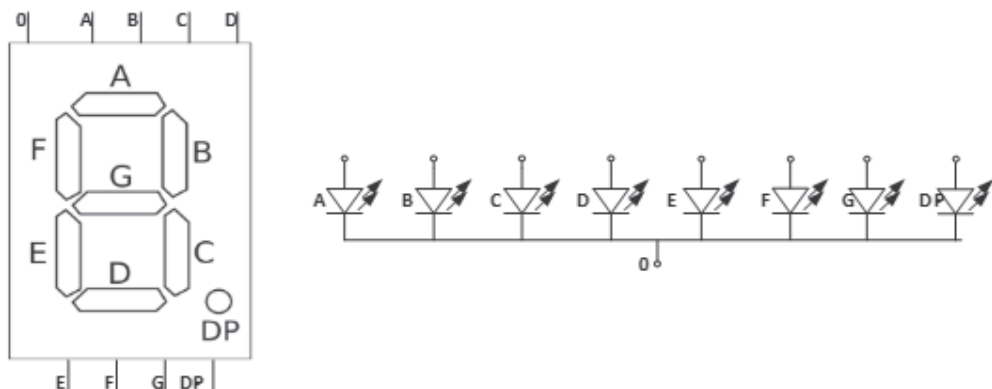
9.2. SEDMOSEGMENTNI LED DISPLEJ

Sedmosegmentni LED displej spada u red najjednostavnijih displeja i može se koristiti za predstavljanje cifara od 0 do 9 kao i pojedinih slovnih karaktera (A,B,C,D,E,F,...), te se često naziva i alfanumeričkim displejom. Tipičan fizički izgled ovog displeja prikazan je na slici 9 - 1.

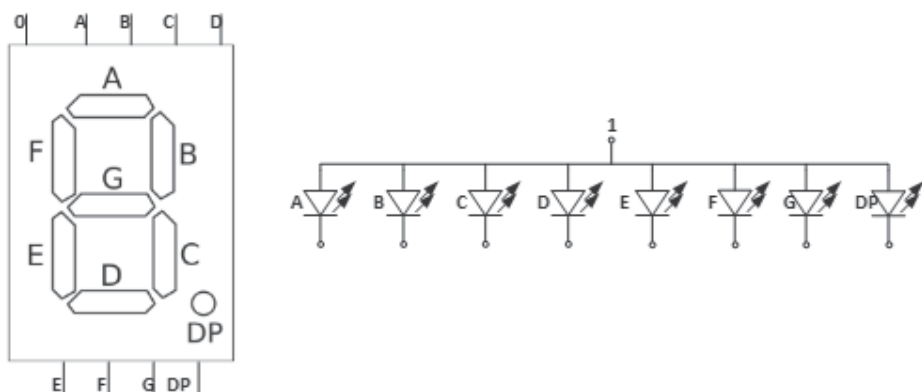


Slika 9 - 1 - Sedmosegmentni led displej

Naziv ovog displeja potiče od činjenice da se sastoji iz sedam jasno diferenciranih segmenata od kojih svaki sadrži po jednu LE diodu. Dioda unutar displeja mogu biti povezane tako da su njihove katode ili anode zajednički povezane u jednu tačku.



Slika 9 - 2 - Električna šema unutrašnjosti sedmosegmentnog displeja (zajednička katoda)



Slika 9 - 3 - Električna šema unutrašnjosti sedmosegmentnog displeja (zajednička anoda)

Iz displeja izlaze pinovi od kojih jedan predstavlja zajedničku katodu (anodu) dok ostali predstavljaju anode (katode) LE dioda ostalih segmenata. Šematski izgled sedmosegmentnog displeja sa zajedničkom katodom prikazan je na slici 9 - 2, dok je električna šema displeja sa zajedničkom anodom prikazana na slici 9 - 3. Kroz ovaj priručnik ćemo koristiti konfiguraciju sa zajedničkom katodom.

Često ovaj displej ima i dodatni segment u vidu tačke iza sedmosegmentnog dela, pa u tom slučaju zapravo ima osam segmenata

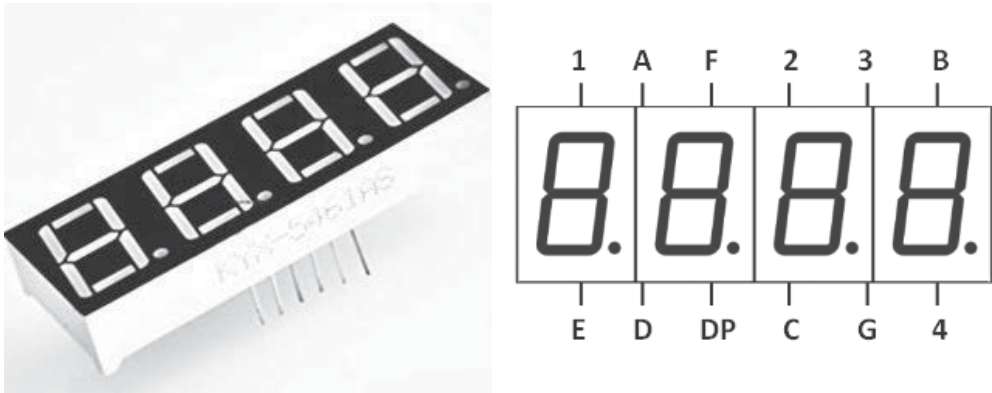
Dovođenjem visokog naponskog nivoa (logičke jedinice) na neki od pinova displeja, aktivira se (počinje da emituje svetlost) odgovarajući segment displeja. Ukoliko želimo da na displeju dobijemo neki smislen sadržaj (cifru ili slovo) neophodno je istovremeno uključivanje više segmenata displeja. U tabeli 9 - 1 je dat pregled cifara i slova koja se najčešće koriste uz informaciju o segmentima koji moraju biti aktivni da bi se odgovarajući simbol ispisa (1 označava da segment treba da bude aktivan, a 0 neaktivan).

Tabela 9 - 1 – Mapiranje segmenata u memorij

Simbol	Aktivni segmenti							
	A	B	C	D	E	F	G	DP
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	0	1	0
3	1	1	1	1	0	0	1	0
4	0	1	1	0	0	1	1	0
5	1	0	1	1	0	1	1	0
6	1	0	1	1	1	1	1	0
7	1	1	1	0	0	0	0	0
8	1	1	1	1	1	1	1	0
9	1	1	1	1	0	1	1	0
A	1	1	1	0	1	1	0	0
B	0	0	1	1	1	1	1	0
C	1	0	0	1	1	1	0	0
D	0	1	1	1	1	0	1	0
E	1	0	0	1	1	1	1	0
F	1	0	0	0	1	1	1	0

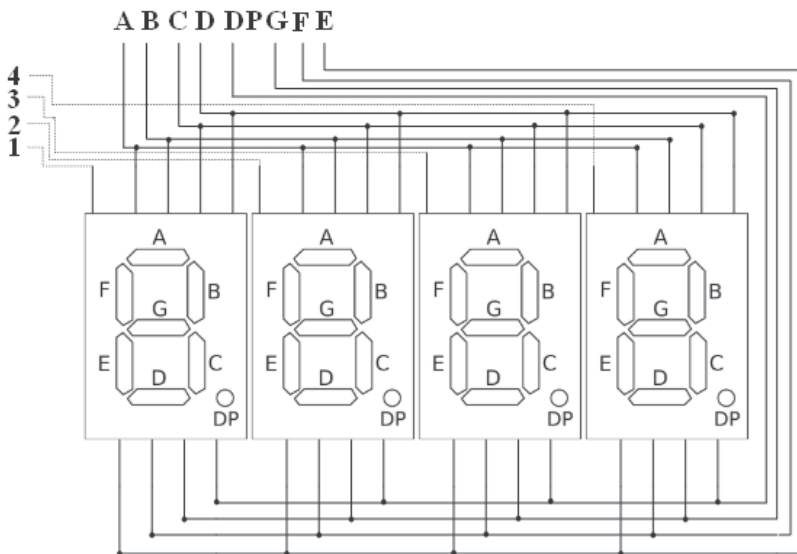
Često se tabela 9 - 1 već nalazi u memoriji platforme sa kojom se radi, implementirana u vidu neke biblioteke, dok je nekada neophodno ovu tabelu implementirati samostalno.

Često postoji potreba da se više cifara ili slova istovremeno prikazuju. U tu svrhu razvijeni su displeji koji sadrže veći broj pojedinačnih sedmosegmentnih displeja u jednom integrisanom kućištu. Fizički izgled kao i električna šema jednog ovakvog displeja prikazani su na slici 9 - 4 i slici 9 - 5 respektivno.



Slika 9 - 4 - Četvorocifreni LED displej

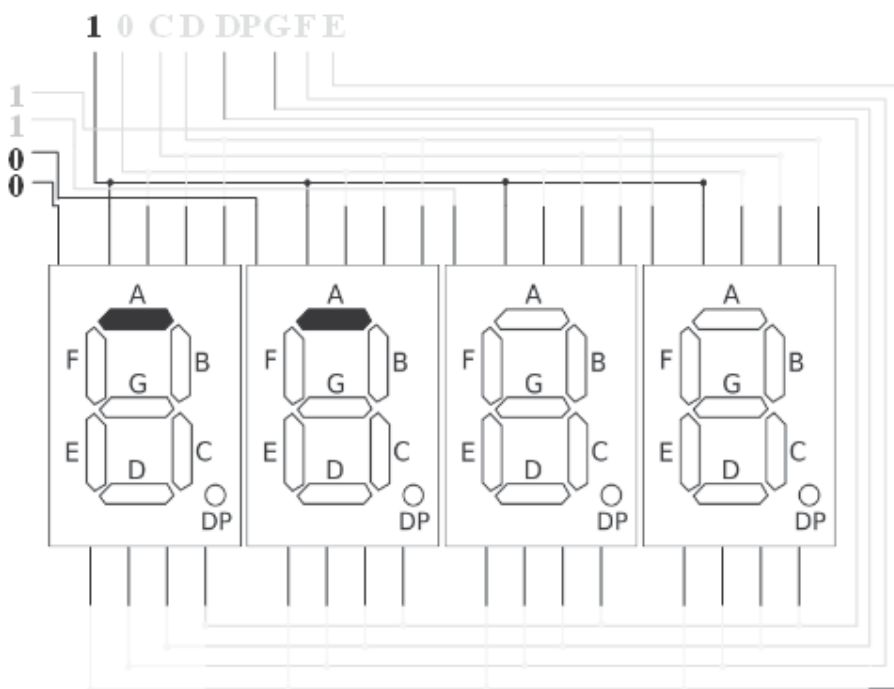
Ovakav displej ima 12 pinova od kojih 4 pina služe za selekciju željenog displeja, a preostalih 8 za aktivaciju odgovarajućih segmenata u okviru odabranog displeja. U slučaju displeja sa zajedničkom katodom odabir nekog (ili svih) displeja vrši se dovođenjem logičke nule na odgovarajući pin za odabir displeja, dok se aktivacija željenih segmenata u okviru odabranih displeja vrši dovođenjem logičke jedinice na pinove povezane sa odgovarajućim segmentima. Na šemi sa slike 9 - 5 brojevima od 1 do 4 označeni su pinovi za aktivaciju displeja dok su slovima od A do G označeni pinovi koji aktiviraju određene segmente.



Slika 9 - 5 - Električna šema četvorocifrenog LED displeja

Pinovi za selekciju displeja su zapravo zajedničke katode pojedinačnih displeja. Zbog toga dovođenje visokog naponskog nivoa na neki od ovih pinova onemogućava paljenje bilo kog segmenta na odgovarajućem displeju. Sa druge strane anode svakog od segmenata na različitim displejima su kratko spojene (otuda samo 8 pinova za aktivaciju pojedinačnih segmenata, a ne 32), tako da ukoliko se na neki od ovih pinova dovede visok naponski nivo, aktiviraće se odgovarajući segment na svim displejima čije su zajedničke katode na logičkoj nuli. Slika 9 - 6 ilustruje situaciju u kojoj su aktivna dva displeja na levoj strani (dok su druga dva neaktivna) i gde je aktiviran segment A koji svetli na oba aktivna displeja.

Na primer, ukoliko želimo da prikazemo broj 4 samo na prvom displeju, potrebno je dovesti logičku nulu na pin za selekciju prvog displeja, a logičku jedinicu na ostale selekzione pinove. Na pinove od A do G treba dovesti kombinaciju „0“ i „1“ koja rezultuje paljenjem segmenata koji čine broj 4 (videti tabeli 9 - 1). Ukoliko se želi ispis broja 4 na svim displejima, potrebno je na pinove od 1 do 4 dovesti logičku nulu. Kako su segmenti displeja zapravo LE diode, prilikom povezivanja displeja potrebno je anode povezati na digitalne pinove Arduina preko otpornika koji ograničavaju struju kroz aktivne segmente displeja (videti diskusiju u poglavlju 4.2).



Slika 9 - 6 – Ilustracija aktivacije različitih segmenata na četvorocifrenom sedmosegmentnom displeju



Da bi se ovaj displej mogao koristiti sa Arduino platformom najpre je neophodno instalirati biblioteku 7Seg, a nakon toga povezati displej na način prikazan na slici 9 - 7.

9.3. LCD (LIQUID CRYSTAL DISPLAY)

Rad LCD displeja bazira se na svojstvu tečnih kristala da polarizuju svetlost. Tečni kristal ne može direktno da emituje svetlost, već se koristi pozadinski izvor svetlosti, koja onda prolazi kroz sloj tečnog kristala. Pojedini delovi površine ispunjene tečnim kristalom vrše drugačiju polarizaciju svetlosti od okolnog kristala, što se na površini displeja vidi kao svetlost različite boje i na ovaj način se na displeju može ispisivati željeni sadržaj. Fizički izgled LCD displeja koji se koristi u ovom priručniku dat je na slici 9 - 8.



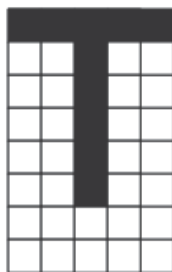
Slika 9 - 8 - LCD displej

Ovaj displej naziva se još i 16x2 displej (oznaka LCD-WH1602B-TMI-ET [10]), što znači da se pomoću njega maksimalno mogu prikazati 2 reda od po 16 karaktera. Displej ima plavo pozadinsko osvetljenje i bela slova. Suprotno displeju nalaze se pinovi preko kojih se LCD displej povezuje sa Arduinoom. Tabela 9 - 2 daje spisak i opis funkcionalnosti ovih pinova.

Tabela 9 - 2 - Značenje pinova LCD displeja

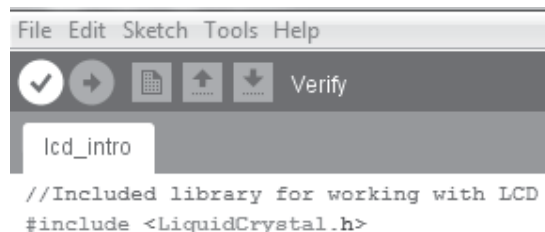
Broj pina	Simbol	Značenje
1	Vss	Masa (Ground)
2	Vdd	Napajanje 5V
3	VO	Pin za podešavanje kontrasta
4	RS	Kada je ovaj pin na visokom nivou (5V) prenose se podaci u suprotnom prenose se komande
5	R/W	Kada je ovaj pin na visokom nivou (5V) upisuje se u modul u su-protnom se čita iz modula
6	E	Selektuje čip
7-14	DB0-DB7	Linije za prenos informacija
15	A	Napajanje za pozadinsko osvetljenje
16	K	Masa (Ground) za pozadinsko osvetljenje

Svaki od 16x2 segmenata LCD displeja je zapravo matrica 8x5 polja ispunjena tečnim kristalom koji se može električno pobuditi tako da polariše svetlost koja kroz njega prolazi da ona bude bele boje. Polarizaciju polja za svaki segment vrši poseban regulator koji je sastavni deo displeja. Primer polarizacije polja u matrici jednog segmenta za slovo "T" dat je na slici 9 - 9.



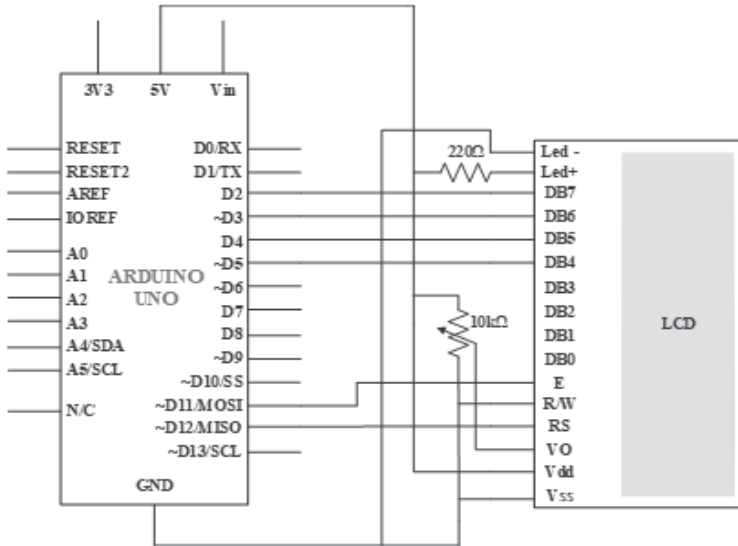
Slika 9 - 9 - Primer ispisa slova T na jednom segmentu LCD displeja

Pristup displeju u Arduino softverskom okruženju je olakšan jer postoji implementirana biblioteka koja omogućava jednostavnu komunikaciju Arduina i displeja. Uključivanje ove biblioteke vrši se tako što se u meni baru izabere Sketch-> Include library i iz izlistanih biblioteka se odabere biblioteka pod nazivom LiquidCrystal. Ukoliko je biblioteka uspešno uključena na početku programa će se pojaviti segment kao na slici 9 - 10.

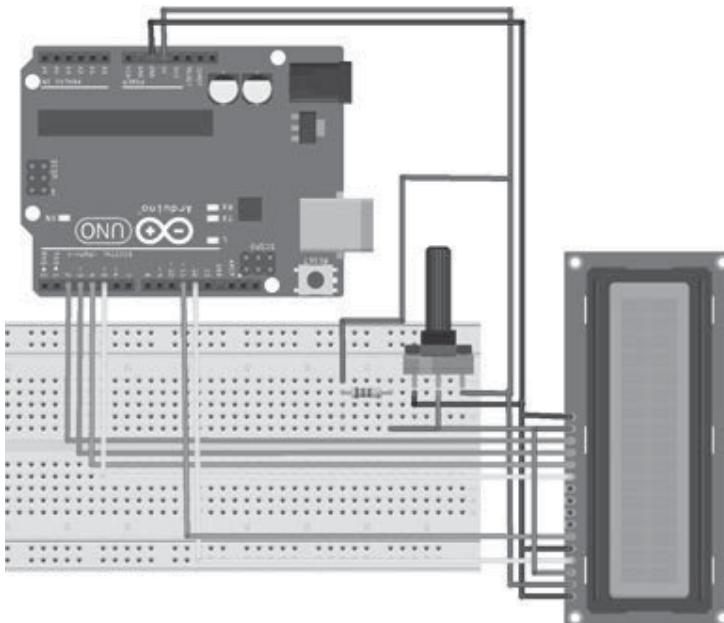


Slika 9 - 10 - Izgled zaglavlja programskog koda kada je biblioteka za rad sa LCD displejom ispravno uključena

Električna šema koja pokazuje ispravan način povezivanja ovog modula sa Arduino platformom prikazana je na slici 9 - 11, dok je način povezivanja displeja u Arduino okruženju prikazan na slici 9 - 12.



Slika 9 - 11 - Električna šema povezivanja Arduina i LCD displeja



Slika 9 - 12 - Povezivanje LCD displeja sa Arduino hardverskom platformom

9.4. PRIMERI

Primeri koji slede imaju za cilj da čitaocu demonstriraju rad sa LED i LCD displejima i da uvedu neke osnovne koncepte pri radu sa ova dva tipa displeja.

Primer 9.1. - Ispis jednocifrenog broja na LED displej

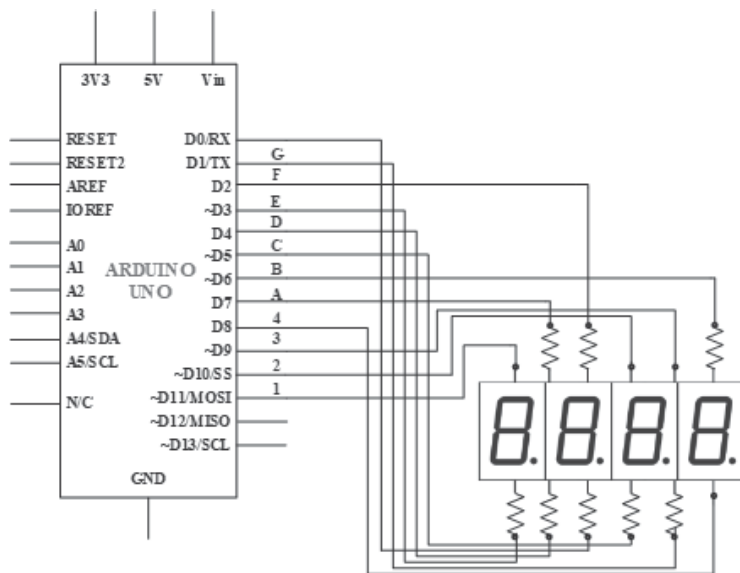
- Realizovati Arduino sistem koji se sastoji od LED displeja gde se na LED displeju ispisuju cifre od 0-9. Na početku je vrednost koja se ispisuje na displeju 0, a zatim se sa periodom od 1 sekunda vrednost uvećava za 1. Kada vrednost na displeju stigne do 9, u narednoj sekundi se vraća na 0 i ciklus se ponavlja. Napisati kod za Arduino hardversku platformu koji realizuje zahtevanu funkcionalnost i potvrditi uspešnost realizovanog sistema.

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 8 otpornika od 220Ω
- Protobord
- 12 MŽ Kratkospajča
- Četvorocifreni LED displej

Električna šema sistema prikazana je na slici 9 - 13.



Slika 9 - 13 -Električna šema sistema iz primera 9.1

Ovaj primer ima za cilj da demonstrira jednostavan ispis jednocifrenog broja na LED displej. Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//Include library for working with 7 segments LED display
#include <7Seg.h>
#include <TimerOne.h>
//This variable stores current counter value
volatile int Counter = 0;
const unsigned long TimerPeriod = 1000000;
void TimerISR(){
    if(Counter == 9){
        //return caunter to 0
        Counter = 0;
    }
    else{
        //increment counter
        Counter++;
    }
}
void setup() {
    //First init LED diplay
    Init7Seg();
    //Turn on display number 4
    TurnDspOn(4);
    Timer1.initialize(TimerPeriod);
    Timer1.attachInterrupt(TimerISR);
}
void loop() {
    //Write counter value to display
    WriteNum(Counter);
}
```

Funkcije koje se koriste u ovom primeru su funkcije iz biblioteke 7seg. Da bi korišćenje četvorocifrenog sedmosegmentnog LED displeja bilo moguće najpre je neophodno izvršiti inicijalizaciju sistema pozivom funkcije

Init7Seg();

Jednocifreni displej iz niza od 4 jednocifrena displeja se uključuje pozivom funkcije

TurnDspOn(DSPNumber);

DSPNumber – Broj displeja od 1 do 4 koji želimo da uključimo

Ukoliko želimo da ispišemo neku cifru na prethodno uključeni displej pozivamo funkciju

WriteNum(Digit);

Digit – Cifra od 0-9 koju ispisujemo na sedmosegmentni displej.

Unutar *setup* funkcije vrši se inicijalizacija LED displeja i uključivanje četvrtog LED displeja. Takođe se unutar ove funkcije inicijalizuje tajmer, tako da perioda tajmera bude *TimerPeriod*. Unutar glavne programske petlje *loop* vrši se ispis promenljive *Counter* na sedmosegmentni LED displej. Promena vrednosti promenljive koja se ispisuje na displeju vrši se unutar prekidne funkcije tajmera. Unutar ove funkcije najpre se vrši provera da li je vrednost prešla broj 9 i ukoliko jeste, vraća se na 0 dok se u suprotnom vrši uvećanje vrednosti promenljive *Counter* za 1.

Primer 9.2. - Promena aktivnog LED displeja

- *Realizovati Arduino sistem koji se sastoji od četvorocifrenog LED displeja gde se na displeju prikazuje trenutna vrednost brojačke promenljive. Brojačka promenljiva se kreće u ospegu od 0-9. U početku je vrednost ove promenljive prikazana na displeju broj 4. Kada promenljiva dobroji do kraja (broj 9) počinje sa brojanjem od 0 ali sada na displeju broj 3. Kada završi brojanje na displeju broj 3 počinje novo brojanje na displeju broj dva. Kada dođe do kraja četvorocifrenog led, displeja ciklus brojanja se vraća na početak (displej broj četiri). Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Protobord
- 12 MŽ kratkospajča
- Četvorocifreni LED displej

Električna šema povezivanja data je na slici 9 - 13. Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//Include library for working with
//7 segments LED display
#include <7Seg.h>
#include <TimerOne.h>
//This variable store current counter value
volatile int counter = 0;
volatile int ActiveDsp = 4;
const unsigned long TimerPeriod = 1000000;
void TimerISR(){
    //counter value equal to 9
    if(counter == 9)
    {
        //return caunter to 0
```

```

    counter = 0;
    TurnDspOff(ActiveDsp);
    if(ActiveDsp == 1)
    {
        ActiveDsp = 4;
        TurnDspOn(ActiveDsp);
    }
    else
    {
        ActiveDsp --;
        TurnDspOn(ActiveDsp);
    }
}
else
{
    //increment counter
    counter++;
}
}

void setup() {
    //First init LED display
    Init7Seg();
    //Turn on display number 4
    TurnDspOn(ActiveDsp);
    Timer1.initialize(TimerPeriod);
    Timer1.attachInterrupt(TimerISR);
}

void loop() {
    //Write counter value to display
    WriteNum(counter);
}

```

Vrednost promenljive *ActiveDsp* sadrži informaciju o trenutno aktivnom displeju. Nakon što brojač dobroji do kraja vrši se umanjene ove promenljive prethodno proveravajući da li je njena vrednost 1. Ukoliko jeste, promenljiva uzima vrednost 4, čime se ispis cifara vratio na početak.

Primer 9.3. - Ispis dvocifrenog broja na LED displej (Multipleksiranje)

- Realizovati Arduino sistem koji se sastoji od LED displeja na kome se ispisuju brojevi od 0 do 99. Na početku je vrednost koja se ispisuje na displeju 0, a zatim se sa periodom od 1 sekund uvećava za 1. Kada vrednost na displeju stigne do 99 u narednoj sekundi se vraća na 0 i ciklus se ponavlja. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost praktično potvrditi uspešnost realizacije.

Rešenje:

Električne komponente potrebne za realizaciju ovog sistema kao i električna šema sistema su isti kao i za primer 9.1.

Ako se detaljnije pogleda način povezivanja višecifrenog LED displeja, može se primetiti da su svi segmenti kratko spojeni, što znači da ako se uključe dva displeja i upiše se neki broj, taj broj će biti prikazan na oba displeja. Postavlja se pitanje kako nezavisno na dvocifrenom led displeju prikazati dvocifren broj, tj na svakom od displeja po jednu cifru? Odgovor na ovo pitanje leži u mehanizmu pod nazivom *Multipleksiranje displeja*.

Ideja koja stoji iza mehanizma multipleksiranja je da se najpre aktivira segment za ispis niže cifre, na kome se ispisuje ta cifra. Ovaj segment ostaje aktivan određeno vreme, pa se potom gasi. Umesto njega aktivira se segment za ispis više cifre i na njemu se vrši ispis. I ovaj segment ostaje aktivan određeni vremenski period, da bi se potom ugasio. Nakon toga se ponovo aktivira segment za ispis niže cifre, čime se ceo ciklus ponavlja. Pitanje koje se postavlja jeste koliko dugo treba držati aktivan displej da se ne bi videla promena?

Odgovor na ovo pitanje praktično je dat u diskusiji u okviru primera 6.2, gde je objašnjeno kako se može postići naizmenično paljenje i gašenje LED diode, a da treperenje ne bude vidljivo golim okom. Ovo se ostvaruje ukoliko dioda treperi sa periodom manjom od 20 ms. U ovom slučaju može se primeniti isti princip, samo što sada treba dva displeja da promene sadržaj, a da se promena ne vidi golim okom. Na osnovu ovoga može se izvesti formula koja daje informaciju o maksimalnom vremenu T koliko jedan displej sme da bude aktivan, a da se ne vidi treperenje:

$$T = \frac{20 \text{ ms}}{\text{Broj displeja}} \quad (11)$$

Na osnovu formule (11) dobija se da je maksimalno vreme aktivnosti svakog displeja 10 ms. Ovo vreme se još naziva i periodom osvežavanja sadržaja LED displeja. U nastavku je dat kod koji predstavlja rešenje ovog zadatka.

```
#include <7Seg.h>
#include <TimerOne.h>
int DisplayRefreshTime = 10000; //microseconds
//Currently active dsp;
volatile int NextDspUpd;
int counter = 0;
volatile int NoOfEnter = 0; // in interrupt service routine
//Low digit of counter value
volatile int LowDigit = 0;
//High digit of counter value
volatile int HighDigit = 0;
void setup() {
    //First init LED display
    Init7Seg();
    NextDspUpd = 4;
    Timer1.initialize(DisplayRefreshTime);
    Timer1.attachInterrupt(DisplayRefresh);
}
//Extract low and high digit from Number
void Convert(const int Number)
{
    int Temp = Number;
    //Extract higher digit
    HighDigit = Temp/10;
    //Calculate rest
    Temp%=10;
    //Extract lower digit
    LowDigit = Temp/1;
}
//Timer callback function
//This function refreshes display contents
void DisplayRefresh()
{
    NoOfEnter++;
    switch(NextDspUpd)
    {
        case 3:
            //Turn off display 4
            TurnDspOff(4);
            //Turn on display 3
            TurnDspOn(3);
            //Write digit to dsp3
            WriteNum(HighDigit);
            //Next active display is 4
            NextDspUpd = 4;
            break;
        case 4:
            //Turn off display 3
            TurnDspOff(3);
            //Turn on display 4
```

```

        TurnDspOn(4);
        //Write lower digit
        WriteNum(LowDigit);
        // Next active display is 3;
        NextDspUpd = 3;
        break;
    }
}
void loop() {
    //Call function for extracting lower and higher digit
    Convert(counter);
    //check whether the value has reached 99
    if(NoOfEnter == 100){
        if(counter == 99)
        {
            counter = 0;
        }
        //if not
        else
        {
            //increment counter value
            counter++;
        }
        NoOfEnter = 0;
    }
}

```

Promenljiva *NextDspUpd* čuva informaciju o displeju koji će sledeći da promeni svoju vrednost. To znači da ukoliko je trenutno aktivni displej 3, displej koji sledeći treba da promeni vrednost je 4, dok ukoliko je aktivan displej 4, sledeći displej koji treba da promeni vrednost je 3. Promenljiva *HighDigit* sadrži višu cifru dvocifrenog broja, koja treba da se ispisuje na displeju broj 3, dok promenljiva *LowDigit* sadrži nižu cifru koja treba da se ispisuje na displeju 4.

U funkciji *setup* vrši se inicijalizacija sedmosegmentnog displeja pozivom funkcije *Init7Seg*. Zatim se inicijalizuje tajmer sa periodom izračunatom po formuli (11).

Funkcija *Convert* ima zadatak da iz broja *Number*, koji joj se prosleđuje kao argument, izdvoji višu i nižu cifru i da vrednost smesti u promenljive *HighDigit* i *LowDigit*.

Funkcija *DisplayRefresh* predstavlja prekidnu funkciju tajmera, koja se poziva sa periodom od 10 ms. Ova funkcija ima zadatak da na osnovu promenljive *NextDspUpd* izvrši promenu stanja displeja, tako što prvo isključi trenutno aktivni displej, zatim uključi displej koji se osvežava, a onda ispiše vrednost. Nakon toga, unutar ove funkcije se podesi nova vrednost promenljive *NextDspUpd* na vrednost koja označava sledeći displej čiji sadržaj treba promeniti.

Unutar glavne programske petlje poziva se funkcija *Convert* koja izdvaja višu i nižu cifru iz promenljive *counter* a zatim proverava da li je vrednost promenljive *counter* dostigla maksimum. Na kraju se odlaže promena vrednosti promenljive *counter* za jednu sekundu.

Primer 9.4. - Ispis sadržaja na LCD displej

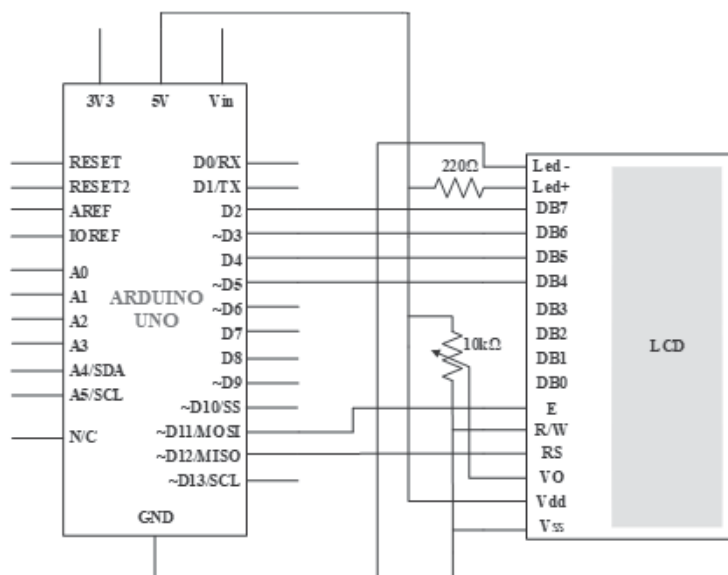
- Realizovati Arduino sistem koji se sastoji od LCD displeja i Arduino hardverske platforme. Na LCD displeju treba da se ispiše "Internet Stvari". Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizovanog sistema.

Rešenje:

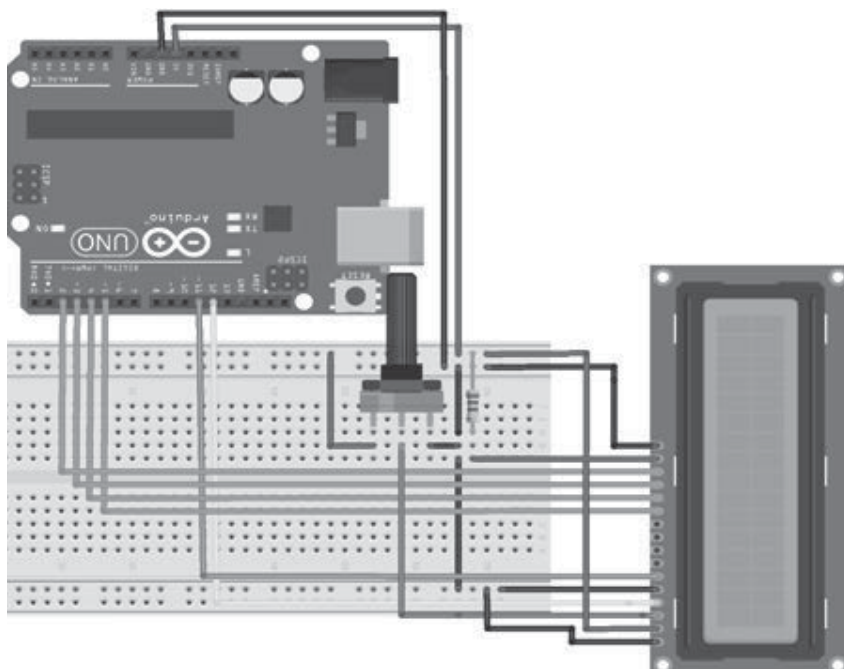
Za realizaciju sistema neophodne su sledeće komponente:

- Arduino UNO
- Otpornik 220Ω
- Protobord
- 14 MM kratkospajача
- Potenciometar $10k\Omega$
- LCD displej

Električna šema sistema, kao i hardverska realizacija prikazani su na slikama 9 - 14 i 9 - 15 respektivno.



Slika 9 - 14 - Električna šema sistema iz primera 9.



Slika 9 - 15 - Hardverska realizacija sistema iz primera 9.4

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//Included library for working with LCD
#include <LiquidCrystal.h>
//Display size
const int DispHeight = 16;
const int DispWidth = 2;
//LCD Pin number
const int RSPinNo = 12;
const int EnablePinNo = 11;
const int D7PinNo = 2;
const int D6PinNo = 3;
const int D5PinNo = 4;
const int D4PinNo = 5;
//Create LiquidCrystal variable and initialize it
LiquidCrystal LCD(RSPinNo, EnablePinNo,
                  D4PinNo, D5PinNo, D6PinNo, D7PinNo);

void setup() {
    //Init display size
    LCD.begin(DispHeight, DispWidth);
    LCD.print("Internet Stvari");
}

void loop() {
}
```

Pre nego što se započne korišćenje bilo koje od funkcija iz biblioteke *LiquidCrystal*, neophodno je definisati promenljivu tipa *LiquidCrystal* na sledeći način:

LiquidCrystal LCD(rs, enable, DB4, DB5, DB6, DB7);

rs – Broj pina na Arduino hardverskoj platformi na koji je povezan RS pin sa LCD displeja

enable – Broj pina na Arduino hardverskoj platformi na koji je povezan Enable pin sa LCD displeja

DB4,DB5,DB6,DB7 – brojevi pinova na Arduino hardverskoj platformi na koje su povezani pinovi LCD displeja za prenos podataka D4, D5, D6 i D7 respektivno.

Nakon uspešne definicije promenljive LCD mogu se koristiti funkcije iz ove biblioteke. Najpre je neophodno inicijalizovati širinu displeja, što se vrši pozivom funkcije

begin(cols, rows);

cols – broj kolona koje ima displej (u našem slučaju 16)

rows – broj redova koje ima displej (u našem slučaju 2)

Za ispis sadržaja na LCD displej koristi se sledeća funkcija:

print(data);

data – podatak koji će biti prikazan na displeju. Podržani tipovi podataka su *char*, *byte*, *int*, *log* ili *string*.

Nakon uključivanja biblioteke za rad sa LCD displejom, definišu se dve promenljive *DispHeight* i *DispWidth* koje nose informaciju o visini i širini displeja. Nakon toga definiše se broj pinova na Arduino platformi na koje su povezani odgovarajući pinove LCD displeja. Zatim se definiše promenljiva *LCD* tipa *LiquidCrystal* koja će biti glavna promenljiva pri radu sa LCD displejom.

Unutar *setup* funkcije vrši se inicijalizacija širine displeja pozivom funkcije *begin*, a nakon toga se na displeju ispisuje poruka „Internet stvari“. Funkcija *loop* je prazna i u ovom primeru praktično samo omogućava da slova ispisana na displeju u okviru *setup* funkcije ostanu ispisana sve dok Arduino ima napajanja.

Primer 9.5. - Promena sadržaja na LCD ekranu pritiskom na taster

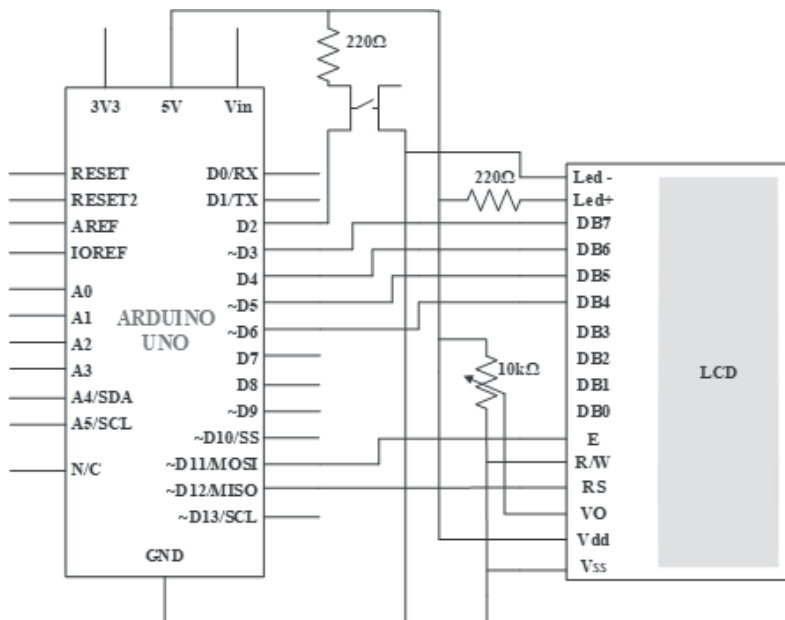
- Realizovati Arduino sistem koga čine Arduino hardverska platforma, taster i LCD displej, kod koga se tekst ispisan na displeju menja pritiskom na taster. Tekstualni sadržaj koji bi trebalo da se menja na ekranu usled pritiska na taster je vaše ime, prezime i "Internet stvari". Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizovanog sistema.

Rešenje:

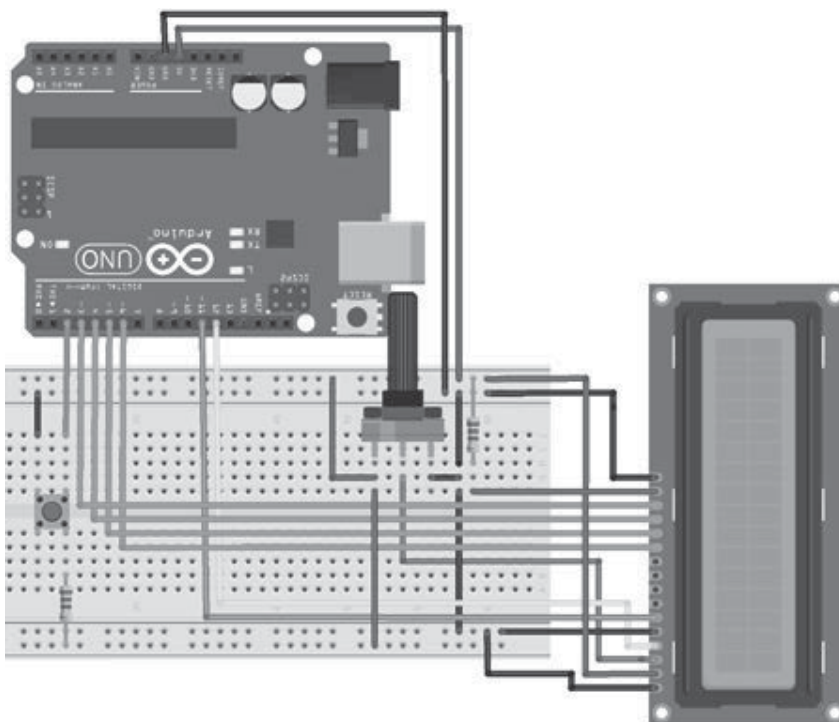
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 2 otpornika od 220Ω
- Protobord
- 22 MM kratkospajача
- Taster
- Potenciometar $10k\Omega$
- LCD displej

Električna šema koja ilustruje način povezivanja komponenti, kao i fizička realizacija sistema prikazani su na slikama 9 - 16 i 9 - 17 respektivno.



Slika 9 - 16 - Električna šema sistema iz primera 9.5



Slika 9 - 17 - Hardverska realizacija sistema iz primera 9.5

Ovaj primer se od prethodnog između ostalog razlikuje po načinu povezivanja. Na električnoj šemi sistema treba primetiti da je za razliku od primera 9.4. (Slika 9 - 14) sada izvršena rekonfiguracija načina povezivanja pinova DB7, DB6, DB5 i DB4 LCD displeja sa Arduino hardverskom platformom. Razlog ove promene je to što je na pin broj 2, koji ima mogućnost generisanja prekida, sada povezan taster.

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//Included library for working with LCD
#include <LiquidCrystal.h>
const int ButtonPinNo = 2;
//LCD message string
String Message = "";
// Message type
typedef enum {FirstName, LastName, Topic} MsgType;
volatile MsgType CurrentMsgType;
//Display size
const int DispHeight = 16;
const int DispWidth = 2;
//LCD Pin number
const int RSPinNo = 12;
```

```

const int EnablePinNo = 11;
const int D7PinNo = 3;
const int D6PinNo = 4;
const int D5PinNo = 5;
const int D4PinNo = 6;
//Create LiquidCrystal variable and initialize it
LiquidCrystal LCD(RSPinNo, EnablePinNo,
                  D4PinNo, D5PinNo, D6PinNo, D7PinNo);

void setup() {
    //Init display size
    LCD.begin(DispHeight, DispWidth);
    //Init variables
    CurrentMsgType = FirstName;
    WriteToLCD();
    //Init Button
    pinMode(ButtonPinNo, INPUT);
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), WriteToLCD, FALLING);
}

void WriteToLCD()
{
    //Clear old content
    LCD.clear();
    //Prepare message to write
    switch(CurrentMsgType)
    {
        case FirstName:
            Message = "Ime";
            CurrentMsgType = LastName;
            break;
        case LastName:
            Message = "Prezime";
            CurrentMsgType = Topic;
            break;
        case Topic:
            Message = "Internet Stvari";
            CurrentMsgType = FirstName;
            break;
    }
    //Print message
    LCD.print(Message);
}

void loop() {
}

```

Brisanje ispisanog sadržaja sa LCD displeja vrši se pozivom funkcije:

clear();

U ovom primeru uveden je nabrojivi tip podataka (*enum*) koji je označen sa *MsgType*. Vrednosti koje promenljive ovog tipa mogu uzeti su *FirstName*, *LastName* i *Topic*. Ovaj tip podataka je uveden da bi program imao informaciju o tome koji od podataka se trenutno ispisuje na LCD displeju i ta informacija se čuva u promenljivoj ovog tipa *CurrentMsgType*.

Pritiskom na taster se ulazi u prekidnu rutinu *WriteToLcd* gde na osnovu trenutnog ispisa menjamo sadržaj LCD displeja. Sadržaj na LCD displeju se menja tako što najpre izbrišemo prethodni sadržaj pozivom funkcije *clear()*, a zatim ispišemo novi sadržaj. Novi sadržaj se ispisuje pozivom funkcije *print()* koja kao argument prima string kome je prethodno dodeljena vrednost. Dodela vrednosti stringu se vrši unutar ove funkcije u *switch-case* strukturi.

Primer 9.6. - Merenje temperature i osvetljenosti

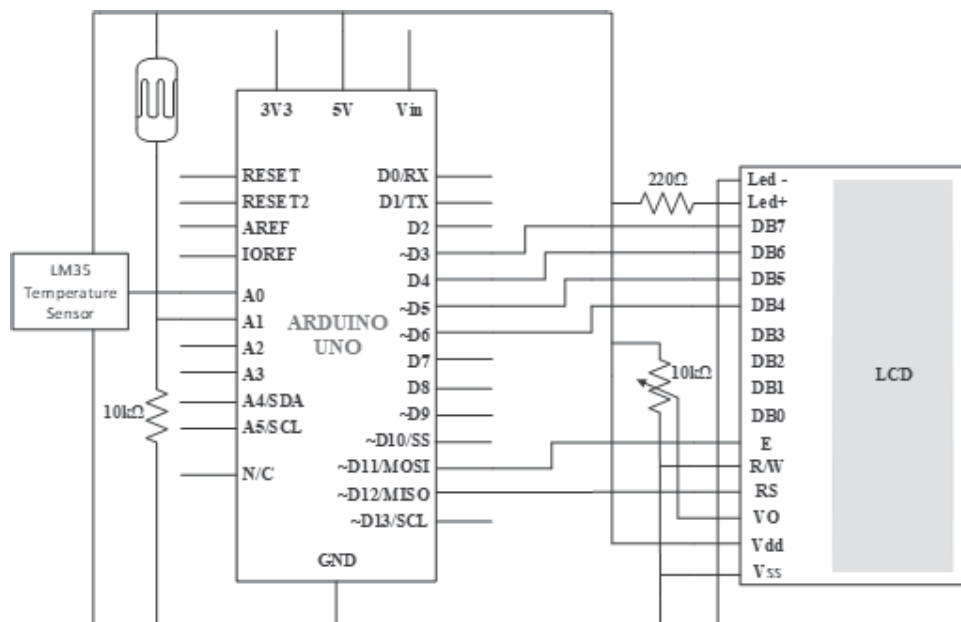
- *Realizovati Arduino sistem koji meri temperaturu i osvetljenost u nekoj prostoriji i rezultate merenja ispisuje na LCD displej. Temperatura treba da bude prikazana u stepenima Celzijusove skale na sredini prvog reda LCD displeja dok osvetljenost treba da bude prikazana bročano u opsegu od 0 do 255 na sredini drugog reda LCD displeja. Potrebno je napisati kod za Arduino, a zatim praktično potvrditi uspešnost realizacije.*

Rešenje:

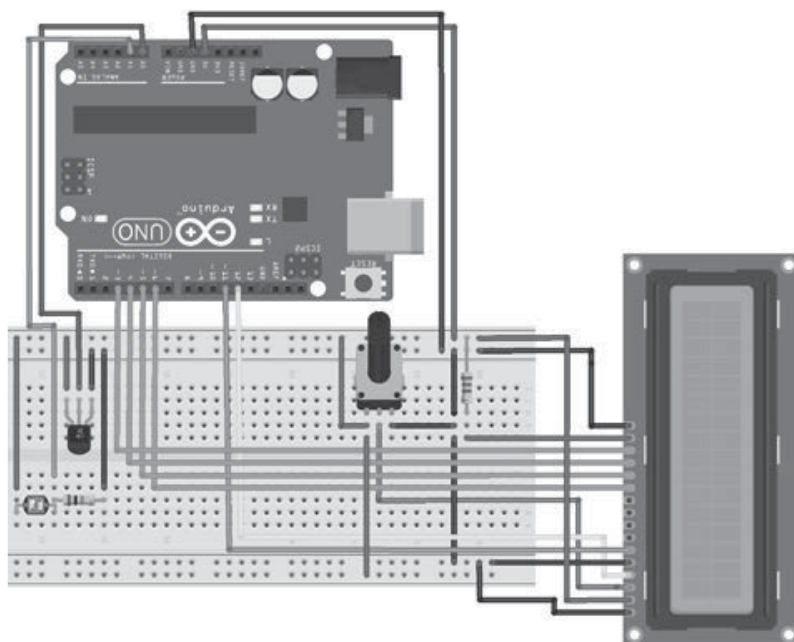
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Otpornik od 220Ω i 10kΩ
- Protobord
- 24 MM kratkospajача
- Fotootpronik
- Temperaturni senzor LM35
- Potenciometar od 10kΩ
- LCD displej

Električna šema koja ilustruje način povezivanja komponenti kao i fizička realizacija sistema u Arduino okruženju prikazani su na slikama 9 - 18 i 9 - 19 respektivno.



Slika 9 - 18 - Električna šema sistema iz primera 9.6



Slika 9 - 19 - Hardverska realizacija sistema iz primera 9.6

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//Included library for working with LCD
#include <LiquidCrystal.h>
#include <TimerOne.h>
#define TempSensorPinNo A1
#define LightSensorPinNo A0
//Timer variable
const unsigned long TimerPeriod = 10000; //micro seconds
//Display size
const int DispHeight = 16;
const int DispWidth = 2;
//Temperature variable
volatile unsigned int CurrentTemp;
volatile unsigned int SumTemp;
volatile unsigned int Temp;
volatile unsigned int CurrentLight;
int counter =0;
const int SamplesNo = 100;
//System variable
typedef enum {DummyRead, TempRead, LightRead} TypeOfRead;
TypeOfRead CurrentRead;
volatile int DispCount = 0;
volatile boolean DispRefresh = false;
//LCD Pin number
const int RSPinNo = 12;
const int EnablePinNo = 11;
const int D7PinNo = 3;
const int D6PinNo = 4;
const int D5PinNo = 5;
const int D4PinNo = 6;
//Create LiquidCrystal variable and initialize it
LiquidCrystal LCD(RSPinNo, EnablePinNo,
                  D4PinNo, D5PinNo, D6PinNo, D7PinNo);

void setup() {
    //Init display size
    LCD.begin(DispHeight, DispWidth);
    Timer1.initialize(TimerPeriod);
    Timer1.attachInterrupt(TimerISR);
    CurrentRead = DummyRead;
    Temp = 0;
    CurrentLight = 0;
}

void WriteToLCD()
{
    LCD.clear();
    //Set cursor at the midle of first row
    LCD.setCursor(DispHeight/2,0);
    //Write temperature to the display
    LCD.print(Temp);
    //Print degree symbol
```

```

    LCD.print((char)0xB0);
    //Print 'C' char
    LCD.print('C');
    LCD.setCursor(DispHeight/2-1,1);
    LCD.print(CurrentLight);
}
void TimerISR(){
    DispCount++;
    switch(CurrentRead){
    case DummyRead:
        CurrentTemp=(5.0*analogRead(TempSensorPinNo)*100.0)/1024;
        CurrentRead = TempRead;
        break;
    case TempRead:
        CurrentTemp=(5.0*analogRead(TempSensorPinNo)*100.0)/1024;
        SumTemp+=CurrentTemp;
        counter++;
        if(counter == SamplesNo){
            //Count average
            Temp = SumTemp/SamplesNo;
            //Reset counter and temperature
            SumTemp = 0;
            counter = 0;
        }
        CurrentRead = LightRead;
        break;
    case LightRead:
        CurrentLight = analogRead(LightSensorPinNo);
        CurrentLight /= 2;
        CurrentLight = constrain(CurrentLight,0,255);
        CurrentRead = DummyRead;
        break;
    }
    if(DispCount == 50){
        DispCount = 0;
        DispRefresh = true;
    }
}
void loop() {
    if(DispRefresh == true){
        DispRefresh = false;
        WriteToLCD();
    }
}

```

Ovaj primer uvodi mehanizam očitavanja sa dva senzora i uvodi koncept osvežavanja sadržaja LCD displeja pomoću tajmera. Na LCD displeju se prikazuju dve vrednosti: srednja vrednost temperature i osvetljenost prostorije.



Na Arduino UNO hardverskoj platformi postoji problem prilikom očitavanja više analognih pinova. Da bi se ovaj problem izbegao neophodno je ubaciti kašnjenje od 10ms između čitanja, dok se prvo čitanje odbacuje. Potrebno je naglasiti da ovaj problem nije karakterističan pri radu sa mikrokontrolerima, već je povezan sa hardverskom realizacijom Arduino UNO hardverske platforme.

Očitavanje vrednosti sa senzora se vrši unutar prekidne rutine tajmera. Pošto se u istoj prekidnoj rutini očitava više vrednosti, uveden je nabrojivi tip podataka *TypeOfRead* čiji elementi su moguće vrednosti koje se očitavaju sa senzora. Promenljiva ovog tipa, koja nosi informaciju o tome koje se trenutno očitavanje vrši je *CurrentRead*.

DummyRead predstavlja prvo očitavanje senzora koje je nevalidno i koje se odbacuje. *TempRead* predstavlja drugo, validno, očitavanje temperature sa senzora. Da bi se dobio stabilniji prikaz vrednosti temperature, sprovodi se mehanizam usrednjavanja koji je objašnjen u primeru 8.4. *LightRead* predstavlja očitavanje vrednosti sa fotootpornika.

U istoj prekidnoj rutini se vrši uvećanje brojača *DispCount*, koji se koristi za implementaciju periode od 500ms. Naime, svaki put kada se uđe u prekidnu rutinu tajmera, koja se poziva svakih 10ms, vrednost ove promenljive se uvećava sve do vrednosti od 50. Kada ova promenljiva dostigne vrednost 50, to je znak da je istekla perioda od 500ms, promenljivoj *DispRefresh* se dodeljuje vrednost *true*. Vrednost ove promenljive se proverava u glavnoj programskoj petlji i po potrebi se poziva funkcija *WriteToLCD* za osvežavanje displeja.

Unutar funkcije *WriteToLCD* se vrši ispis srednje vrednosti temperature *Temp* i trenutne osvetljenosti prostorije. Algoritam za osvežavanje displeja najpre obriše prethodni sadržaj, a zatim pozicionira kursor za ispis temperature nakon čega se ispiše vrednost izmerene temperature. Nakon toga sledi ispis specijalnog simbola za stepen „°“, a potom se ispisuje simbol Celzijusove skale (slovo „C“). Nakon ispisa vrednosti temperature, ispisuje se vrednost osvetljenosti u opsegu od 0-255.

Da bi na LCD displeju ispisali simbole koji nisu brojevi i slova već, predstavljaju specijalne karaktere kao što su „Ω“ ili „°“, neophodno je poznavati implementiranu tabelu simbola koja se nalazi u memoriji kontrolera LCD displeja. Sadržaj ove tabele prikazan je na slici 9 - 20.

Upper 4 bit Lower 4 bit	LLLL	LLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)	+														
LLH	CG RAM (2)	!	1	A	B	a	9	0	a	i			J	T	Y	U
LLHL	CG RAM (3)	7	"	2	B	R	b	r	e	B	a	*	o	e	e	z
LHLL	CG RAM (4)	7	#	3	C	S	c	s	a	b	a	c	P	T	e	4
LHLH	CG RAM (5)	7	\$	4	D	T	d	t	a	b	a	c	'	e	r	z
LHHL	CG RAM (6)	7	%	5	E	U	e	u	b	a	b	a	'	a	n	w
LHHH	CG RAM (7)	7	&	6	F	V	f	v	a	b	a	c	'	u	j	e
LHHH	CG RAM (8)	7	'	7	G	W	w	g	O	R	x	3	A	L	W	
HLLL	CG RAM (1)	7	<	E	H	X	h	x	e	g	5	+	e	e	k	H
HLH	CG RAM (2)	7	>	9	I	Y	i	y	a	b	a	c	'	T	T	x
HLHL	CG RAM (3)	7	*	#	J	Z	j	z	e	O	R	z	7	e	u	F
HLHH	CG RAM (4)	7	+	#	K	C	k	c	i	R	a	&	L	T	v	a
HHLL	CG RAM (5)	7	*	<	L	/	l	/	i	N	o	>	J	e	z	O
HHLH	CG RAM (6)	7	~	~	M	J	m	j	i	a	b	a	c	'	u	u
HHHL	CG RAM (7)	7	*	>	N	n	n	n	a	b	a	c	'	J	e	p
HHHH	CG RAM (8)	7	*	/	7	O	o	o	a	b	a	c	'	J	e	u

Slika 9 - 20- Izgled tabele specijalnih simbola LCD kontrolera

Prvi horizontalni red odozgo predstavlja viših 4 bita osmobitne adrese dok prvi vertikalni red sa leve strane predstavlja nižih 4 bita osmobitne adrese na kojoj se nalazi dati simbol. Slovo H u ovim redovima označava „1“ dok slovo L označava „0“. Da bi ispisali neki simbol iz tabele simbola sa slike 9 - 20 neophodno je kao argument funkcije print() proslediti njegovu poziciju u tabeli.

Primer 9.7. - Digitalni merač otpornosti

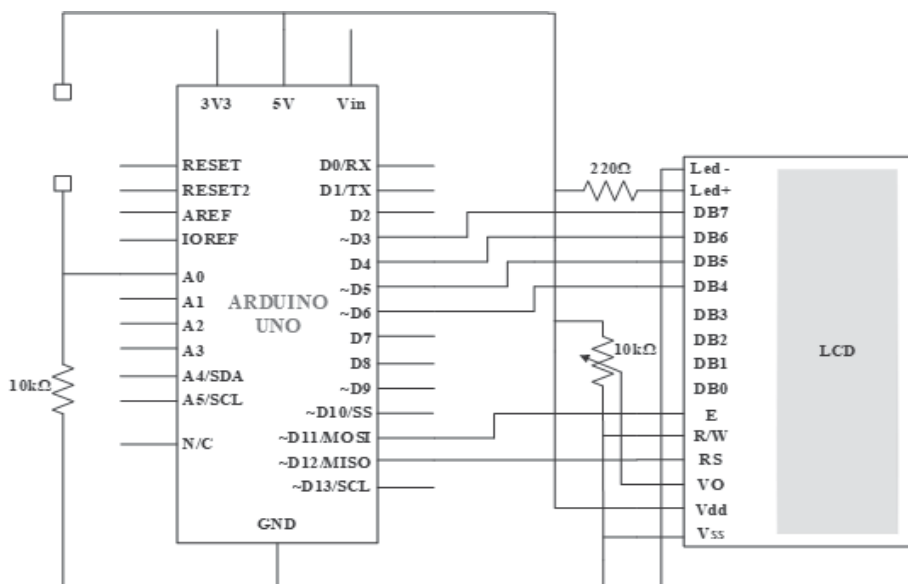
- Realizovati Arduino sistem za merenje otpornosti. Senzor radi na principu razdelnika napona, gde se između analognog pina Arduina i mase povezuje otpornik poznate otpornosti, dok se između istog analognog pina i napona od 5V povezuje otpornik čija se otpornost želi meriti. Vrednost izmerene otpornosti potrebno je ispisati na LCD ekranu. Sistem treba da bude u stanju da detektuje kada je otpornost koja treba da se meri prisutna, a kada se ne nalazi u sistemu i da u skladu sa tim ispisuje korisniku poruku da treba da se postavi željeni otpornik za merenje ili da mu ispisuje vrednost merene otpornosti. Napisati kod za Arduino hardversku platformu koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

Rešenje:

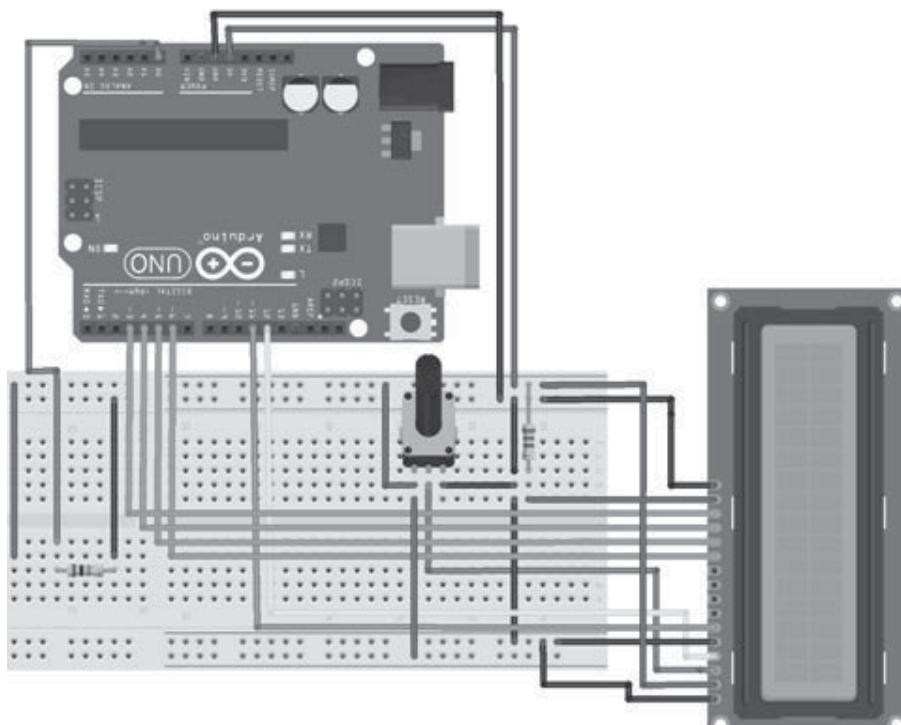
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Otpornici od 220Ω i $10k\Omega$
- Protobord
- 26 MM kratkospajача
- Potenciometar $10k\Omega$
- LCD displej

Električna šema koja ilustruje način povezivanja ovih komponenti prikazana je na slici 9 - 21, dok je fizička realizacija sistema u Arduino okruženju prikazana na slici 9 - 22.

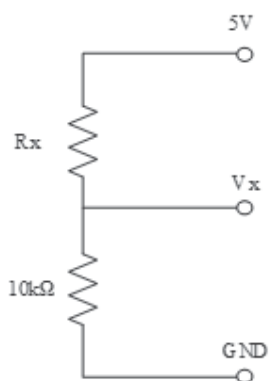


Slika 9 - 21 - Električna šema sistema iz primera 9.7



Slika 9 - 22- Hardverska realizacija sistema iz primera 9.7

U ovom primeru se realizuje razdelnik napona, koji se sastoji od poznate otpornosti ($10k\Omega$) i otpornika R_x čija se otpornost meri. Električna šema ove grane kola prikazana je na slici 9 - 23.



Slika 9 - 23 - Razdelnik napona

Merenje otpornosti u ovom zadatku se bazira na merenju napona u tački V_x . Zato je neophodno naći matematičku formulu koja daje vezu između napona u tački V_x i nepoznate otpornosti R_x . Izvođenje započinje od dobro poznate formule (4) za razdelnik napona. Kada se ta formula primeni na konkretan problem dobija se da je napon u tački V_x jednak:

$$V_x = \frac{5V}{R_x + 10k\Omega} \times 10k\Omega \quad (12)$$

Sada je neophodno iz ove formule, primenom matematičkih jednakosti, izraziti otpornost R_x u zavisnosti od napona V_x . Iz formule (12) se dobija:

$$R_x = 10k\Omega \left(\frac{5V}{V_x} - 1 \right) \quad (13)$$

Formula (13) daje vezu između napona V_x koji se meri i nepoznate otpornosti R_x koja se nalazi u kolu. Takođe iz ove formule može se izvesti još jedan interesantan zaključak, a to je da kada je napon V_x jednak nuli (slučaj kada u kolu nemamo otpornost R_x , jer kroz otpornik od $10k\Omega$ ne protiče struja), formula ne važi jer deljenje sa nulom nije moguće! Ova činjenica će biti korišćena za detekciju situacije u kojoj eksterni otpornik R_x nije povezan u kolo.

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//Included library for working with LCD
#include <LiquidCrystal.h>
#include <TimerOne.h>
#define ResistorSensorPinNo A0
//Display size
const int DispHeight = 16;
const int DispWidth = 2;
//Resistor variable
volatile float ResValue = 0;
volatile float Voltage = 0;
//LCD Pin number
const int RSPinNo = 12;
const int EnablePinNo = 11;
const int D7PinNo = 3;
const int D6PinNo = 4;
const int D5PinNo = 5;
const int D4PinNo = 6;
const unsigned long ReadPeriod = 1000000;
//Create LiquidCrystal variable and initialize it
LiquidCrystal LCD(RSPinNo, EnablePinNo,
                  D4PinNo, D5PinNo, D6PinNo, D7PinNo);
void TimerISR(){
    //Read current temperature
    Voltage = analogRead(ResistorSensorPinNo);
    if(Voltage > 100){
        ResValue=10000*(1024/Voltage-1);
        WriteToLCD();
    }
    else{
        LCD.clear();
        LCD.print("Povezite otpornost");
    }
}
void setup() {
    //Init display size
    LCD.begin(DispHeight, DispWidth);
    //Init timer
    Timer1.initialize(ReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}
```

```
void WriteToLCD()
{
    LCD.clear();
    //Write temperature to the display
    LCD.print(ResValue);
    //Print degree symbol
    LCD.print((char)0xDE);
}
void loop() {
}
```

Najpre se uključuje biblioteka za rad sa LCD displejom *LiquidCrystal.h*. Nakon toga se definiše pin *ResistorSensorPinNo* koji predstavlja broj analognog pina na koji je povezan razdelnik napona (V_x iz šeme sa slike 9 - 21). Promenljiva *ResValue* čuva informaciju o vrednosti otpornosti koja se dobija na osnovu formule (13) dok promenljiva *Voltage* predstavlja digitalni ekvivalent ulaznog napona na analognom pinu.

Nakon inicijalizacije sistema u funkciji *setup* program ulazi u glavnu programsku petlju *loop* unutar koje prvo čita digitalnu vrednost napona. Ukoliko je ta vrednost veća od 100, tada se vrednost otpornosti računa na osnovu formule (13) dok se u suprotnom ispisuje poruka korisniku na LCD displeju da postavi otpornik za merenje.

Primer 9.9. – Štoperica

- *Neophodno je realizovati Arduino sistem koji implementira funkcionalnost štoperice. Štoperica se sastoji od LCD displeja i dva tastera. Pritisak na jedan taster pokreće/zaustavlja štopericu, dok pritisak na drugi taster resetuje štopericu. Štoperica ima mogućnost merenja proteklog vremena do maksimalno 60 minuta. Ukoliko štoperica dođe do kraja vremenskog intervala sama se resetuje i kreće ispočetka. Izmereno vreme se ispisuje na displeju u formatu minuti:sekunde:stotineke. Napisati kod za Arduino hardversku platformu koji implementira opisanu funkcionalnost štoperice i praktično potvrditi uspešnost realizacije. (Posmatrati koliko se razlikuje rezultat merenja štoperice od rezultata merenja sata na računaru na kome pišete kod. Objasniti zašto se izmereno vreme razlikuje).*

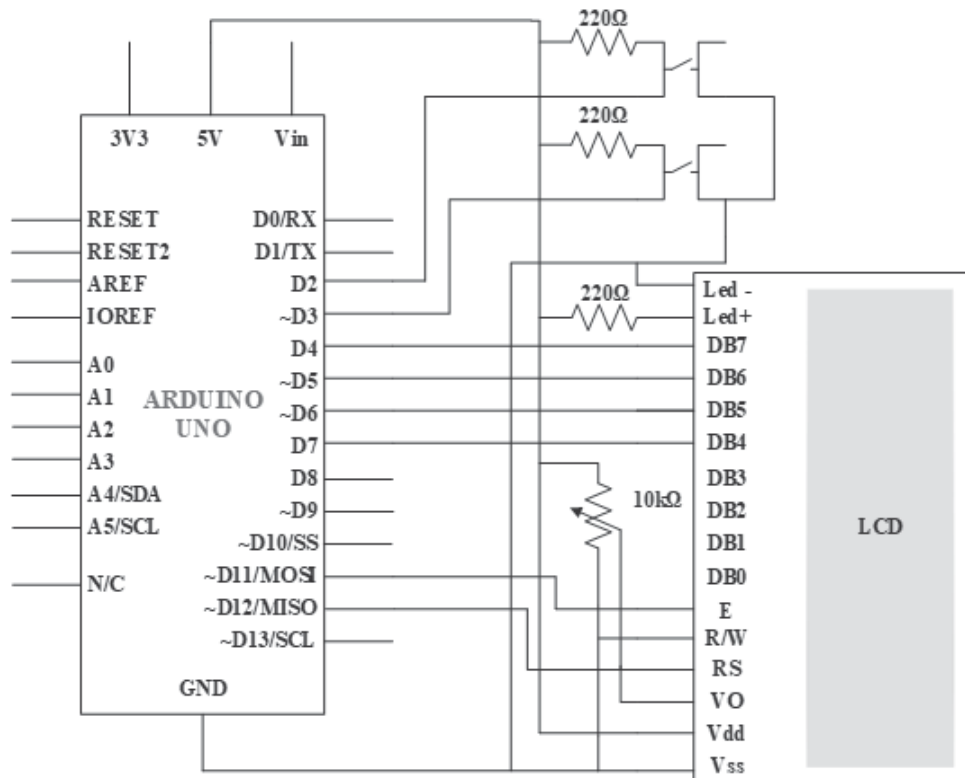
Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

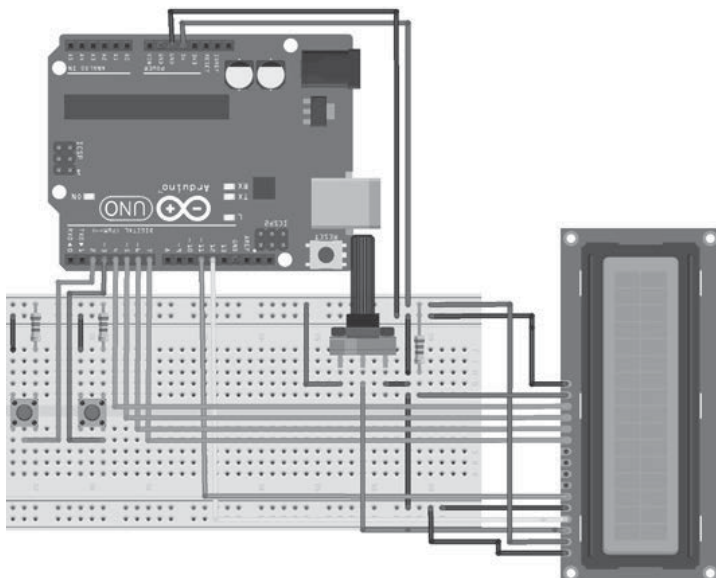
- Arduino UNO
- 3 otpornika od 220Ω
- 2 tastera
- Potenciometar $10k\Omega$

- Protobord
- 21 MM kratkospajач
- Displej

Električna šema koja ilustruje način povezivanja komponenti kao i fizička realizacija sistema u Arduino okruženju su prikazani na slikama 9 - 24 i 9 - 25 respektivno.



Slika 9 - 24 - Električna šema sistema iz primera 9.9



Slika 9 - 25 - Hardverska realizacija sistema iz primera 9.9

U realizaciji rešenja za ovaj zadatak biće korišćen ispis na LCD ekranu u formatu min:sec:stoti gde se na svakih 0.01s broj stotinki uveća za jedan. Kada broj stotinki dođe do 99, resetuje se, a broj sekundi se uveća za jedan, dok kada broj sekundi dođe do 59, njihov broj se resetuje, a broj minuta se uveća za 1. Povećanje broja stotinki vršiće se u tajmerskoj prekidnoj rutini koja se javlja na 0.01s. U ovoj funkciji biće vršeno i osvežavanje sadržaja LCD displeja.

```
#include <LiquidCrystal.h>
#include <TimerOne.h>
const int ButtonStopStartPinNo = 2;
const int ButtonClearPinNo = 3;
//Display size
const int DispHeight = 16;
const int DispWidth = 2;
//LCD Pin number
const int RSPinNo = 12;
const int EnablePinNo = 11;
const int D7PinNo = 4;
const int D6PinNo = 5;
const int D5PinNo = 6;
const int D4PinNo = 7;
LiquidCrystal LCD(RSPinNo, EnablePinNo,
                  D4PinNo, D5PinNo, D6PinNo, D7PinNo);
//When timer is stoped this value is true
//When timer is started this value is false
volatile boolean TimerStop = true;
```

```
//This is stopwatch resolution in microseconds
const unsigned long TimerRes = 10000;
//Time variables
volatile unsigned int Hund = 0; //Hundredths
volatile unsigned int Sec = 0; //seconds
volatile unsigned int Min = 0; //minutes
//Convert miliseconds to min:sec:msec
void Convert_ms(){
    if(Hund > 99){
        //if miliseconds is greater than 1000 increase seconds
        Sec+=1;
        Hund = 0;
        if(Sec > 59){
            //if seconds is greather than 59 increase minutes
            Min+=1;
            Sec = 0;
            if(Min > 59){
                //if stopwatch is reached max value reset to 0
                Min = 0;
            }
        }
    }
}
//This is ISR reset stopwatch and start counting from 0
void ClearISR(){
    Hund = 0;
    Sec = 0;
    Min = 0;
    WriteToDsp();
}
//If stopwatch counting than stop stopwatch otherwise run stopwatch
void StopStartISR(){
    if(TimerStop == true ){
        Timer1.start();
        TimerStop = false;
    }
    else{
        Timer1.stop();
        TimerStop = true;
    }
}
//Write minutes, seconds and milisecond in
//following format min:sec:miliSec
void WriteToDsp(){
    LCD.clear();
    LCD.print(Min);
    LCD.print(':');
    LCD.print(Sec);
    LCD.print(':');
    LCD.print(miliSec);
}
```

```

}
//increase miliseconds value if timer is not stoped and refresh LCD
display
void TimerISR(){
    if(TimerStop == false){
        Hund+=1;
    }
    WriteToDsp();
}
//Initialize system
void setup() {
    //Initialize buttons pins as input pin
    pinMode(ButtonStopStartPinNo, INPUT);
    pinMode(ButtonClearPinNo, INPUT);
    //Attach interrupt to the buttons
    attachInterrupt(digitalPinToInterrupt(ButtonStopStartPinNo),
                    StopStartISR, FALLING);
    attachInterrupt(digitalPinToInterrupt(ButtonClearPinNo), ClearISR, FALLING);
    //Init LCD
    LCD.begin(DispHeight, DispWidth);
    //Set timer stopwatch resolution
    Timer1.initialize(TimerRes);
    //Attach interrupt function to the timer
    Timer1.attachInterrupt(TimerISR);
}
void loop() {
    //Always convert
    Convert_ms();
}

```

Promenljive *ButtonsStartStopPinNo* i *ButtonClearPinNo* označavaju pinove na koje su povezani tasteri za zaustavljanje/startovanje štoperice i resetovanje štoperice, respektivno. Definicija promenljivih koje se odnose na pinove na koje je povezan LCD displej su dobro poznate iz prethodnih primera. Promenljiva *TimerStop* čuva informaciju o tome da li je tajmer zaustavljen ili nije.

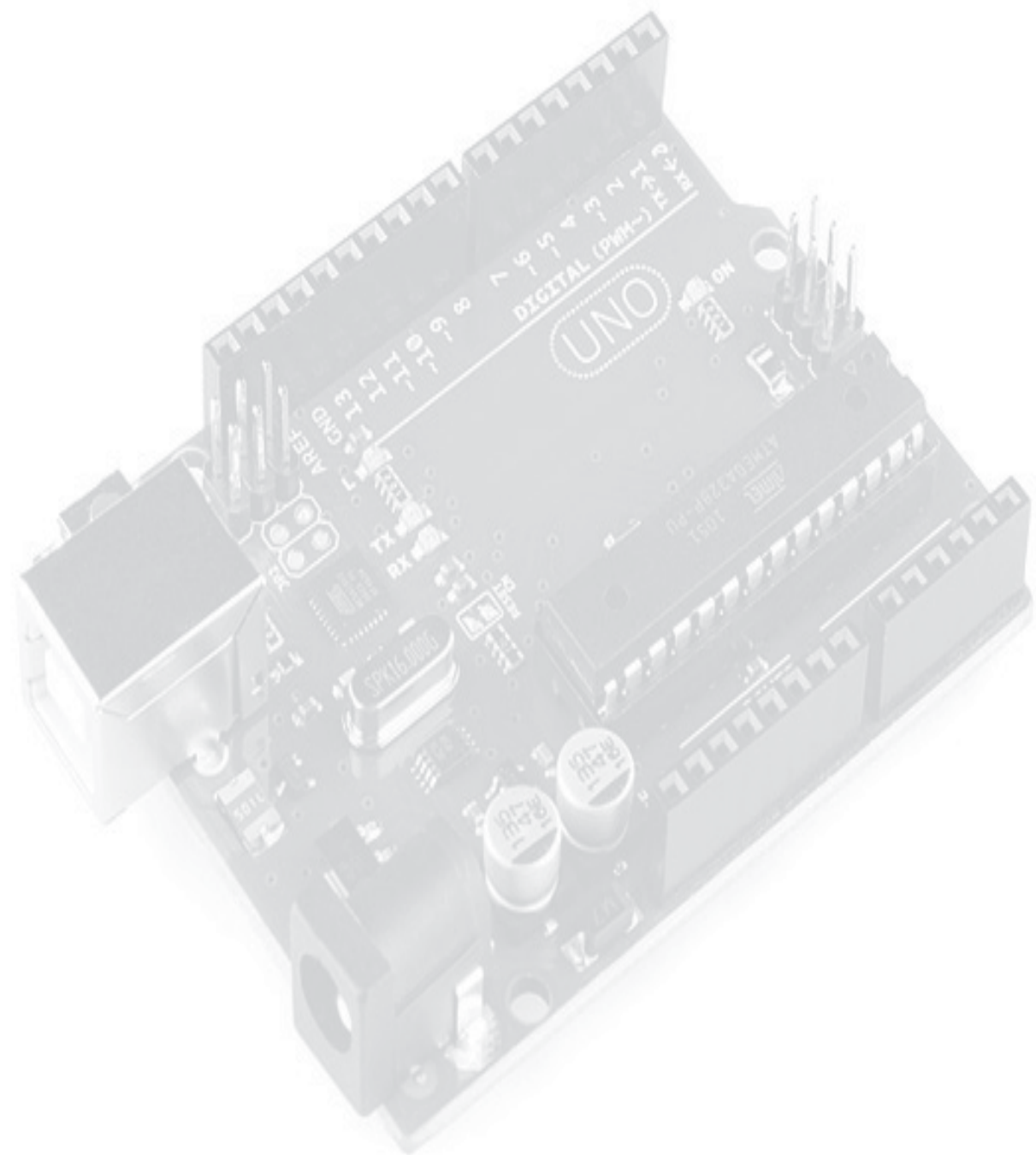
Vrednost ove promenljive se menja unutar prekidne rutine *StartStopISR* dodeljene pinu broj 2 na koji je povezan taster za pokretanje/zaustavljanje štoperice. Promenljiva *TimerRes* predstavlja rezoluciju štoperice, odnosno periodu tajmera.

Promenljive koje se koriste za čuvanje informacija o proteklom vremenu su *Hund*, *Sec* i *Min*. Promenljiva *Hund* čuva informaciju o broju proteklih stotinki, *Sec* čuva informaciju o broju proteklih sekundi, dok *Min* čuva informaciju o broju proteklih minuta. Promenljiva *Hund* se menja u prekidnoj rutini tajmera *TimerISR*, tako što se na prethodnu vrednost doda jedna stotinka, odnosno 10 milisekundi. Vrednost ove promenljive se proverava unutar funkcije *Convert_us* čiji zadatak jeste da konvertuje stotinke u sekunde i minute.

Ispis vremenskih promenljivih (*Hund*, *Sec* i *Min*) se vrši pozivom funkcije *WriteToDsp* koja ispisuje ove promenljive na LCD displej pozivom funkcije iz biblioteke *LiquidCrystal*. Ova funkcija se poziva svaki put kada se uveća vrednost promenljive *Hund* ili kada se asinhrono, pritiskom na drugi taster, resetuje štoperica. Resetovanje štoperice se vrši u funkciji *ClearISR* koja obrađuje prekid na pinu broj 3.

9.5. ZADACI ZA SAMOSTALAN RAD

1. Realizovati arduino sistem koji se sastoji od 2 tastera i LCD displeja. Na LCD displeju se ispisuje broj pritiska na tasteru 1, dok se tasterom 2 vrši resetovanje brojača. Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
2. Realizovati Ardino sistem koji se sastoji od 2 tastera i LCD displeja. Prilikom pokretanja sistema na LCD displeju, na sredini prvog reda, je ispisan karakter „C“. Pritisak na jedan taster treba da izaziva pomeranje karaktera u levo, dok pritisak drugog tastera treba da izaziva pomeranje karaktera u desno. Kada karakter dođe na kraj displeja, a taster je i dalje pritisnut, karakter ne treba da se pomera. Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
3. Realizovati Arduino sistem koji se sastoji od potencijometra i LCD displeja. Na LCD displeju se prikazuje vrednost od 0 do 10kΩ u zavisnosti od položaja potencijometra. Vrednost otpornosti se ispisuje na sredini LCD displeja. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.
4. Realizovati Arduino sistem koji se sastoji od Arduino hardverske platforme i LCD displeja. Na LCD displeju je neophodno naizmenično menjati sadržaj. Potrebno je ispravno povezati Arduino, LCD ekran i prateće komponente, napisati i testirati kod koji naizmenično vrši ispisivanje Vašeg imena, prezimena i reči Internet Stavri na LCD ekranu. Potrebno je da jedan sadržaj bude vidljiv 5 sekundi posle čega treba ispisati drugi sadržaj.
5. Realizovati Arduino sistem za merenje temperature koji se sastoji od Arduino hardverske platforme, senzora temperature i LCD displeja. Temperatura se meri pomoću senzora temperature LM35. Vrednost temperature se prikazuje na LCD displeju. Potrebno je ispravno povezati Arduino, LCD displej i senzor. Napisati i testirati kod koji omogućava čitanje vrednosti sa senzora i ispis vrednosti temperature na LCD displej. Temperatura se ispisuje na sredini LCD displeja. Očitana vrednost treba da varira u što manjim granicama. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.



10. SERIJSKA KOMUNIKACIJA I MODULI

U svetu elektronike komunikacija predstavlja veoma važan proces. Koristi se radi prenosa digitalnih podataka između dva uređaja. Podaci se prenosi preko magistrala (skup žica kroz koje protiče struja). U zavisnosti od načina na koji se podaci prenose postoje dva tipa komunikacije:

- paralelna
- serijska

Paralelna komunikacija služi za prenos više digitalnih podataka (bitova) istovremeno. Za prenos podataka se koristi veći broj kanala (žica) i to su najčešće 8, 16 ili 32 kanala. Omogućava veoma brz prenos podataka ali broj resursa (žica), koji su neophodni za ovaj tip komunikacije, je veoma veliki. Takođe ovaj vid komunikacije zahteva posebne mehanizme za sinhronizovani prijem paralelno poslatih bita. Najčešće se koristi unutar integrisanih kola (čipova) i sreće se u raznim tipovima RAM memorije i nije toliko zastupljenja kao tip komunikacije među uređajima.

Serijska komunikacija omogućava serijski (bit za bitom) prenos podataka. Brzine prenosa ovom magistralom su znatno manje od brzine prenosa podataka putem paralelne magistrale. Međutim, broj resursa (žica) koja koje se koriste tokom ove komunikacije je znatno manji (od četiri pa sve do jedne žice) i nema potrebe za posebnim mehanizmima za detekciju paralelno poslatih bita. Serijska komunikacija se dosta koristi za komunikaciju među uređajima ili za povezivanje senzora sa platformama kao što je Arduino.

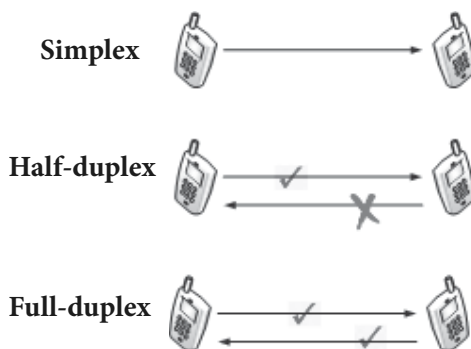
Da bi uređaji koji se povezuju mogli međusobno da se sinhronizuju (dogovore kada ko šalje, odnosno prima podatak) često jedna od žica na magistrali predstavlja takt. Takve magistrale se nazivaju *sinhronim serijskim magistralama*. Međutim postoje i serijske magistrale koje ne prenose takt, već uređaji koji komuniciraju u sebi imaju integrisane uređaje za sinhronizaciju. Takve magistrale se nazivaju *asinhronne serijske magistrale*. U zavisnosti od toga da li uređaji mogu istovremeno da komuniciraju, tj. da istovremeno šalju podatke, ili ne, postoje tri tipa komunikacije [13]:

- half-duplex
- full-duplex
- simplex

Simplex komunikacija se odvija u jednom smeru. **Half-duplex** komunikacija se odvija u oba smera, ali ne istovremeno. Drugim rečima, dok jedan uređaj “priča”, drugi “sluša”. **Full-duplex** komunikacija odvija se istovremeno u oba smera. Ovi principi komunikacije ilustrovani su na slici 10 - 1.

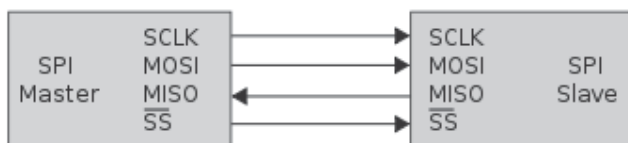
Neki od najčešćih tipova serijskih magistrala su:

- SPI
- I2C
- UART



Slika 10 - 1 - Principi rada različitih vrsta serijske komunikacije

SPI (Serial Peripheral Interface) je serijska sinhrona full-duplex magistrala koju najčešće čine četiri linije [14]. Dizajnirana je u cilju komunikacije uređaja koji nisu puno udaljeni. Interfejs ove magistrale je razvijen od strane Motorole osamdesetih godina i postao je standard.



Slika 10 - 2 - SPI Magistrala

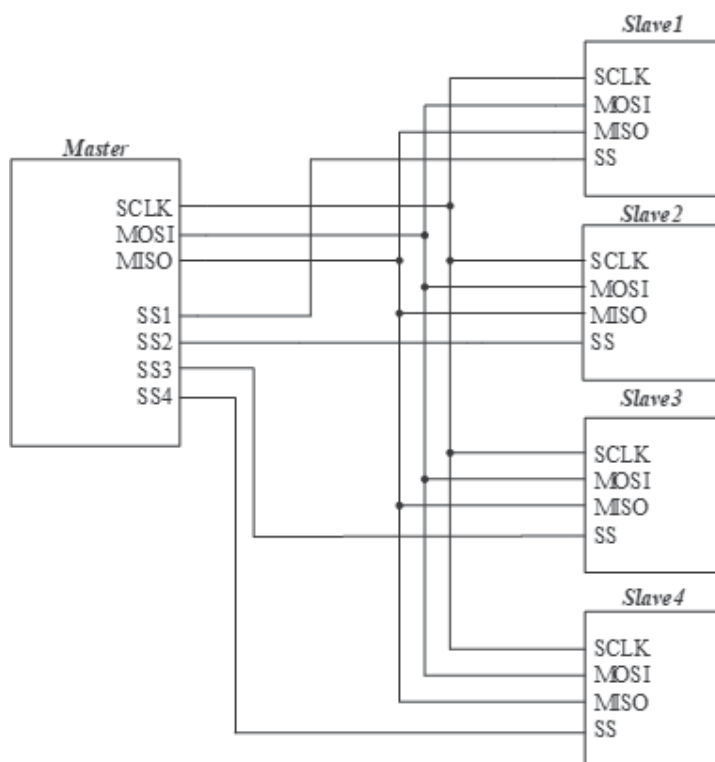
Četiri linije koje čine ovu magistralu su :

- **SCLK (Serial Clock)** – Linija kojom se prenosi takt
- **MOSI (Master Output Slave Input)** – Linija kojom se prenose podaci od mastera ka slejvu
- **MISO (Master In Slave Output)** – Linija kojom se prenose podaci od slejva ka masteru
- **SS (Slave Select)** – Služi za selekciju slejva u sistemu sa više slejv uređaja.

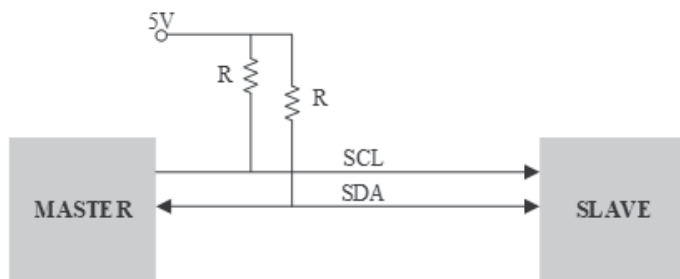
Master je uređaj na magistrali koji upravlja procesom prenosa. On generiše takt i započinje prenos. Slejv predstavlja uređaj koji prima naredbe od mastera i odgovara masteru u skladu sa primljenim naredbama. Na ovoj magistrali može biti samo jedan master dok broj slejvova nije ograničen. U slučaju kada na magistrali postoji više slejv uređaja, master uz pomoć Slave Select linije signalizira sa kojim od njih u datom trenutku komunicira. Master u ovom slučaju ima onoliko slave select pinova, koliko ima slejv uređaja na mreži. Način povezivanja više slejvova korišćenjem ove magistrale ilustroan je na slici 10 - 3.

I2C (Inter-Integrated Circuit) predstavlja sinhronu serijsku half-duplex magistalu koju čine dve linije [15]. Originalno je razvijena 1982. godine od strane *Philips Semiconductor* u cilju povezivanja sporijih periferija sa mikrokontrolerima ili procesorima koji nisu puno udaljeni. Linije koje čine ovu magistralu su:

- **SCL (Serial Clock)** – Prenosi takt
- **SDA (Serial Data)** – Prenosi podatke



Slika 10 - 3 - Multi- slave sistem na realizovan korišćenjem SPI magistrale



Slika 10 - 4 - I2C Magistrala

I SCL i SDA linije imaju nedefinisana stanja i zbog toga se u praksi obe linije pull-up otpornicima vezuju na visok naponski nivo, čime im se definiše naponski nivo u neaktivnm stanju. Pravilno povezivanje ove magistrale ilustrirano je na slici 10 - 4. Ova magistrala dozvoljava postojanje više mastera i više slejvova na jednoj magistrali.

UART (Universal asynchronous receiver-transmitter) predstavlja asinhronu serijsku magistralu. Postoji full-duplex, half-duplex i simplex varijanta ove magistrale. U ovom priručniku razmatramo full-duplex magistrala koja se sastoji od dve linije:

- **TX (Transmit line)** – linija po kojoj se šalju podaci
- **RX (Receive line)** – linija po kojoj se primaju podaci

TX izlaz jednog uređaja se povezuje na RX pin drugog uređaja i obratno. Pravilno povezivanje uređaja, koji komuniciraju putem UARTa, prikazano je na slici 10 - 5.



Slika 10 - 5 - UART komunikacija

Arduino UNO platforma podržava serijsku komunikaciju.

Kao TX i RX pinovi UART komunikacije se koriste digitalni pinovi 1 i 0

Kao SCL i SDA pinovi I2C komunikacije se koriste analogni pinovi A5 i A4

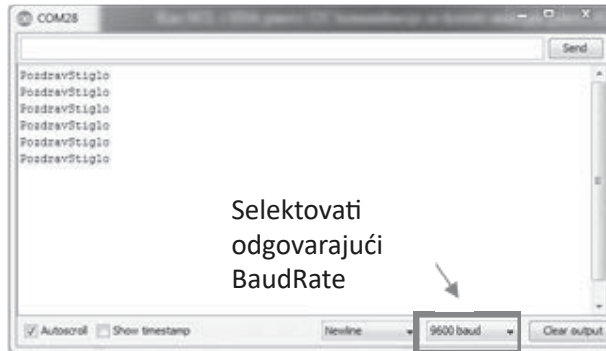
Kao SCK, MISO, MOSI i SS pinovi SPI komunikacije se koriste digitalni pinovi 13, 12, 11 i 10 respektivno



Na pinove 0 i 1, koji se koriste kao UART interfejsi, interno je povezan USB konektor što znači da su ovi pinovi deljeni. To zahteva da ukoliko instaliramo program na Arduino platformu, koja koristi UART komunikaciju, neophodno je pre toga izvaditi kratkospajalice iz pinova 0 i 1, pa tek nakon uspešne instalacije vratiti kratkospajalice.

Arduino IDE pruža mogućnost praćenja stanja UART serijskog intefejsa, odnosno prikaz primljenih podataka na tom interfejsu. Ta opcija se naziva *Serial Monitor* i u Arduino softverskom okruženju pokreće se u meniju *tools* a slekcijom opcije *Serial Monitor*. Nakon pokretanja ove opcije dobijamo prozor prikazan na slici 10 - 6.

Nakon pokretanja ovog pomoćnog alata neophodno je u donjem desnom uglu , kao što je prikazano na slici 10 - 6, odabrati brzinu prenosa (tzv. *BaudRate*) kojom se vrši prenos podataka. Detalji vezani za inicijalizaciju brzine prenosa biće ilustrirani primerima koji slede.

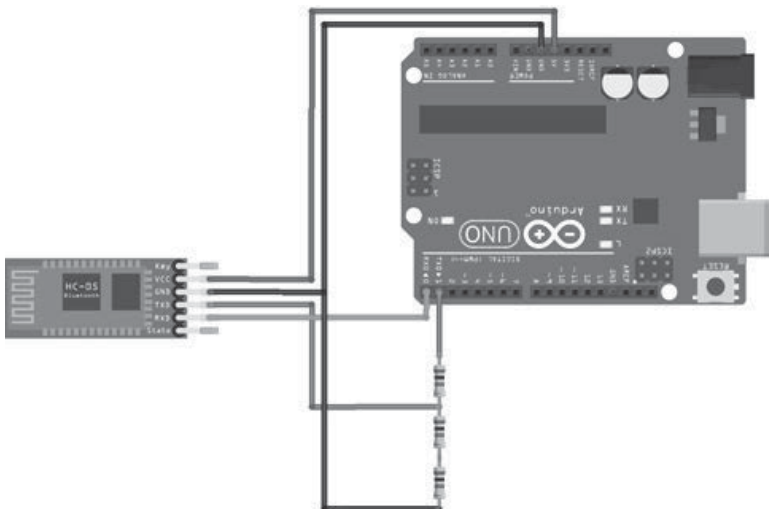


Slika 10 - 6 - Arduino IDE Serial monitor

10.1. BLUETOOTH MODUL

Bluetooth modul koji se razmatra u ovom priručniku je modul koji sadrži HC-06 [9] bluetooth kontroler koji se sa Arduino platformom povezuje pomoću UART interfejsa. *Baudrate* kojim se odvija komunikacija između Arduina i modula iznosi 9600.

Bluetooth modul ima logičke nivoe na 3.3V i 0V, dok Arduino ima logičke nivoe na naponima od 5V i 0V. Da bi TX linija kojom se šalju podaci od Arduina ka Bluetooth modulu bila kompatibilna sa RX linijom na Bluetooth modulu neophodno je spustiti nivo od 5V na nivo od 3.3V što se može postići razdelnikom napona kao što je prikazano na slici 10 - 7.



Slika 10 - 7 - Način povezivanja Bluetooth modula i Arduina

Za komunikaciju mobilnog telefona sa *bluetooth* modulom, u ovom priručniku koristićemo terminal za *Bluetooth* aplikaciju koja se može besplatno skinuti sa interneta [18]. Nakon instalacije ove aplikacije, komunikacija između mobilnog telefona i *Bluetooth* modula može se uspostaviti sprovođenjem sledećih koraka:

1. Uključiti *Bluetooth* komunikaciju na telefonu, pronaći *Bluetooth* modul pod nazivom HC-06 i poslati zahtev za uparivanjem. (Slika 10 -8).
2. Uneti lozinku za uparivanje koja je za ovaj uređaj (najčešće 1234). (Slika 10 - 9).
3. Bluetooth modul je uparen sa telefonom o čemu korisnik dobija odgovarajuće obaveštenje. (Slika 10 - 10).
4. U aplikaciji je neophodno ostvariti povezivanje sa uparenim *Bluetooth* uređajem. Povezivanje aplikacije sa *Bluetooth* uređajem se postiže tako što se klikne na simbol viljuške u gornjem centralnom delu aplikacije (Slika 10 - 11).
5. Nakon toga se izlistaju uređaju dostupni za povezivanje. Neophodno je izabrati HC-06 (Slika 10 - 12).
6. Ukoliko je povezivanje uspešno, aplikacija se može koristiti za komunikaciju (Slika 10 - 13).
7. Slika 10 - 13 prikazuje izgled ekrana aplikacije kada Arduino, putem *Bluetooth* modula, šalje telefonu pozdravnu poruku.



Slika 10 - 8 - Pretraga dostupnih Bluetooth uređaja



Slika 10 - 9 - Uparivanje sa uređajem HC-06



Slika 10 - 10 - Potvrda uspešnog uparivanja



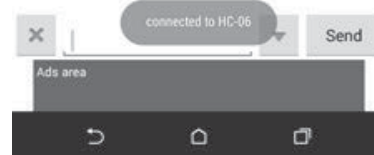
Slika 10 - 11 - Android aplikacija za komunikaciju sa uređajem



Slika 10 - 12 - Odabir uparenog Bluetooth modula iz aplikacije



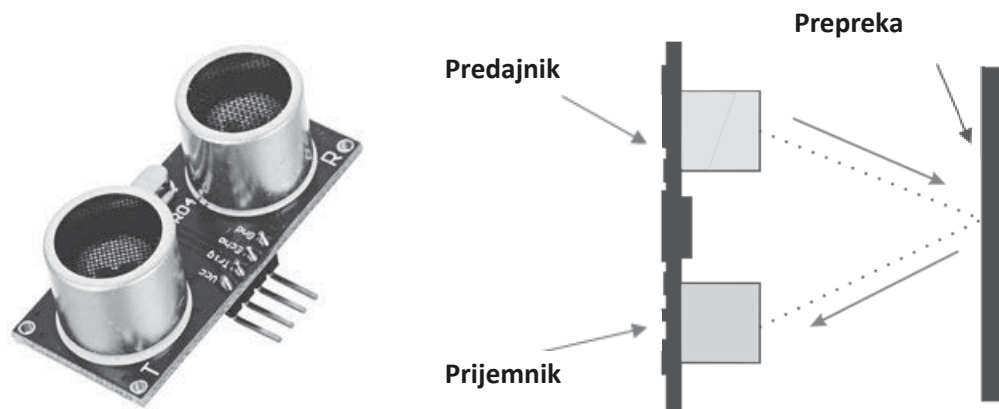
Pozdrav
Pozdrav



Slika 10 - 13 - Potvrda uspešne konekcije

10.2. ULTRAZVUČNI MODUL

Ultrazvučni modul na sebi sadrži predajnik i prijemnik ultrazvuka koji se koriste u cilju merenja razdaljine između senzora i nekog objekta. Radi na principu merenje vremena potrebnog da se emitovani talas odbije od objekta i vrati ka prijemniku. Na osnovu poznate brzine zvuka kroz vazduh moguće je izračunati razdaljinu između senzora i objekta. Izgled ovog modula i njegov princip rada prikazani su na slici 10 - 14.



Slika 10 - 14 – Fizički izgled ultrazvučnog modula (levo) i princip njegovog rada (desno)

Ovaj modul omogućava beskontaktno merenje razdaljine u opsegu od 2 do 400cm. Pored ultrazvučnog predajnika i prijemnika, modul sadrži i kontrolnu elektroniku. Modul ima 4 interfejsna pina:

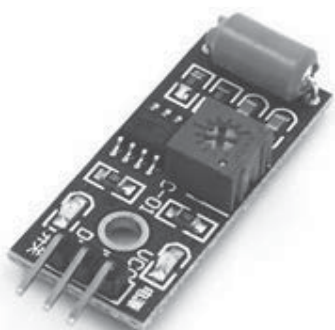
- VCC (Napon napajanja 5V)
- GND (Ground)
- Echo (Koristi se u komunikaciji)
- Triger (Koristi se u komunikaciji)

Najpre se na pin *Triger* dovodi visok naponski nivo u trajanju od 10us, a zatim se očitava trajanje visokog naponskog nivoa na pinu *Echo*. Kada se na pin *Triger* dovede visok naponski nivo, vrši se emitovanje ultrazvučnog signala učestanosti 40kHz. Signal se emituje kroz vazduh brzinom zvuka od 340m/s (0.034cm/us) i dolazi do prepreke. Zvučni signal se zatim odbija od prepreke i vraća do modula. Na osnovu poznate brzine i proteklog vremena od emitovanja sa predajne strane do trenutka pristizanja talasa na prijemnoj strani, možemo izračunati razdaljinu objekta od senzora. Put, koji pređe talas, dobija se iz formule:

$$S = \frac{V_{\text{zvuka}} \cdot \text{vreme}}{2} \quad (14)$$

U formuli ceo izraz se deli sa 2, jer zvuk prvo dođe do objekta pa se zatim vrati, tako da je pređeni put dva puta veći od razdaljine između senzora i objekta koja se meri.

10.3. MODUL ZA DETEKCIJU VIBRACIJA

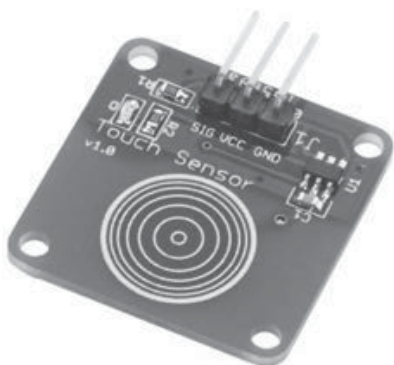


Slika 10 - 15 - Modul za detekciju vibracija

Ovaj modul sadrži senzor za detekciju vibracija, koji radi na principu pokretnog prekidača koji uspostavlja i prekida električno kolo usled mehaničkih pokreta.

Na pločici modula postoje tri pina od kojih su dva za napajanje (VCC-5V i GND) dok se treći pin koristi za detekciju vibracija. Ukoliko je na ovom pinu stanje logičke jedinice, to je znak da postoje vibracije dok u suprotnom vibracije ne postoje. Kolika je jačina vibracija može se detektovati na osnovu vremena trajanja stanja logičke jedinice. Ovaj modul prikazan je na slici 10 - 15.

10.4. MODUL OSETLJIV NA DODIR

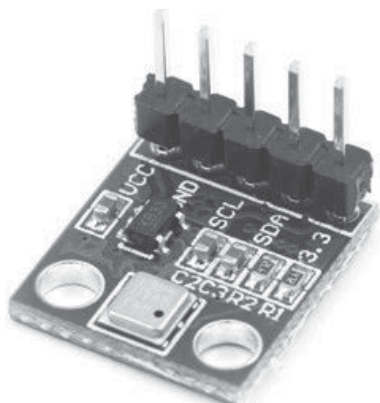


Slika 10 - 16 - Modul sa senzorom osetljivim na dodir

Modul osetljiv na dodir sadrži kapacitivni senzor koji menja kapacitivnost pri dodiru. Nema mehaničkih delova i kao takav je veoma pogodan za korišćenje. Kapacitivni senzor, korišćen u ovom priručniku, prikazan je na slici 10 - 16.

Modul na kome se nalazi senzor ima 3 pina od kojih su dva za napajanje (GND i VCC-5V), dok treći pin predstavlja interfejs ka senzoru. Kada je kapacitivni senzor aktivan, tj kada je objekat blizu senzora, na ovom pinu (SIG) se javlja logička jedinica, dok je u suprotnom na pinu logička nula.

10.5. DIGITALNI BAROMETAR



Slika 10 - 17 - Modul za merenje vazdušnog pritiska

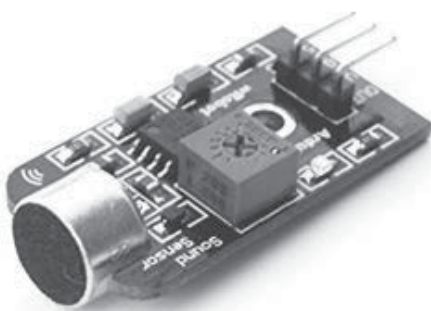
Ovaj modul služi za merenje vazdušnog pritiska. U ovom priručniku se koristi modul koji na sebi ima senzor BMP180. Ovaj modul prikazan je na slici 10 - 17.

Modul ima 5 pinova. Za povezivanje sa Arduinoom se koriste četiri od kojih dva za napajanje (GND i VCC-5V). Modul sa Arduinoom komunicira putem I2C magistrale, pri čemu SDA i SCL linije treba povezati na A4 i A5 respektivno.

Ovaj senzor pored očitavanja pritiska nudi i očitavanje temperature. Na osnovu trenutne vrednosti vazdušnog pritiska takođe je moguće izračunavanje nadmorske visine.

Za korišćenje ovog modula postoji gotova biblioteka koja se može preuzeti sa interneta [19]. Nakon preuzimanja biblioteke neophodno je izvršiti njenu instalaciju i uključivanje u projekat. Pre uključivanja preuzete biblioteke neophodno je uključiti biblioteku *Wire* koja omogućava rad sa uređajem putem I2C magistrale. Ova biblioteka je već sadržana u Arduino IDE softverskom orkuženju, pa je neophodno samo uključiti heder fajl *Wire.h*

10.6. MODUL ZA DETEKCIJU ZVUKA



Slika 10 - 18 - Modul za detekciju zvuka

Ovaj modul sadrži senzor za detekciju zvuka i prikazan je na slici 10 - 18.

Na pločici ovog modula postoje 3 pina od kojih su dva za napajanje GND i VCC-5V, dok pin OUT predstavlja izlazni pin senzora. Kada nije detektovan zvuk na ovom pinu se očitava napon od 5V, tj. logička jedinica, dok je u prisutvu zvuka napon na izlazu ovog pina 0V. Potenciometrom, koji se nalazi u plavom kućištu, moguće je podešavati osetljivost mikrofona.

10.7. PRIMERI

Primeri koji slede imaju za cilj da čitaoca upoznaju sa osnovnim principima rada uređaja koji imaju mogućnost serijske komunikacije.

Primer 10.1. - Serijska veza dva Arduina

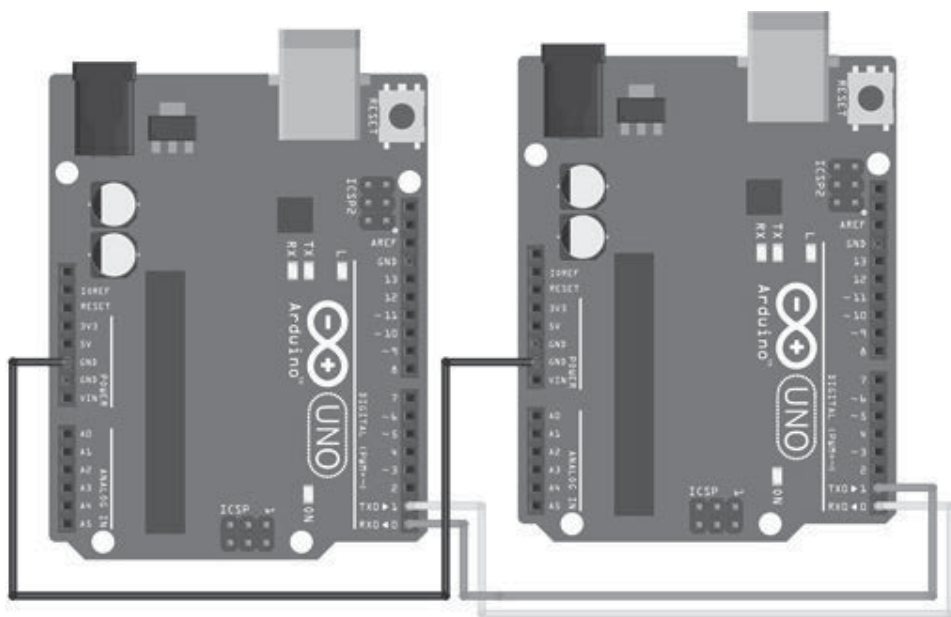
- *Potrebno je realizovati sistem koji se sastoji od dva Arduino hardverska modula koji su povezani putem UART serijske magistrale. Arduino, koji predstavlja master uređaj na magistrali, šalje poruku "Pozdrav" svake 2 sekunde. Tu poruku prima Arduino koji predstavlja slejv uređaj. Nakon prijema, slejv treba da detektuje da li je primljena poruka identički jednaka reči "Pozdrav" i u ukoliko jeste, treba putem UART magistrale da odgovori master Arduino sa "Pozdrav+ACK".*

Rešenje:

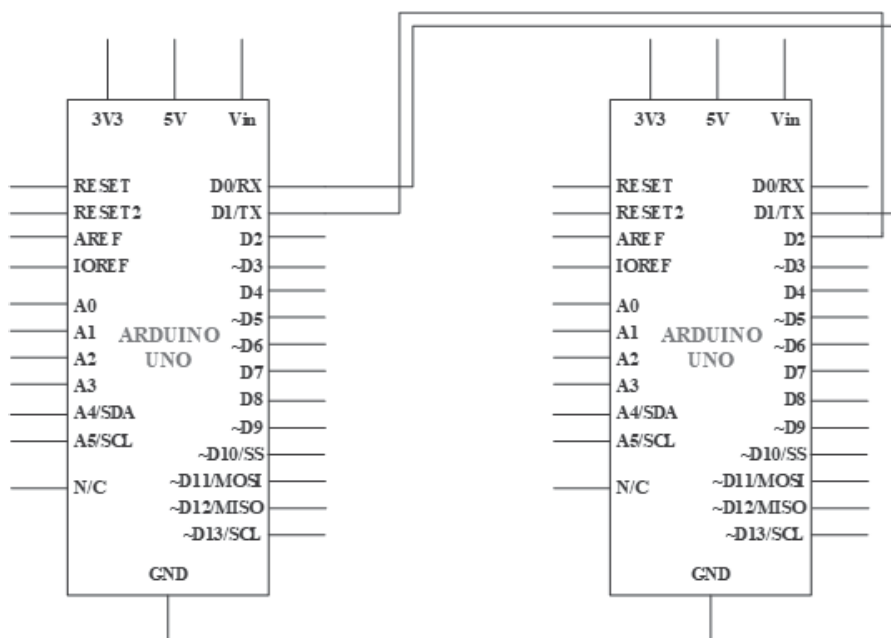
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Dva Arduino
- 3 MM kratkospajča

Fizička realizacija sistema i električna šema povezivanja prikazani su na slikama 10 - 19 i 10 - 20.



Slika 10 - 19 - Hardverska realizacija sistema iz primera 10.1



Slika 10 - 20 - Električna šema sistema iz primera 10.1

Prvo se vrši povezivanje Master Arduino platforme i PC-a, a zatim se instalira kod koji ima zadatak da inicijalizuje prenos na brzini 9600 i da zatim pošalje zahtevanu poruku.

```
#include <TimerOne.h>
const long UARTSpeed = 9600;
const unsigned long MsgPeriod= 2000000;

void SendMsg(){
    Serial.write("Pozdrav");//Send string
}

void setup() {
    Serial.begin(UARTSpeed); //Init communication
    Timer1.initialize(MsgPeriod);
    Timer1.attachInterrupt(SendMsg);
}

void loop() {
}
```

Nakon instalacije koda na master Arduino, neophodno je instalirati kod na slave Arduino tako što se sada povezuje slave Arduino i PC. Nakon instalacije programa na slave uređaj, putem kratkospajanja je neophodno povezati dva Arduina kao što je prikazano na slici 10 - 19.

```
const long UARTSpeed = 9600;
String ReceivedString; // Recieved mesage is stored here
void setup() {
    Serial.begin(UARTSpeed); //Init communication
}
void loop() {
    //Is data byte received?
    if(Serial.available() > 0)
    {
        //Read string
        ReceivedString = Serial.readString();
        //Check equality
        if(ReceivedString == "Pozdrav"){
            // Print Ack
            Serial.println(ReceivedString+"-Ack");
        }
    }
}
```

Da bi se uspešno inicijalizovala UART komunikacija, najpre je neophodno podesiti brzinu komunikacije, što se postiže pozivom funkcije:

Serial.begin(SerialSpeed)

SerialSpeed – Brzina na kojoj se odvija komunikacija. Postoji skup mogućih brzina komunikacije, ali najčešće korišćene su 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 i 115200.

Slanje podataka u vidu niza karaktera (stringa) putem UART interfejsa se vrši pozivom funkcije

Serial.write(msg)

msg – String koji se šalje

Provera da li je primljen neki karakter na prijemnoj strani se vrši pozivom funkcije

Serial.available()

koja vraća pozitivan broj ukoliko je primljen podatak, dok u suprotnom vraća negativan broj.

Čitanje primljenog stringa se vrši pozivom funkcije

Receivedmsg = Serial.readString()

Receivedmsg – Prihvatni string u koji smeštamo primljenu poruku.

Ispis stringa na Serijski monitor se vrši pozivom funkcije

Serial.println(msg)

msg – string koji treba ispisati na Serijsku konzolu (monitor)

Zadatak prikazanog koda za slejv uređaj jeste da inicijalizuje UART komunikaciju na određenoj brzini. U glavnom programu se čeka na prijem stringa. Nakon toga se vrši provera da li je primljeni string jednak očekivanom, i ukoliko jeste, vrši se modifikacija primljenog stringa i on se šalje nazad. String će takođe biti ispisan i na serijsku konzolu.

Primer 10.2. - Kontrola diode putem serijske komunikacije

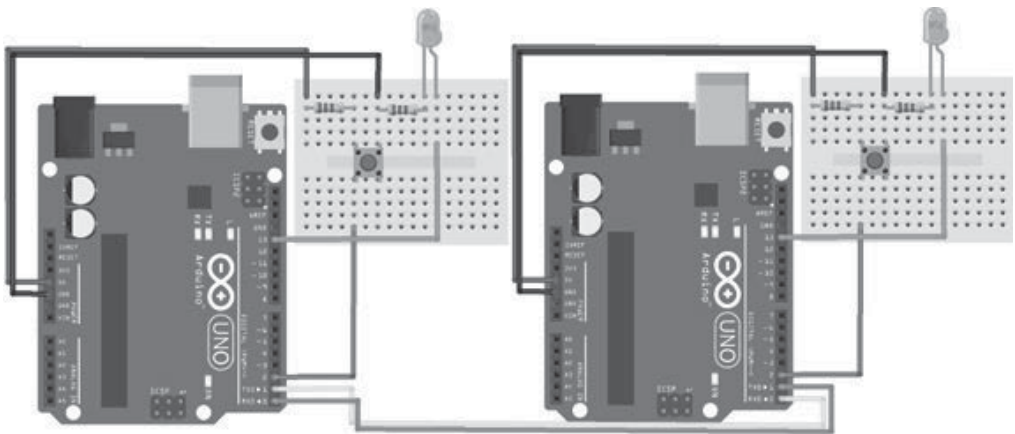
- *Realizovati sistem koji se sastoji od dve Arduino hardverske platforme koje serijski komuniciraju. Za svaki od dva Arduina su povezani LE dioda i taster. Potrebno je uspostaviti serijsku komunikaciju između dva Arduino mikrokontrolera i implementirati program kojim će se dioda povezana za jedan Arduino uključivati, odnosno isključivati pritiskom na taster koji je povezan na drugi Arduino kontroler. S obzirom da su komponente povezane na isti način na oba Arduin-a, potrebno je implementirati jedinstven programski kod koji će funkcionisati na oba kontrolera. Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.*

Rešenje:

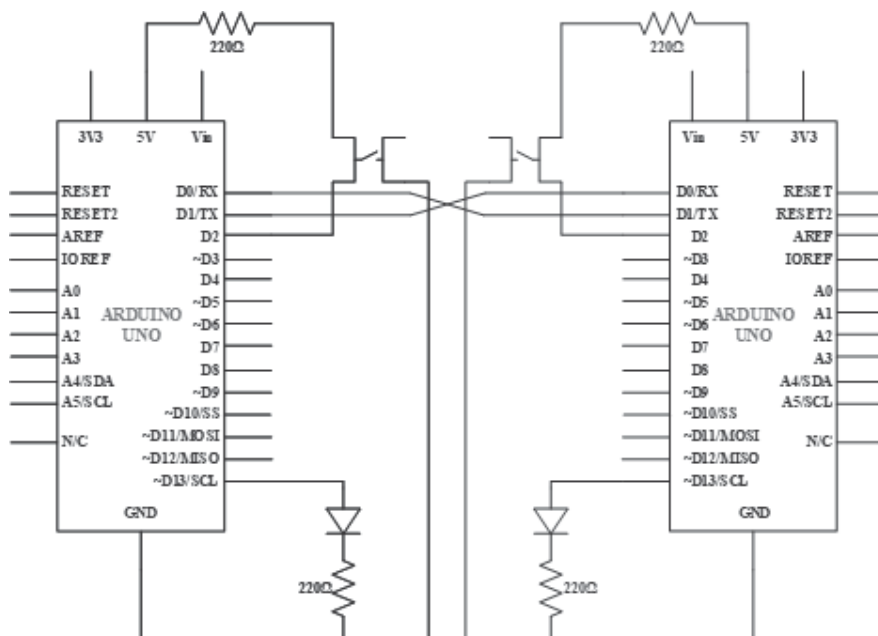
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Dve Arduino hardverske platforme
- Četiri otpornika od po 220Ω
- Dve diode iste boje
- Protobord
- 10 MM kratkospajča
- Dva tastera

Hardverska realizacija sistema prikazana je na slici 10 - 21, dok je električna šema koja ilustruje način povezivanja prikazana na slici 10 - 22.



Slika 10 - 21 - Hardverska realizacija sistema iz primera 10.2



Slika 10 - 22 - Električna šema sistema iz primera 10.2

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
//System variables
int DiodePinNo          = 13;
int ButtonPinNo         = 2;
int BouncingTime        = 200;
int ButtonStatus;
int DiodeStatus = LOW;
//Serial variables
const int BaudRate = 9600;
//Recieved string is stored here
char ControlChar;
void setup() {
    //Diode pin direction as output
    pinMode(DiodePinNo,OUTPUT);
    //Button pin direction as input
    pinMode(ButtonPinNo,INPUT);
    //Initialize diode state
    digitalWrite(DiodePinNo, DiodeStatus);
    //Attach interrupt to the pin
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
    Serial.Begin(BaudRate);
}
void ButtonISR()
{

```

```

//wait
delay(BouncingTime);
//check again if button is pressed
ButtonStatus = digitalRead(ButtonPinNo);
if(ButtonStatus == LOW){
    //Send signal to other Arduino
    Serial.write('P');
}
void loop() {
    //Check receive buffer
    if(Serial.available()>0)
    {
        //Check if signal received?
        ControlChar = Serial.read();
        if(ControlChar == 'P')
        {
            DiodeStatus = !DiodeStatus;
            digitalWrite(DiodePinNo, DiodeStatus);
        }
    }
}
}

```

Svaki od kontrolera detektuje pritiskanje svog tastera uz pomoć prekida na digitalnom pinu za koji je vezan taster. U okviru prekidne rutine se serijskom vezom šalje karakter „P“ kao informacija da je taster pritisnut. U okviru *loop* funkcije svaki kontroler proverava da li ima novu poruku koja je stigla preko serijske veze i ukoliko je stigao karakter „P“, invertuje stanje digitalnog izlaza za koji je vezana dioda (pali ili gasi diodu).

Primer 10.3. - Slanje poruka putem Bluetooth modula

- *Realizovati Arduino sistem koji sadrži Bluetooth modul. Sistem treba da bude realizovan tako da sa periodom od 2 sekunde, putem Bluetooth modula, šalje poruku sadržine „Internet stvari“ na instaliranu aplikaciju na mobilnom telefonu koja predstavlja Bluetooth terminal. Napisati kod koji implementira zahtevanu funkcionalnosti i praktično potvrditi uspešnost realizacije.*

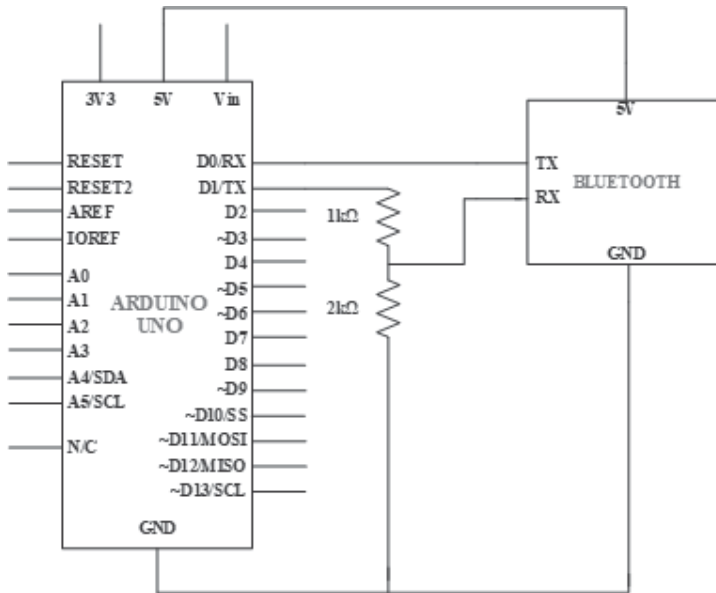
Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

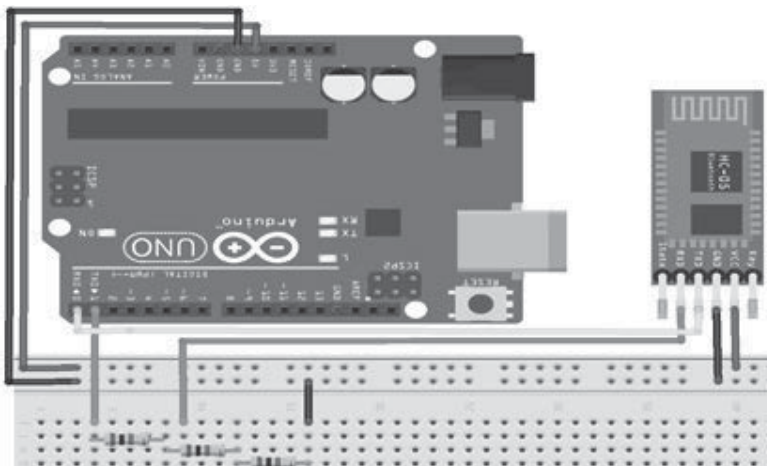
- Arduino UNO
- Otpornik od 1kΩ i otpornik od 2kΩ
- Protobord
- 4 MM kratkospajача

- 4 MŽ kratkospajача
- Bluetooth modul

Električna šema sistema i njegova fizička realizacija u Arduino orkuženju prikazani su na slikama 10 - 23 i 10 - 24 respektivno.



Slika 10 - 23 - Električna šema sistema iz primera 10.3



Slika 10 - 24 - Hardverska realizacija sistema iz primera 10.3

Kada smo povezali sistem kao na prikazanim šemama neophodno je izvršiti uparivanje bluetooth modula i telefona kao što je objašnjeno u poglavlju 10.1. Kod koji realizuje funkcionalnost zahtevanu postavkom zadatka je sledeći:

```
#include <TimerOne.h>
const long BaudRate = 9600;
const unsigned long MsgPeriod= 2000000;

void SendMsg(){
    Serial.write("Internet Stvari");//Send string
}

void setup() {
    Serial.begin(BaudRate); //Init communication
    Timer1.initialize(MsgPeriod);
    Timer1.attachInterrupt(SendMsg);
}
void loop() {
}
```

Najpre se odabere brzina komunikacije (*BaudRate*) sa uređajem. Nakon toga se inicijalizuje komunikacija a zatim se, u prekidnoj funkciji tajmera *SendMsg*, šalje poruka "Internet stvari".

Primer 10.4. - Slanje očitane temperature putem Bluetooth modula

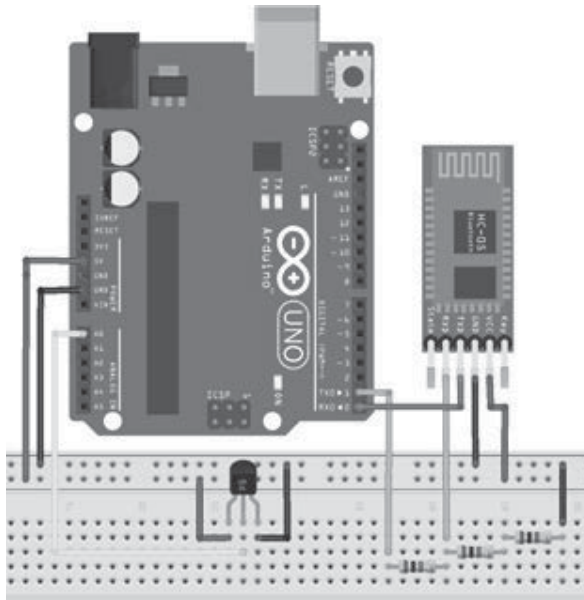
- *Realizovati Arduino sistem koji se sastoji od senzora temperature LM35 i bluetooth modula. Sistem treba da očitava temperaturu sa senzora LM35 i da očitanu vrednost temperature, u Celzijusovoj skali, šalje putem Bluetooth modula na terminal aplikacije na telefonu. Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

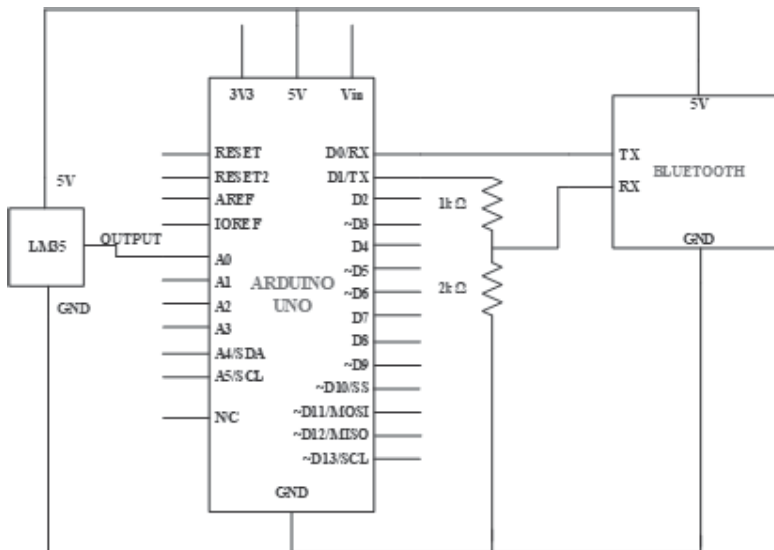
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Otpornici od 1kΩ i 2kΩ
- Protobord
- 7 MM kratkospajča
- 4 MŽ kratkospajča
- Temperaturni senzor LM35
- Bluetooth modul

Fizički izgled sistema, kao i električna šema koja ilustruje način povezivanja, prikazani su na slikama 10 - 25 i 10 - 26.



Slika 10 - 25 - Hardverska realizacija sistema iz primera 10.4



Slika 10 - 26 - Električna šema sistema iz primera 10.4

Temperaturni senzor LM35 povezuje se sa analognim pinom A0. Formula za konvertovanje temperature izvedena je u poglavlju 8.3 (formula (10)). Kod koji realizuje zahtevanu funkcionalnost je:

```
#include <TimerOne.h>
#define SensorPin A1
const int BaudRate = 9600;
const int DelayTime = 100; //ms
volatile float Temperature;
volatile int counter = 1;
const int NumberOfSamples = 100;
void ReadSensor(){
    Temperature += (5.0 * analogRead(SensorPin) * 100.0) / 1024;
    counter++;
    if(counter == NumberOfSamples)
    {
        Temperature /= counter;
        Serial.println(Temperature);
        Temperature = 0;
        counter = 1;
    }
}
void setup() {
    //Init serial communication
    Serial.begin(BaudRate);
    Timer1.initialize(ReadPeriod);
    Timer1.attachInterrupt(ReadSensor);
}
void loop() {
}
```

Primer 10.5. – Merenje razdaljine

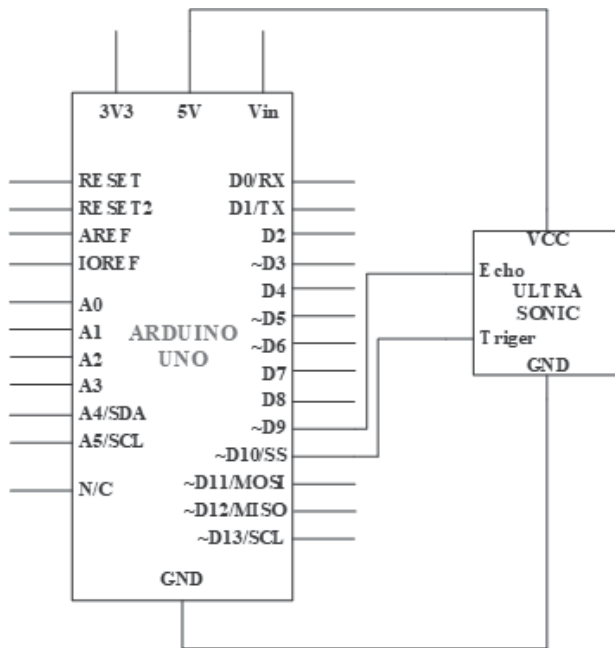
- *Realizovati Arduino sistem koji služi za merenje razdaljine i izmerene rezultate ispisuje na serijsku konzolu. Razdaljina se meri pomoću ultrazvučnog modula. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

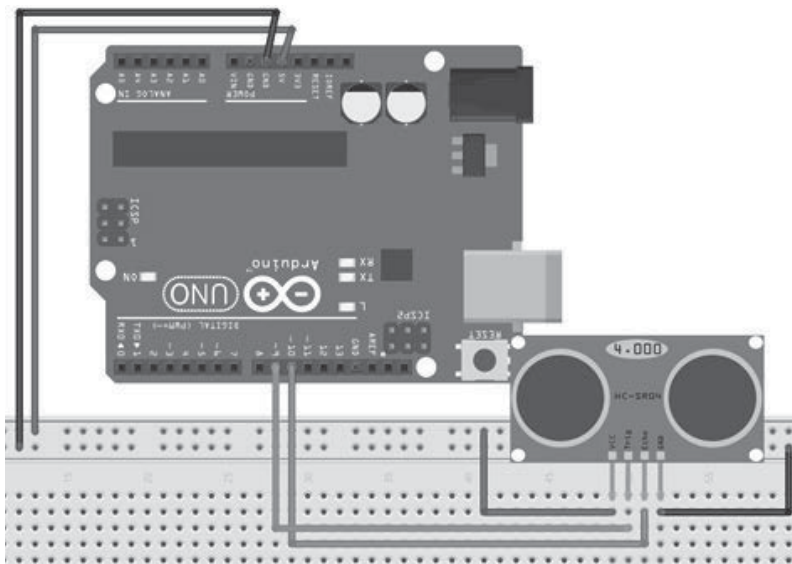
Za realizaciju ovog sistema neophodni su sledeći elementi:

- Arduino UNO
- 6 MM kratkospajача
- Protobord
- Ultrazvučni senzor

Električna šema koja ilustruje način povezivanja komponenti kao i fizička realizacija sistema u Arduino okruženju prikazani su na slikama 10 - 27 i 10 - 28.



Slika 10 - 27 - Električna šema sistema iz primera 10.5



Slika 10 - 28 - Hardverska realizacija sistema iz primera 10.5

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
#include <TimerOne.h>
const int EchoPin = 10;
const int TrigPin = 9;

volatile unsigned long Duration;
volatile float Distance;

const unsigned long TimerRes = 2000;
//Define system state
typedef enum {Transmit, Receive, Print, Wait} ModulState;
ModulState CurrentState;
//Used to created diferent timer periods
volatile int TimerCounter = 0 ;
void TimerISR(){
    TimerCounter++;
    switch(CurrentState){
        case Transmit:
            //Generate Trig level to high which duration is 10seconds
            digitalWrite(TrigPin,HIGH);
            if(TimerCounter == 4){
                CurrentState = Receive;
                TimerCounter = 0;
            }
            break;
        case Receive:
            //Pull down Trig level
            digitalWrite(TrigPin,LOW);
            //Listen on the Echo pin
            Duration = pulseIn(EchoPin,HIGH);
            //Calculate distance
            Distance = Duration*0.034/2;
            CurrentState = Print;
            TimerCounter = 0;
            break;
        case Print:
            Serial.println(Distance);
            CurrentState = Wait;
            TimerCounter = 0;
            break;
        case Wait:
            //wait fot 1seconds
            if(TimerCounter == 500){
                CurrentState = Transmit;
                TimerCounter = 0;
            }
            break;
    }
}

void setup() {
    pinMode(EchoPin, INPUT);
    pinMode(TrigPin, OUTPUT);
    Serial.begin(9600);
}
```

```
CurrentState = Transmit;
digitalWrite(TrigPin, LOW);

Timer1.initialize(TimerRes);
Timer1.attachInterrupt(TimerISR);
}

void loop() {
}
```

Merenje vremenskog trajanja određenog logičkog naponskog nivoa na nekom pinu Arduina, vrši se pomoću funkcije

pulseIn(pin, value)

pin – broj pina na kome merimo trajanje nivoa signala

value – logički nivo za koji se vrši merenje. Moguće vrednosti HIGH ili LOW

Ovaj primer ima za cilj da demonstrira rad ultrazvučnog modula. Prvo se generiše signal na *Trig* pinu u trajanju od 10 mili sekundi, a zatim se mere impulsi na *Echo* pinu.

Promenljive *EchoPin* i *TrigPin* predstavljaju pinove Arduino hardverske platforme na koje su povezani odgovarajući pinovi ultrazvučnog senzora. Vrednost promenljive *Duration* predstavlja vreme trajanja signala na *Echo* pinu, dok vrednost promenljive *Distance* predstavlja izračunatu razdaljinu.

Vrednost promenljiva *TimerRes* predstavlja periodu pozivanja tajmerske funkcije *TimerISR*. Unutar ove funkcije se sprovodi algoritam za merenje razdaljine. Ovaj algoritam koristi promenljivu *CurrentState* u switch – case strukturi. Ova promenljiva je nabrojivog tipa *SystemState* i označava trenutnu operaciju koju sistem izvršava.

Promenljiva *TimerCounter* se koristi u cilju računanja vremenskih intervala na osnovu rezolucije tajmera. Naime, tajmerska funkcija se poziva sa periodom od 2ms. Svaki put kada se uđe u prekidnu funkciju tajmera, vrednost ove promenljive se uveća za jedan. To znači da se proteklo vreme dobija množenjem ove promenljive sa 2ms.

Prilikom prvog ulaska u funkciju *TimerISR*, vrednost promenljive *CurrentState* je *Transmit*, što označava da sistem trenutno emituje signal na *Echo* pinu. U ovom stanju sistem ostaje 8ms dok je trajanje signala na *Echo* pinu 10ms. Razlika od 2ms potiče iz činjenica da je promena stanja izvršena posle 8ms, u stanju *Transmit*, dok je upis vrednosti LOW na *Echo* pin izvršen u *Receive* stanju u koje će se sistem naći tek nakon 2ms.

Unutar *Receive* dela programkog koda, vrši se merenje trajanja signala na *Echo* pinu i računa se razdaljina. Nakon toga se odlazi u *Print* deo koda koji je zadužen za ispis na serijsku konzolu.

Opisana softverska implementacija ima karakteristike mašine stanja. Detaljno objašnjena mašina stanja dato je u poglavlju 12.

Primer 10.6. – Instrument za merenje razdaljine sa LED displejom

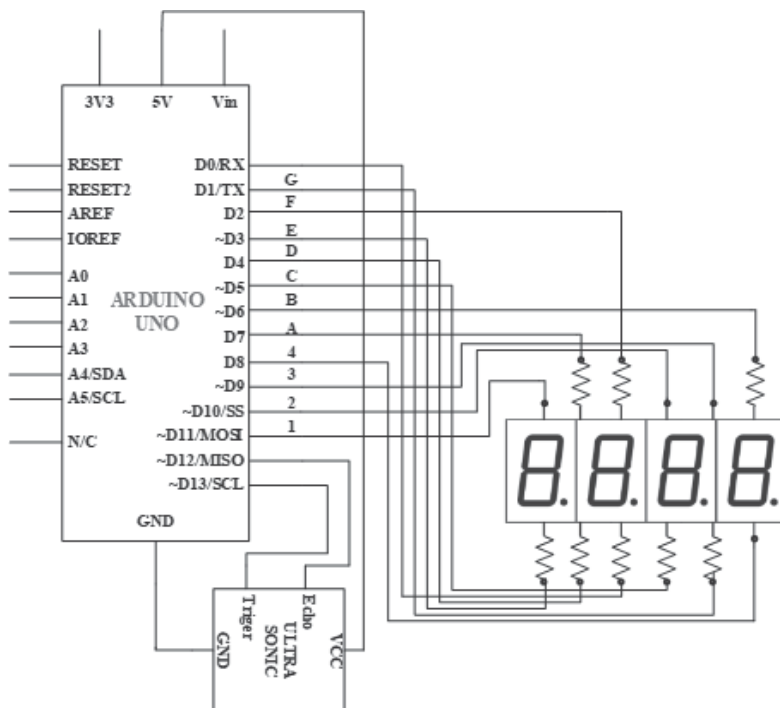
- Realizovati Arduino sistem koji predstavlja digitalni instrument za merenje udaljenosti. Sistem se sastoji od ultrazvučnog senzora i četvorocifrenog LED displeja. Na displeju je neophodno ispisivati trenutnu udaljenost nekog objekta u centimetrima. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 12 MΩ kratkospajča
- 4 MM kratkospajča
- Protobord
- Četvorocifreni LED displej
- Ultrazvučni senzor

Električna šema koja ilustruje način povezivanja ovih komponenti prikazana je na slici 10 - 29.



Slika 10 - 29 - Električna šema sistema iz primera 10.6

Kod koji realizuje implementiranu funkcionalnost je sledeći:

```
#include <TimerOne.h>
#include <7Seg.h>
const int EchoPin = 12;
const int TrigPin = 13;
volatile unsigned long Duration;
volatile unsigned long Distance;
volatile int TimerCounter = 0;
volatile int Digit1 = 0;
volatile int Digit2 = 0;
volatile int Digit3 = 0;
volatile int Digit4 = 0;
const unsigned long TimerRes = 1000;
int ActiveDsp = 4;
void setup() {
    Init7Seg();
    pinMode(EchoPin, INPUT);
    pinMode(TrigPin, OUTPUT);
    digitalWrite(TrigPin, LOW);
    Timer1.initialize(TimerRes);
    Timer1.attachInterrupt(TimerISR);
}
void Convert(const int Number)
{
    int Temp = Number;
    Digit1 = Temp/1000;
    Temp%=1000;
    Digit2 = Temp/100;
    Temp%=100;
    Digit3 = Temp/10;
    Temp%=10;
    Digit4 = Temp;
}
void DispRefresh(){
    switch(ActiveDsp)
        case 1:
            //Turn off display 4
            TurnDspOff(4);
            //Turn on display 3
            TurnDspOn(1);
            //Write digit to dsp3
            WriteNum(Digit1);
            //Next active display is 4
            ActiveDsp = 2;
            break;
        case 2:
            //Turn off display 4
            TurnDspOff(1);
```

```

//Turn on display 3
  TurnDspOn(2);
//Write digit to dsp3
  WriteNum(Digit2);
//Next active display is 4
  ActiveDsp = 3;
break;
case 3:
//Turn off display 4
  TurnDspOff(2);
//Turn on display 3
  TurnDspOn(3);
//Write digit to dsp3
  WriteNum(Digit3);
//Next active display is 4
  ActiveDsp = 4;
break;
case 4:
//Turn off display 3
  TurnDspOff(3);
//Turn on display 4
  TurnDspOn(4);
//Write lower digit
  WriteNum(Digit4);
// Next active display is 3;
  ActiveDsp = 1;
break;
}
}
void TimerISR()
{
  TimerCounter++;
  DispRefresh();
  if(TimerCounter >= 500){
    if(TimerCounter <= 510){
      digitalWrite(TrigPin,HIGH);
    }
    else{
      digitalWrite(TrigPin,LOW);
      Duration = pulseIn(EchoPin,HIGH);
      Distance = Duration*0.034/2;
      Convert(Distance);
      TimerCounter = 0;
    }
  }
}
}
void loop() {
}

```

Ovaj primer ilustruje jedan drugačiji pristup realizaciji sistema. Signal na *Triger* pinu se generiše unutar *TimerISR* funkcije koja predstavlja prekidnu funkciju tajmera. Takođe se unutar ove funkcije poziva funkcija *DispRefresh* koja je zadužena za multi-pleksiranje LED displeja.

Primer 10.7. - Detekcija vibracija

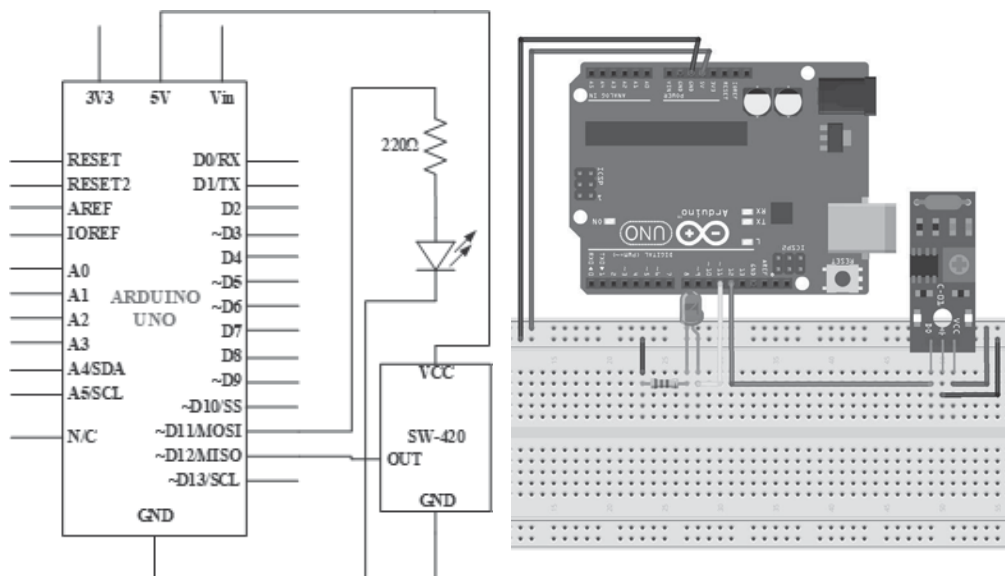
- Realizovati Arduino sistem koji detektuje vibracije na nekom uređaju. Sistem se sastoji od senzora vibracija SW-420 i diode. Sistem treba neprestano, putem serijskog interfejsa, da ispisuje trenutnu vrednost jačine vibracija i ukoliko je ona veća od nekog predefinisano praga, treba da uključi diodu. Dioda treba da bude uključena 1 sekundu nakon detekcije. Napisati kod koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

Rešenje:

Za realizaciju sistema neophodne su sledeće komponente:

- Arduino UNO
- Otpornik 220Ω
- Dioda
- Senzor vibracija SW-40
- 3 MŽ i 3 MM kratkospajача
- Protobord

Električna šema koja ilustruje način povezivanja komponenti i fizička realizacija sistema prikazani su na slici 10 - 30.



Slika 10 - 30 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 10.7

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
#include <TimerOne.h>
const int SensorPinNo = 13;
const int DiodePinNo = 12;

volatile unsigned int VibrationAmplitude = 0;
const int Threshold = 500;
volatile boolean VibrationDetected = true;

const unsigned long ReadPeriod = 1000000;
int TimerCounter = 0;

void TimerISR(){
    if(VibrationDetected == true){
        VibrationDetected = false;
    }
}

void setup() {
    pinMode(SensorPinNo, INPUT);
    pinMode(DiodePinNo, OUTPUT);
    digitalWrite(DiodePinNo, LOW);
    Serial.begin(9600);

    Timer1.initialize(ReadPeriod);
    Timer1.attachInterrupt(TimerISR);
}

void loop() {
    VibrationAmplitude = pulseIn(SensorPinNo, HIGH);
    if(VibrationAmplitude > Threshold){
        VibrationDetected = true;
    }
    if(VibrationDetected == true){
        digitalWrite(DiodePinNo, HIGH);
    }
    else{
        digitalWrite(DiodePinNo, LOW);
    }
}
```

Funkcionalnost ovog koda je podeljena između prekidne rutine i glavne programske petlje *loop*. Nakon detekcije da je amplituda vibracija *VibrationAmplitude* prešla predefinisani prag *Threshold*, vrednost promenljive *VibrationDetected* se postavlja na *true*. Vrednost ove promenljive se menja unutar prekidne rutine tajmera (*TimerISR*) nakon jedne sekunde. Takođe, u glavnoj programskoj petlji, menja se i stanje diode u zavisnosti od promenljive *VibrationDetected*.

Primer 10.8. - Senzor osetljiv na dodir

- Realizovati Arduino sistem koji se sastoji od senzora osetljivog na dodir koji uključuje diodu, tj. sve dok je prst na senzoru dioda treba da bude uključena. Napisati kod za Arduino koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Protobord
- 7 MM kratkospajča
- Senzor osetljiv na dodir

Električna šema koja ilustruje način povezivanja komponenti i fizička realizacija sistema prikazani su na slici 10 - 31.

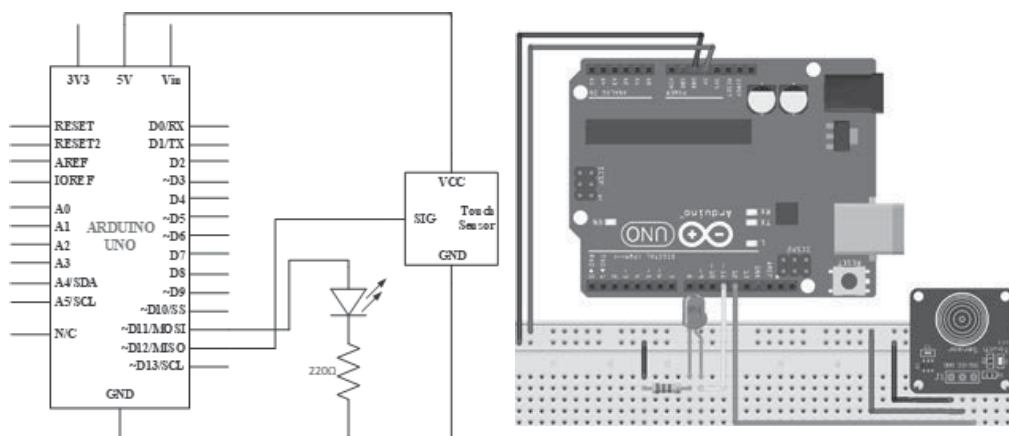
Kod koji realizuje traženu funkcionalnost je sledeći:

```
#include <TimerOne.h>
const int SensorPin = 12;
const int DiodePin = 11;
volatile int Touch = 0;
const unsigned long ReadPeriod = 10000;

void ReadSensorValue(){
    Touch = digitalRead(SensorPin);
}

void setup() {
    pinMode(SensorPin, INPUT);
    pinMode(DiodePin, OUTPUT);
    digitalWrite(DiodePin, LOW);
    Timer1.initialize(ReadPeriod);
    Timer1.attachInterrupt(ReadSensorValue);
}

void loop() {
    if(Touch == HIGH){
        digitalWrite(DiodePin, HIGH);
    }
    else{
        digitalWrite(DiodePin, LOW);
    }
}
```



Slika 10 - 31 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 10.8

Primer 10.9 - Detekcija zvuka

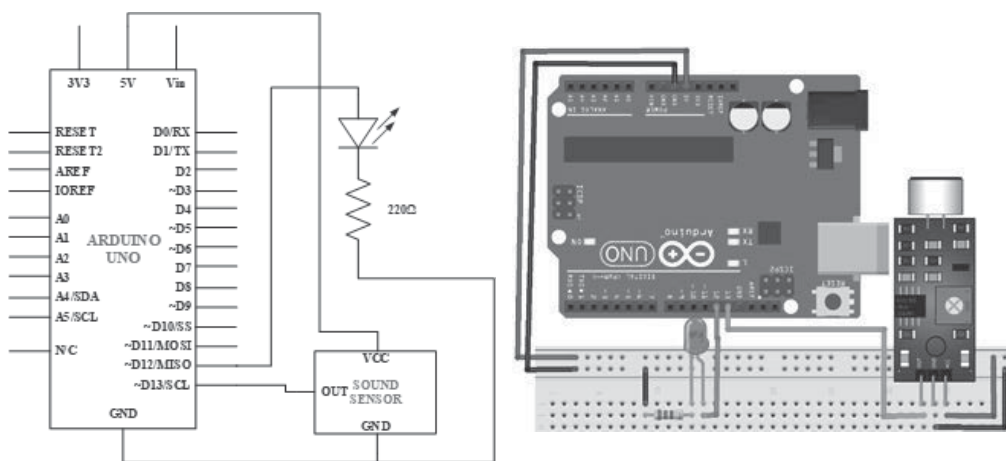
- Realizovati Arduino sistem koji se sastoji od senzora zvuka i LE diode. Potrebno je napisati kod za Arduino koji uključuje diodu pri detekciji zvuka, dok je u suprotnom dioda isključena. Praktično potvrditi uspešnost realizovanog sistema.

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Protobord
- 7 MM kratkospajča
- Arduino UNO
- LE dioda
- Otpornik 220Ω
- Modul za detekciju zvuka

Električna šema koja ilustruje način povezivanja komponenti i izgled fizičke realizacije u Arduino okruženju prikazani su na slici 10 - 32.



Slika 10 - 32 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 10.9

Kod koji realizuje zahtevanu funkcionalnost je sledeći :

```
const int SoundSensorPinNo = 13;
const int DiodePinNo       = 12;
void setup() {
    pinMode(SoundSensorPinNo, INPUT);
    pinMode(DiodePinNo, OUTPUT);
}
void loop() {
    if(digitalRead(SoundSensorPinNo) == LOW)
    {
        digitalWrite(DiodePinNo, HIGH);
    }
    else
    {
        digitalWrite(DiodePinNo, LOW);
    }
}
```

U glavnoj programskoj petlji stalno se vrši čitanje vrednosti na digitalnom pinu *SoundSensorPinNo* na koji je povezan izlaz modula za detekciju zuka. Ukoliko je očitana vrednost jednala *LOW*, uključuje se dioda dok je u suprotnom dioda isključena. Treba primetiti da nivo logičke jedinice označava aktivnu vrednost.

Primer 10.10. - Očitavanje nadmorske visine primenom digitalnog barometra

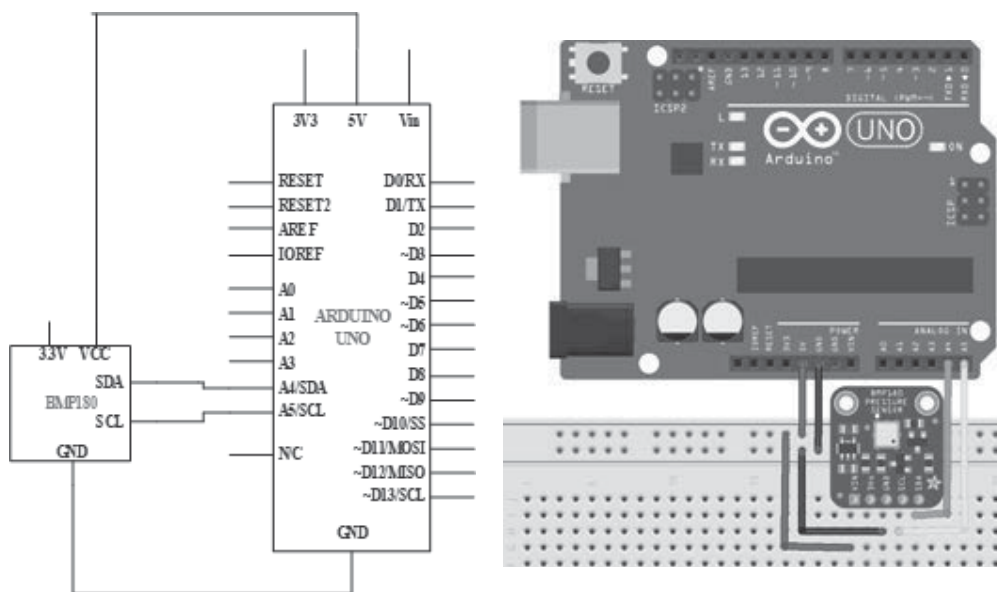
- Realizovati Arduino sistem za očitavanje nadmorske visine. Sistem se sastoji od senzora pritiska. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 6 MM kratkospajča
- BMP180 senzor pritiska
- Protobord

Električna šema povezivanja i fizički izgled sistema prikazani su na slici 10 - 33.



Slika 10 - 33 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 10.10

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
#include <SFE_BMP180.h>
#include <Wire.h>
// You will need to create an SFE_BMP180 object,
// here called presssure
SFE_BMP180 pressure;
// baseline pressure
double baseline;

void setup()
{
    Serial.begin(9600);
    // Initialize the sensor (it is important to get calibration
    // values stored on the device).
    if (pressure.begin())
        Serial.println("Uspesna inicijalizacija");
    else
    {
        // Oops, something went wrong
        Serial.println("Neuspesna inicijalizacija\n\n");
        // Pause forever.
        while(1);
    }
    // Get the baseline pressure:
    baseline = getPressure();
}

double getPressure()
{
    char status;
    double T,P,p0,a;
    // You must first get a temperature measurement to perform a pressure
    // reading.
    // Start a temperature measurement:
    status = pressure.startTemperature();
    if (status != 0)
    {
        // Wait for the measurement to complete:
        delay(status);
        // Retrieve the completed temperature measurement:
        status = pressure.getTemperature(T);
        if (status != 0)
        {
            // Start a pressure measurement:
            status = pressure.startPressure(3);
            if (status != 0)
            {
                // Wait for the measurement to complete:
                delay(status);
                // Retrieve the completed pressure measurement:
                status = pressure.getPressure(P,T);
                if (status != 0)
                {
```

```

        return(P);
    }
    else Serial.println("Nije moguće očitati pritisak\n");
}
else Serial.println("Nije moguće započeti očitavanje pritiska\n");
}
else Serial.println("Nije moguće očitati temperaturu\n");
}
else Serial.println("Nije moguće započeti očitavanje temperature\n");
}
void loop()
{
    double a,P;
    // Get a new pressure reading:
    P = getPressure();
    // Show the relative altitude difference between
    // the new reading and the baseline reading:
    a = pressure.altitude(P,baseline);
    Serial.print("Relativna visina: ");
    if (a >= 0.0) Serial.print(" "); // add a space for positive numbers
    Serial.print(a,1);
    Serial.print(" m \n ");
}

```

Kada radimo sa ovim senzorom i želimo da pročitamo vrednost pritiska, najpre je neophodno pročitati vrednost temperature, a zatim vrednost pritiska. Očitavanje neke vrednosti (temperature ili pritiska) se vrši tako što prvo započnemo proces čitanja a nakon toga vršimo čitanje vrednosti.

Započinjanje procesa čitanja vrednosti temperature se vrši pozivom funkcije:

return = startTemperature(void);

return – pozitivna vrednost (predstavlja broj ms koliko treba čekati) ukoliko je uspešno započet proces čitanja dok je u suprotnom 0

Kada se završi proces očitavanja vrednosti temperature, trenutnu vrednost temperature je moguće pročitati pozivom funkcije:

return = getTemperature(Temperature);

return – 1 ukoliko je očitavanje uspešno, 0 ukoliko je očitavanje neuspešno

Temperature – očitana vrednost temperature

Započinjanje procesa čitanja vrednosti pritiska vrši se pozivom funkcije:

return = startPressure(3);

return – pozitivna vrednost (predstavlja broj ms koliko treba čekati) ukoliko je uspešno započet proces čitanja dok je u suprotnom 0

Kada se završi proces očitavanja vrednosti pritiska, trenutnu vrednost pritiska je moguće pročitati pozivom funkcije:

return = getPressure(Presure, Temperature);

Presure – očitana vrednost pritiska

Temperature – prethodno izmerena temperatura

return – 1 ako je očitavanje vrednosti pritiska uspešno izvršeno dok je u suprotnom ova vrednost 0.

Kada imamo vrednost pritiska na osnovu razlike pritisaka moguće je odrediti relativnu visinu. To nam omogućava funkcija:

Alt = altitude(CurrentPressure, BasePressure);

CurrentPressure – Trenutna vrednost pritiska

BasePressure – Vrednost pritiska u donosu na koju računamo visinu

Alt – Izračunata visina izražena u metrima

Na početku je neophodno uključiti dve biblioteka. Biblioteka *Wire* omogućava pristup senzoru putem I2C interfejsa dok biblioteka *SPE_BMP180* implementira same funkcije za rad sa senzorom. Nakon toga deklarišemo globalnu promenljivu *pressure* koja će se koristiti za pristup ovim funkcijama. Promenljiva *basepressure* nosi informaciju o pritisku u odnosu na koji računamo visinu. Vrednost ove promenljive se dobija pozivom funkcije *getPressure()* prilikom inicijalizacije sistema u funkciji *setup*. Pored definisanja vrednosti promenljive *basepressure*, u funkciji *setup* se vrši uspostavljanje komunikacije sa senzorom. Ukoliko je komunikacija uspešno uspostavljena, sistem nastavlja sa radom dok se u suprotnom njegov rad pauzira u beskonačnoj *while(1)* petlji.

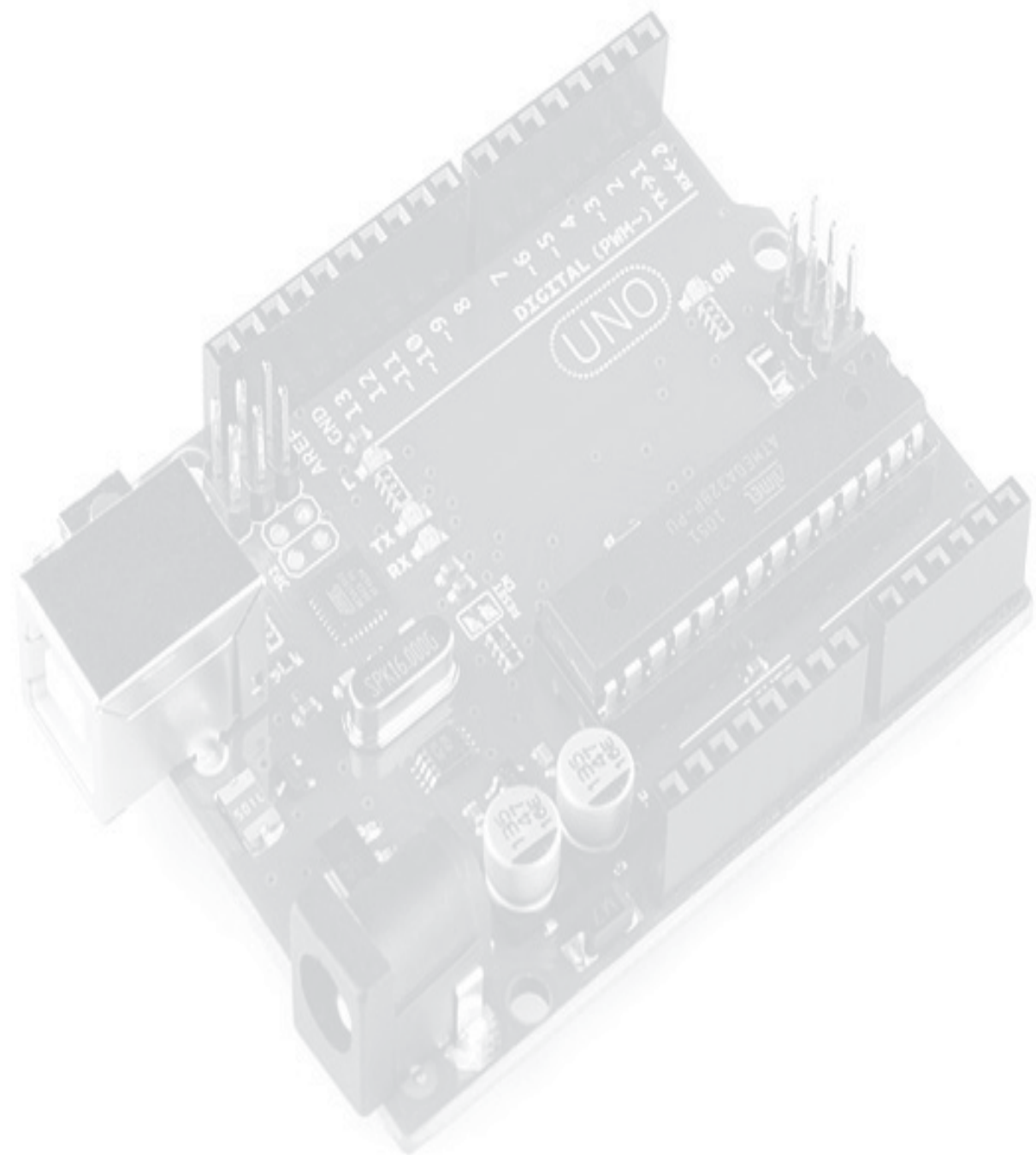
Kada imamo vrednost promenljive *basepressure* u odnosu na koju merimo pritisak, neophodno je izmeriti trenutnu vrednost pritiska i smestiti je u promenljivu *P*. Nakon toga, pozivom funkcije *altitude* izračunavamo visinu.

Dakle, u ovom zadatku ne računamo apsolutnu (nadmorsku) visinu već relativnu visinu. Visina se meri u odnosu na položaj uređaja tokom inicijalizacije. To znači da je u procesu inicijalizacije, u funkciji *setup*, izmeren pritisak primenom funkcije *getPressure()* i ta vrednost je smeštena u promenljivu *baseline*. Vrednost ove promenljive predstavlja referentnu vrednost pritiska. U glavnoj programskoj petlji svaki put vršimo merenje trenutnog pritiska. Pozivom funkcije *altitude()*, kojoj prosleđujemo referentnu i trenutnu vrednost pritiska, dobijamo relativnu visinu, tj. visinu na kojoj se nalazi uređaj u odnosu na poziciju na kojoj je bio u trenutku inicijalizacije sistema.

10.8. ZADACI ZA SAMOSTALNI RAD

1. Realizovati Arduino sistem koji se sastoji od jedne diode i bluetooth modula. Kontrola diode se vrši slanjem poruka putem Bluetooth terminala na telefonu. Kada na Arduino pristigne poruka sadržine „DON“ treba uključiti diodu, dok u slučaju da pristigne poruka „DOFF“ diodu treba isključiti. Sve ostale poruke se ignorišu. Napisati kod za Arduino UNO hardversku platformu i praktično potvrditi uspešnost realizacije.
2. Realizovati Arduino sistem koji se sastoji od tri diode (crvena, žuta i zelena), tastera i bluetooth modula. Promena aktivne diode se može vršiti putem tastera ili putem bluetooth modula. Promena aktivne diode podrazumeva da ukoliko je bila aktivna crvena dioda, sada aktivna dioda postaje žuta, dok ukoliko je bila žuta, aktivna dioda postaje zelena dioda. Ukoliko je aktivna dioda bila zelena, nova aktivna dioda postaje crvena dioda. Sistem treba da reaguje na pritisak tastera i na poruku pristiglu sa bluetooth modula tako što će promeniti aktivnu diodu. Ukoliko je poruka pristigla sa bluetooth modula menja se aktivna dioda jedino ako je sadržina poruke identična stringu „CHANGE_AD“, dok u suprotnom sistem ne reaguje. Napisati kod koji implementira zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
3. Realizovati sistem koji se sastoji od 3 diode (crvena, žuta i zelena), tastera i bluetooth modula. Promena aktivne diode vrši se sa periodom od 5 sekundi. Promena aktivne diode podrazumeva da ukoliko je bila aktivna crvena dioda, sada aktivna dioda postaje žuta, dok ukoliko je bila žuta, nova aktivna dioda postaje zelena dioda. Ukoliko je aktivna dioda bila zelena, nova aktivna dioda postaje crvena dioda. Pritiskom na taster se na bluetooth modul šalje poruka koja označava koja dioda je trenutno aktivna. Napisati kod koji implementira zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
4. Realizovati sistem od jedne diode i bluetooth modula. Dioda treperi sa nekom početnom periodom od 2 sekunde. Putem bluetooth modula na Arduino hardversku platformu stižu dve poruke „UP“ i „DOWN“. Kada Arduino hardverska platforma primi poruku „UP“ tada treba povećati periodu treperenja dok u slučaju poruke sadržine „DOWN“ treba smanjiti periodu treperenja diode. Korak promene vrednosti periode je 0.1s. Kada dođe do promene periode sa Arduino platforme na Bluetooth terminal na telefonu treba poslati poruku koja sadrži trenutnu periodu. Implementacija ne treba da dozvoli da perioda postane manja od 0 sekundi. Napisati kod koji implementira zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
5. Realizovati Arduino sistem za merenje udaljenosti koji se sastoji od dve diode i ultrazvučnog senzora. Crvena dioda se uključuje ukoliko je prepreka od senzora udaljena manje od 4 cm, dok je u suprotnom uključena zelena dioda. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

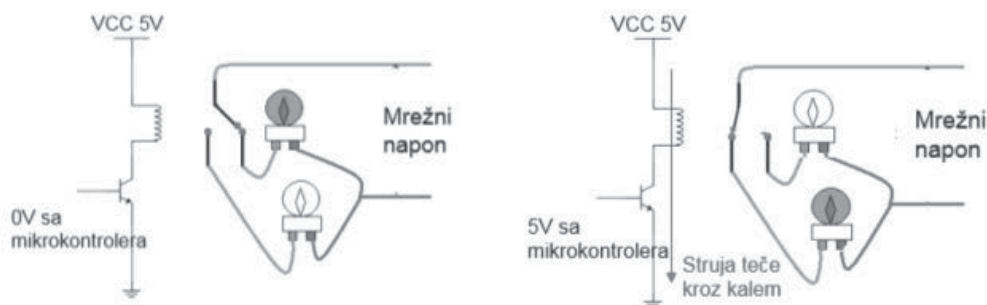
6. Realizovati Arduino sistem za merenje udaljenosti koji rezultate merenja ispisuje na LCD displeju. Praktično potvrditi uspešnost realizacije.
7. Realizovati Arduino sistem za merenje udaljenosti koji na jednocifrenom LE displeju prikazuje razdaljinu ukoliko je manja od 10cm. U suprotnom displej je isključen. Napisati kod za Arduino hardversku platformu koji realizuje traženu funkcionalnost i praktično potvrditi uspešnost realizacije.
8. Realizovati Arduino sistem koji služi za merenje trenutnog vazdušnog pritiska i rezultate merenja ispisuje na LCD displej. Pored vazdušnog pritiska na displeju se ispisuje i trenutna temperatura. Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
9. Realizovati Arduino sistem za merenje nadmorske visine koji rezultate merenja ispisuje na četvorocifrenom sedmosegmentnom displeju. Za merenje se koristi senzor pritiska BMP180. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.
10. Realizovati Arduino sistem koji služi za uključivanje diode pljeskanjem ruku. Za realizaciju ovog sistema neophodan je senzor zvuka i dioda. Napisati kod koji implementira uključivanje diode nakon pljeskanja rukama i praktično potvrditi uspešnost realizacije.



11. RAD SA RELEJIMA

Relej je električno kontrolisani prekidač čija je osnovna namena, a da prekida struju u kolima naizmeničnog napona ili kolima jednosmernog napona a velike snage. Arduino kontroler radi sa jednosmernim naponom napajanja od 5V i u svim primerima do sada smo uz pomoć Arduina direktno palili LED diode. Međutim nije moguće na isti način, direktno uz pomoć Arduina paliti i gasiti neke druge potrošače, kao što je na primer sobna lampa koja radi na mrežnom napajanju od 220V naizmeničnog napona. U takvoj situaciji može se koristiti relej, preko koga bi Arduino kontroler neposredno palio i gasio sobnu lampu.

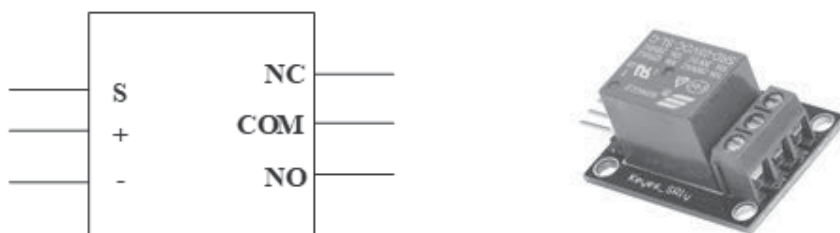
Relej najčešće sadrži kalem sa namotajima bakarne žice, kroz koji kada prolazi struja, generiše se privlačna magnetna sila. Ovaj kalem se povezuje na stranu kontrolera, koji uz pomoć napona na odgovarajućem digitalnom pinu uspostavlja ili prekida struju kroz kalem. Sa druge strane (strane na kojoj su priključci za koje se kači kolo visoke snage) nalazi se kotva koja može da menja položaj. Kada nema magnetne sile kalema, zbog blage opruge koja je gura, kotva se nalazi u jednom od dva moguća položaja. Međutim, kada kroz kalem protiče struja, privlačna magnetna sila je jača od elastične sile opruge, pa prevlači kotvu u drugi položaj. Na ovaj način se vrši mehaničko spajanje ili prekidanje kola visoke snage. Opisani način rada releja ilustrovan je na slici 11 - 1.



Slika 11 - 1 - Ilustracija načina rada releja

Kroz ovaj priručnik koristi se relej koji se na primarnoj strani kontroliše naponom od 5V dok na sekundarnoj strani može da kontroliše uređaje čije se struje kreću u opsegu od 10A do 15A i koji mogu da se napajaju i sa naponima do 250V. Naravno, ograničenja po pitanju napona i struja na sekundarnoj strani predstavljaju maksimalno dozvoljena ograničenja što, znači da se ovi releji mogu koristiti i za mnogo manje napone. Relan relej koji se koristi u primerima koji se nalaze u ovom priručniku prikazan je na slici 11 - 2.

Relej prikazan na slici 11 - 2 ima 6 pinova koji su raspoređeni u dve grupe. Prvu grupu pinova predstavljaju pinovi na primarnoj strani i oni su S, + i -. Pin + predstavlja pin na koji se dovodi napon napajanja sa mikrokontrolera od 5V dok se na pin – dovodi masa mikrokontrolera. Pin S služi za upravljanje prekidačem unutar releja. Kada je na pinu S nizak (0V) naponski nivo, odnosno stanje logičke nule, na sekundarnoj strani su pinovi COM i NC (Normaly closed) kratkospojeni, dok kada je na pinu S visok napon (5V), odnosno stanje logičke jedinice, pinovi COM i NO (Normaly open) su kratkospojeni.



Slika 11 - 2 - Električni simbol releja koji koristimo u ovom priručniku (levo) i njegov fizički izgled (desno)

11.1. PRIMERI

Primeri koji slede ilustruju rad sa relejima. U praksi, releji se najčešće koriste za prekidanje kola sa naizmeničnom ili jednosmernom strujom ali naponom većim od napona napajanja Arduina (5V). Međutim, zbog sigurnosti i praktičnosti, u primerima koji slede releji prekidaju kola jednosmerne struje koja se napajaju istim napajanjem kao i Arduino.

Primer 11.1. - Kontrola releja tasterom

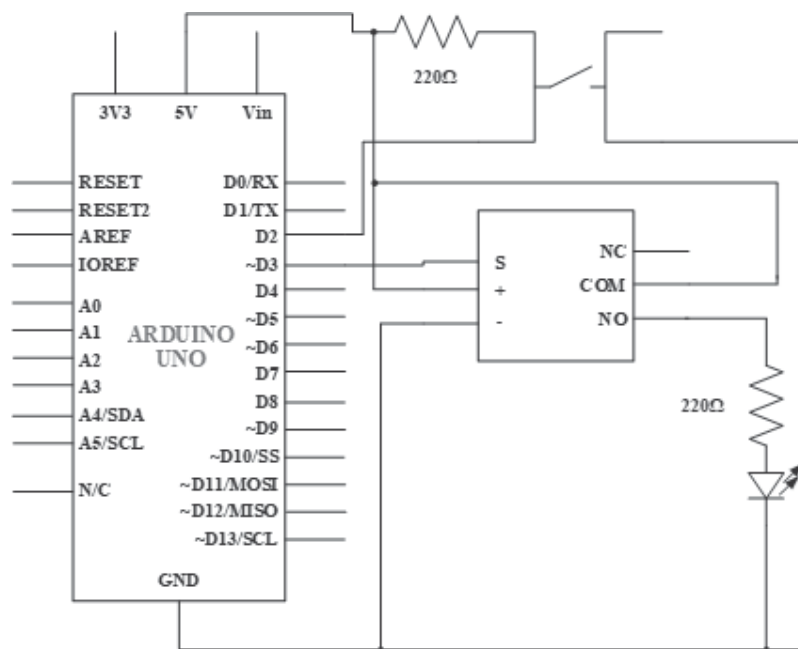
- *Realizovati Arduino sistem koji se sastoji od tastera, releja i diode. Pritisak tastera aktivira relej, koji uključuje ili isključuje diodu. Napisati kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

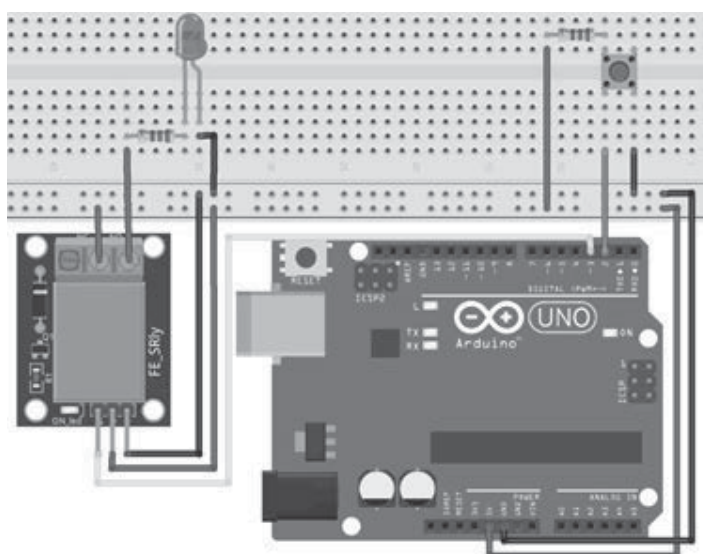
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Dva otpornika od 220Ω
- Dioda
- Protobord
- 8 MM kratkospajача
- 3 MŽ kratkospajача
- Taster
- Relej

Električna šema koja ilustruje način povezivanja navedenih komponenti, kao i prikaz sistema u Arduino orkuženju su dati na slikama 11 - 3 i 11 - 4.



Slika 11 - 3 - Električna šema sistema iz primera 11.1



Slika 11 - 4 - Fizička realizacija sistema iz primera 11.1

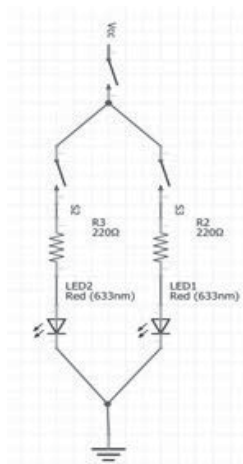
Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
const int ButtonPinNo = 2;
const int RelayPinNo = 3;
volatile int RelayStatus = LOW;
void setup() {
    //Init pins
    pinMode(ButtonPinNo, INPUT);
    pinMode(RelayPinNo, OUTPUT);
    // Init relay pin status
    digitalWrite(RelayPinNo, RelayStatus);
    // Attach interrupt to the pin
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
}
//Button Interrupt
void ButtonISR()
{
    RelayStatus = !RelayStatus;
    digitalWrite(RelayPinNo, RelayStatus);
}
void loop() {
}
```

Kod u ovom primeru praktično je identičan kodu iz primera 5.2, gde se tasterom direktno kontroliše digitalni izlazni pin Arduina koji uključuje ili isključuje diodu. Jedina razlika je što se sada dioda ne kontroliše direktno putem digitalnog pina, već neposredno uz pomoć releja. Dakle, digitalni izlazni pin kontroliše rad releja koji uključuje i isključuje diodu.

Primer 11.2. – Kontrola serijske veze releja

- Realizovati Arduino sistem koji se sastoji od 3 releja, tri tastera i dve diode. Neophodno je realizovati sistem čija je električna šema prikazana na slici 11 - 5. Tasteri 1, 2 i 3 kontrolišu releje 1, 2 i 3 respektivno. Napisati arduino kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.



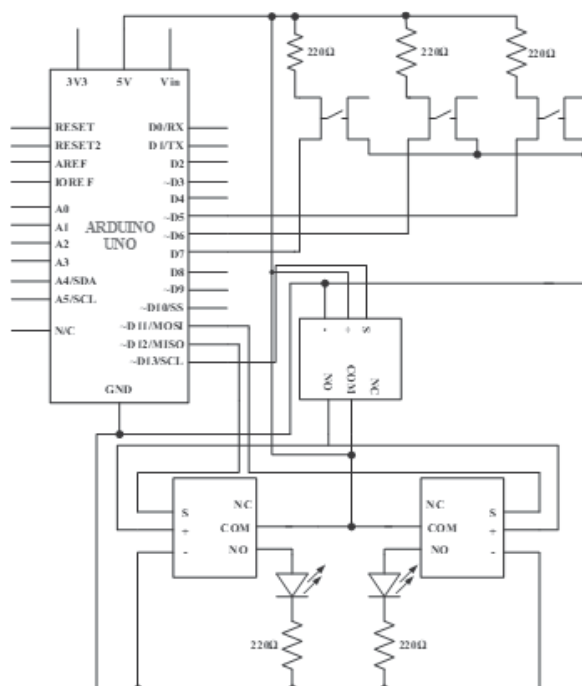
Slika 11 - 5 – Relejno kolo koje treba realizovati u primeru 11.2

Rešenje:

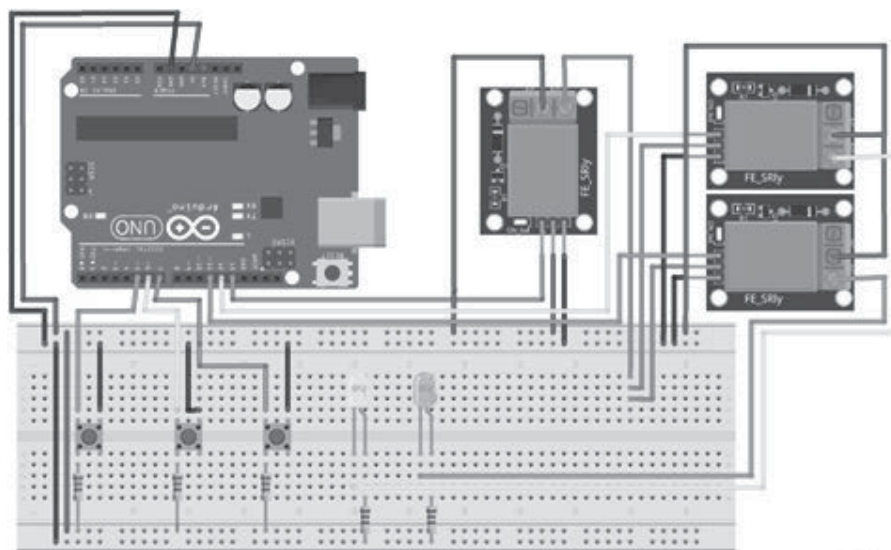
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- Tri otpornika od 220Ω
- Dve diode (zelena i žuta)
- Protobord
- 27 MM kratkospajача
- 9 MŽ Kratkospajача
- 3 tastera
- 3 releja

Nači povezivanja komponenti i fizički izgled sistema dati su na slikama 11 - 6 i 11 - 7.



Slika 11 - 6 - Električna šema sistema iz primera 11.2



Slika 11 - 7 - Hardverska realizacija sistema iz primera 11.2

Kod koji realizuje zahtevanu funkcionalnost je:

```
const int Button1PinNo = 7;
const int Button2PinNo = 6;
const int Button3PinNo = 5;
const int Relay1PinNo = 13;
const int Relay2PinNo = 12;
const int Relay3PinNo = 11;
const int BouncingTime = 100;
int Relay1Status = LOW;
int Relay2Status = LOW;
int Relay3Status = LOW;
int Button1Status ;
int Button2Status ;
int Button3Status ;
void setup() {
    pinMode(Button1PinNo, INPUT);
    pinMode(Button2PinNo, INPUT);
    pinMode(Button3PinNo, INPUT);
    pinMode(Relay1PinNo, OUTPUT);
    pinMode(Relay2PinNo, OUTPUT);
    pinMode(Relay3PinNo, OUTPUT);
    digitalWrite(Relay1PinNo,Relay1Status);
    digitalWrite(Relay2PinNo,Relay1Status);
    digitalWrite(Relay3PinNo,Relay1Status);
}
void loop() {
    Button1Status = digitalRead(Button1PinNo);
    Button2Status = digitalRead(Button2PinNo);
    Button3Status = digitalRead(Button3PinNo);
    if( Button1Status == LOW){
        delay(BouncingTime);
        Button1Status = digitalRead(Button1PinNo);
        if( Button1Status == LOW){
            Relay1Status = !Relay1Status;
            digitalWrite(Relay1PinNo,Relay1Status);
        }
    }
    if( Button2Status == LOW){
        delay(BouncingTime);
        Button2Status = digitalRead(Button2PinNo);
        if( Button2Status == LOW){
            Relay2Status = !Relay2Status;
            digitalWrite(Relay2PinNo,Relay2Status);
        }
    }
    if( Button3Status == LOW){
        delay(BouncingTime);
        Button3Status = digitalRead(Button3PinNo);
        if( Button3Status == LOW){
            Relay3Status = !Relay3Status;
            digitalWrite(Relay3PinNo,Relay3Status);
        }
    }
}
```

Realizovani sistem je takav da dva tastera direktno kontrolišu releje koji su redno spojeni sa diodama, te praktično uključuju i isključuju dve diode u sistemu. Sa druge strane, treći relej ima mogućnost da preseče dotok struje u obe grane sa diodama. Tako da treći taster, koji kontroliše ovaj relej, ima funkciju da uključuje i isključuje obe diode kao centralni prekidač.

Obzirom da se u primeru koriste tri ulazna digitalna pina, a da Arduino ima mehanizam prekida samo na dva pina, u ovom primeru ne koristimo mehanizam prekida, već se stanje ulaznih digitalnih pinova konstantno čita u okviru *loop* funkcije.

Primer 11.3 - Relej sa šifrom

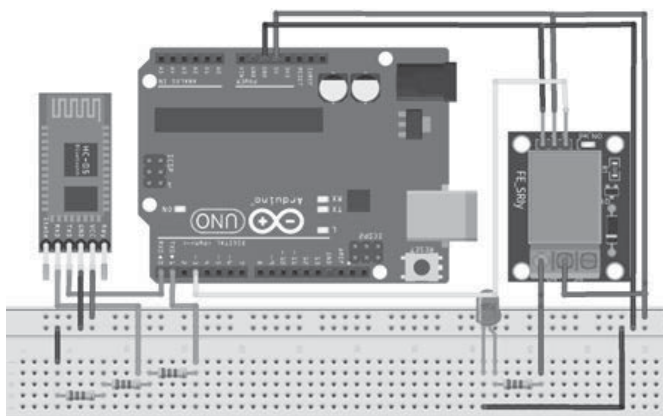
- *Realizovati Arduino sistem koga čine bluetooth modul i relej. Promena stanja releja se vrši samo ako se putem bluetooth terminala na telefonu pošalje string koji odgovara određenoj šifri. Ukoliko korisnik unese pogrešnu šifru potrebno ga je obavestiti da je uneta šifra pogrešna i od njega zatražiti ponovni unos šifre. Napisati kod za Arduino hardversku platformu koji realizuje traženu funkcionalnost sistema i praktično potvrditi uspešnost realizacije.*

Rešenje:

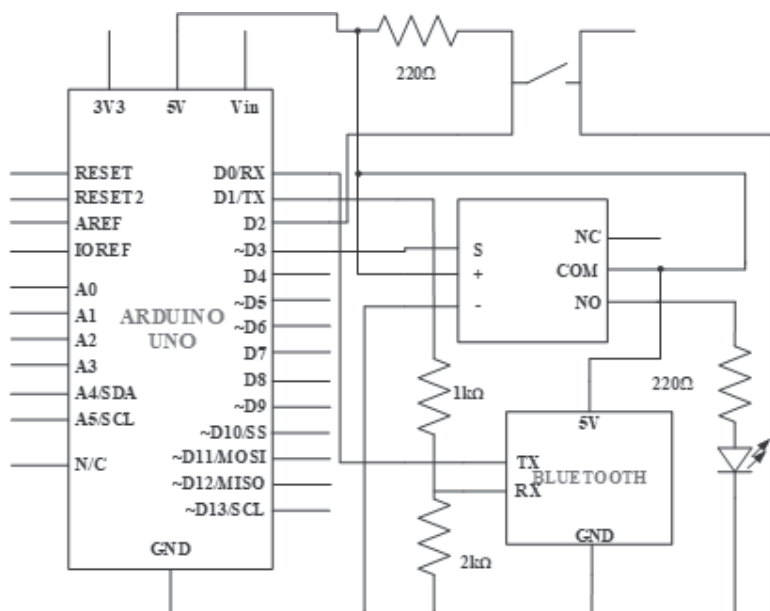
Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 2 Otpornika od 220Ω
- Dioda
- 12 MM kratkospajача
- 3 MŽ kratkospajача
- Protobord
- Bluetooth modul
- Relej

Fizička realizacija sistema i električna šema koja ilustruje način povezivanja dati su na slikama 11 - 8 i 11 - 9.



Slika 11 - 8 - Hardverska realizacija sistema iz primera 11.3



Slika 11 - 9 - Električna šema sistema iz primera 11.3

Kod koji realizuje zahtevanu funkcionalnost je sledeći:

```
const int BaudRate = 9600;
const int RelayPinNo = 13;
int RelayStatus = LOW;
String ReceivedString = "";
```

```

const String Password = "Singi";
void setup()
{
    //Init serial communication
    Serial.begin(BaudRate);
    //Init relay pin
    pinMode(RelayPinNo, OUTPUT);
    digitalWrite(RelayPinNo, RelayStatus);
    //Send string via serial communication
    Serial.println("Unesite sifru:");
}
void SwitchRelay()
{
    RelayStatus = !RelayStatus;
    digitalWrite(RelayPinNo, RelayStatus);

    if(RelayStatus == HIGH){
        Serial.println("Ukljucena dioda");
        Serial.println("Unesite sifru da iskljucite diodu:");
    }
    else{
        Serial.println("Iskljucena dioda");
        Serial.println("Unesite sifru da ukljucite diodu:");
    }
}
void loop()
{
    //Check is Arduino received some data?
    if(Serial.available() > 0)
    {
        //Read received data
        ReceivedString = Serial.readString();
        if(ReceivedString == Password)
        {
            SwitchRelay();
        }
        else
        {
            Serial.println("Pogresna sifra! Pokusajte ponovo");
            Serial.println("Unesite sifru:");
        }
    }
}

```

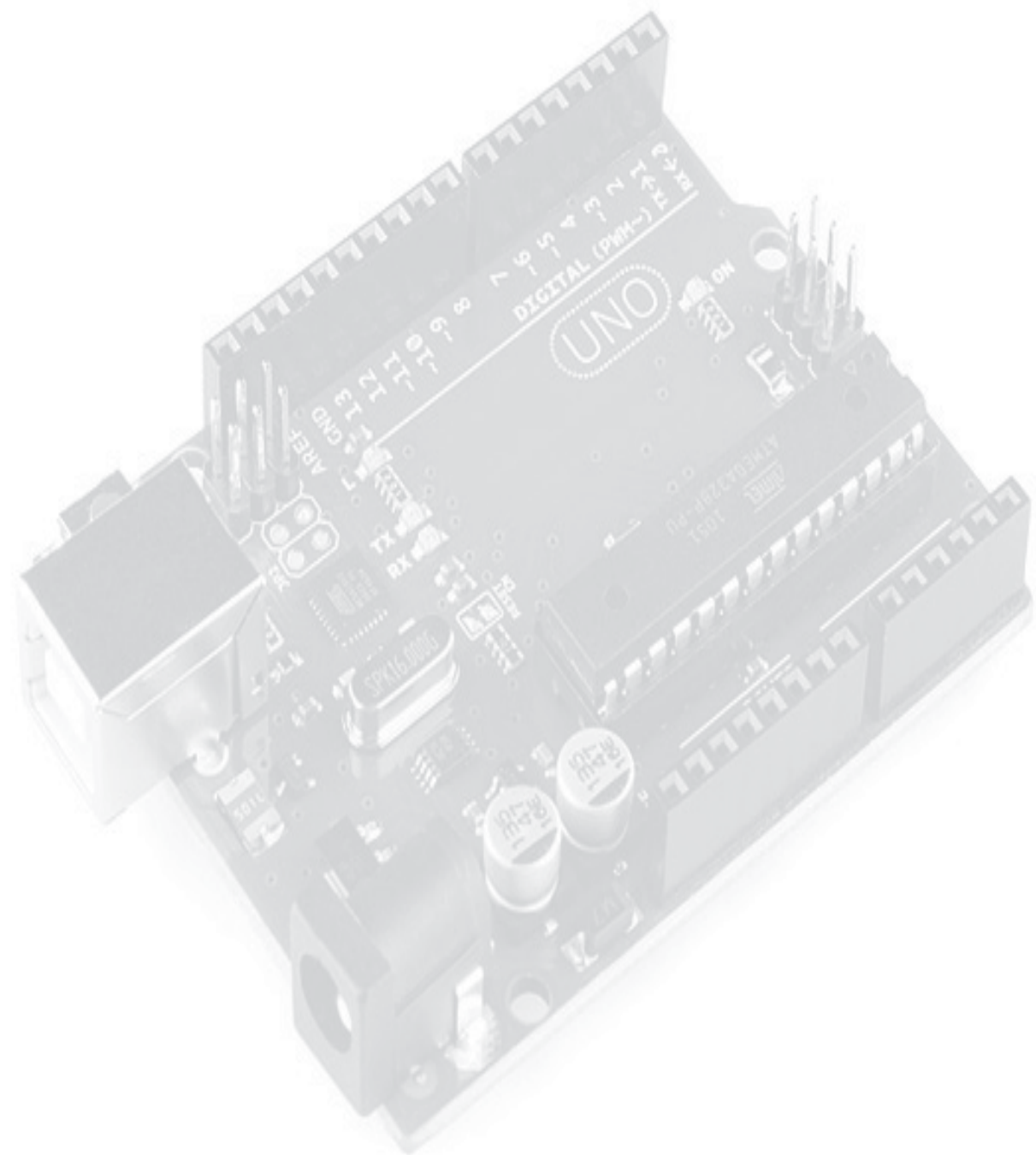
Cilj ovog primera je da implementira sistem koji je u interakciji sa korisnikom tako što se informacije razmenjuju u oba smera, od korisnika ka Arduinu i od Arduina ka korisniku. Kao medijum za prenos informacija se koristi Bluetooth veza.

Pin 13 digitalnog porta se koristi kao signalni pin za relej. U promenljivoj *RelayStatus* se čuva informacija o trenutnom stanju releja, tj. da li je relej uključen (HIGH) ili isključen (LOW). *String ReceivedString* služi za trenutno skladištenje primljenog stringa dok *String Password* čuva informaciju o aktivnoj šifri.

U funkciji *setup* vrše se već poznate komande za inicijalizaciju sistema. U glavnoj programskoj petlji se proverava da li je primljen neki podatak. Zatim se proverava da li je primljeni podatak identičan fiksnoj šifri i ukoliko jeste, poziva se funkcija *SwitchRelay* koja ima zadatak da promeni trenutno stanje releja i da o tome obavesti korisnika putem *Bluetooth* terminala na telefonu. Ukoliko uneta šifra nije ispravna, obaveštava se korisnik putem terminala slanjem određenih stringova.

11.2. ZADACI ZA SAMOSTALNI RAD

1. Realizovati Arduino sistem za kontrolu LE diode putem bluetooth-a. Sistem se sastoji od LCD displeja, bluetooth modula, releja i LE diode. Putem bluetooth terminala, na mobilnom telefonu se šalje poruka na Arduino hardversku platformu čija je sadržina „DON“. Nakon primljene poruke i proverenog sadržaja se uključuje dioda i ostaje uključena 10 sekundi. Za vreme dok je dioda uključena se ignorišu sve ostale poruke koje mogu pristići putem bluetooth modula. Nakon isteka vremenskog intervala dioda se isključuje i Arduino je spreman da primi novu poruku koja uključuje diodu na isti način. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.
2. Realizovati Arduino sistem koji se sastoji od diode, tastera, mreže releja i bluetooth modula. Sistem treba da ima opciju manuelnog uključivanja i isključivanja diode uz pomoć tastera kada je sistem u manuelnom režimu rada. Sistem je moguće prebaciti i u automatski režim rada slanjem poruke „Auto“ putem bluetooth modula. Takođe, korisnik može da zahteva prelazak u manuelni mod porukom „Manual“. Kada je sistem u automatskom režimu, pritisak na taster nema efekat, već se dioda uključuje ili isključuje putem poruka „ON“ i „OFF“ pristiglih sa bluetooth modula. Na početku rada, sistem treba da bude u manuelnom modu. Ukoliko se sistem nalazi u manuelnom modu, a korisnik pošalje poruku za uključivanje ili isključivanje diode, sistem treba da obavesti korisnika da je aktivan manuelni mod i da ga pita da li želi da pređe u automatski mod. Ukoliko je odgovor „DA“, sistem treba da pređe u automatski mod. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.



12. KONAČNE MAŠINE STANJA

Konačne mašine stanja su poseban mehanizam za rešavanje složenih problema u svetu ugrađenih sistema. Mašine stanja nisu mašine u pravom smislu te reči (na primer mašina za kucanje teksta, mašina za sortiranje, itd.), već predstavljaju jedan apstraktan koncept koji omogućava sistematski pristup rešenju problema u svetu programiranja (a naročito u domenu pisanja softvera za mikrokontrolere).

Konačnu mašinu stanja čine spisak konačno mnogo različitih radnih stanja u kojima sistem može da se nađe, definicija funkcionalnosti sistema za svako od tih stanja (ono što sistem treba da radi kada se nalazi u tom stanju), kao i uslovi prelaska iz jednog stanja u drugo. Ovakva specifikacija mašine stanja može biti data grafički ili u vidu tabele. Ovako specificirana mašina stanja u potpunosti definiše funkcionalnost jednog sistema, a prevodi se u softverski kod na dosta jednostavan način. Rad sa mašinama stanja biće ilustrovan kroz konkretne primere.

12.1. PRIMERI

Da bi bolje razumeli sam princip dekompozicije praktičnih problema u svrhu primene mašine stanja, pokušaćemo da sledeće primere rešimo pomoću mehanizma mašine stanja.

Primer 12.1. – Kontrola diode tasterom

- *Ralizovati Arduino sistem koji se sastoji od jedne diode i tastera, gde se pritiskom na taster vrši promena stanja diode . Napisati kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenja:

Za realizaciju sistema neophodne su sledeće komponente:

- Arduino UNO
- Dva otpornik od 220Ω
- Dioda
- Protobord
- 7 MM kratkospajča
- Taster

Električna šema sistema koja ilustruje način povezivanja komponenti, kao i izgled fizičke realizacije sistema u Arduino okruženju prikazani su na slici 5 - 8 (šema je ista kao u primeru 5.1.).

Ovo rešenje se od rešenja primera 5.1. razlikuje u tome što sada pokušavamo da sistem realizujemo primenom mehanizma konačne mašine stanja.

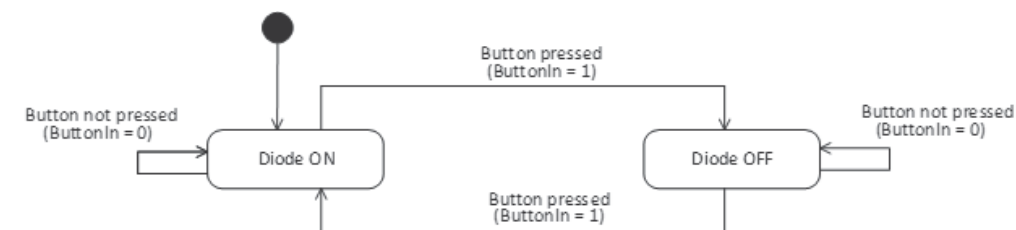
Kada se problemi rešavaju upotrebom mašine stanja, neophodno je identifikovati sva stanja mašine, kao i neophodne aktivnosti sistema u tim stanjima i uslove prelaska među stanjima.

Dva stanja u kojima sistem može da se nađe su da dioda bude uključena ili da bude isključena. Kada je sistem u prvom stanju, neophodno je na odgovarajući digitalni pin Arduina upisati „1“, a u drugom stanju „0“. Uslov promene stanja je da taster bude pritisnut. Tabelarno opisana mašina stanja može se predstaviti kao u tabeli 12 - 1.

Tabela 12 - 1 - Tabelarni opis mašine stanja

Trenutno stanje	Događaj	Novo stanje	Aktivnost
Diode ON	Taster nije pritisnut	Diode ON	Na digitalnom pinu diode je „1“
	Taster pritisnut	Diode OFF	
Diode OFF	Taster pritisnut	Diode ON	Na digitalnom pinu diode je „0“
	Taster nije pritisnut	Diode OFF	

Umesto tabelarno, mašine stanja se češće prikazuju grafički. Grafički prikaz mašine stanja opisane u tabeli 12 - 1 prikazan je na slici 12 - 1.



Slika 12 - 1 - Mašina stanja iz primera 12.1.

Dijagram prelaza se sastoji od stanja i uslova usled koga se prelazi u naredno stanje. Stanje je upisano u pravougaonik sa zaobljenim temenima, dok se uslov prelaza upisuje pored strelice koja označava prelazak u naredno stanje. Inicijalno stanje (stanje u kome se nalazi sistem pri pokretanju) označava se ispunjenim kružićem i strelicom koja ide od kružića ka inicijalnom stanju.

Inicijalno, sistem se nalazi u stanju **DiodeON**. U ovom stanju dioda je uključena. Sve dok se ne pritisne taster, tj. sve dok je vrednost promenljive *ButtonIn* jednaka 0, dioda je uključena i sistem ostaje u stanju **DiodeON**. Kada se pritisne taster, tj. kada *ButtonIn* dobije vrednost „1“, sistem prelazi u stanje **DiodeOFF** u kome je dioda isključena. U ovom stanju se ostaje sve dok je vrednost promenljive *ButtonIn* jednaka „0“. Kada se ponovo pritisne taster i vrednost promenljive *ButtonIn* postane jednaka „1“, sistem se vraća u stanje **DiodeON** i na izlazu **DiodeOut** se generiše „1“ i dioda se uključuje.

Na osnovu iznetog objašnjenja dijagrama stanja, neophodno je napisati kod koji implementira ovu mašinu stanja. Za definisanje stanja unutar koda, koristi se nabrojivi tip podataka enum. Kod koji implementira opisanu mašinu stanja je sledeći:

```
const int DiodePinNo = 13;
const int ButtonPinNo = 2;
volatile boolean ButtonIn = false;
typedef enum {DiodeOn, DiodeOff} MachineStates;
MachineStates CurrentState;
void ButtonISR(){
    ButtonIn = true;
}
void setup() {
    pinMode(DiodePinNo, OUTPUT);
    pinMode(ButtonPinNo, INPUT);

    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
    CurrentState = DiodeOn;
}
void loop() {
    switch(CurrentState){
        case DiodeOn:
            digitalWrite(DiodePinNo, HIGH);
            if(ButtonIn == true){
                ButtonIn = false;
                CurrentState = DiodeOff;
            }
            break;
        case DiodeOff:
            digitalWrite(DiodePinNo, LOW);
            if(ButtonIn == true){
                ButtonIn = false;
                CurrentState = DiodeOn;
            }
            break;
    }
}
```

Nabrojivi tip podataka *MachineStates* definiše stanja sistema. *CurrentState* predstavlja promenljivu ovog tipa i označava u kom stanju se trenutno nalazi mašina stanja. Promenljiva *ButtonIn* nosi informaciju o tome da li je dugme pritisnuto ili nije. Unutar *setup* funkcije vrši se inicijalizacija sistema na već poznat način. Razlika je u tome što sada unutar ove funkcije definišemo i početno stanje sistema, a to je *DiodeOn*.

U glavnoj programskoj petlji *loop* je implementirana mašina stanja pomoći *switch-case* strukture. Ova struktura ima zadatak da dovede sistem u stanje koje odgovara sadržaju promenljive *CurrentState*, odnosno trenutnom stanju sistema.

Ukoliko je sistem u stanju *DiodeOn* i vrednost promenljive *ButtonIn* nije jednaka true, dioda se drži uključenom. Ukoliko je vrednost promenljive *ButtonIn* jednaka true, resetuje se vrednost ove promenljive i sistem prelazi u stanje *DiodeOff* tako što se promenljivoj *CurrentState* dodeljuje vrednost *DiodeOff*.

Ukoliko se sistem nalzi u stanju *DiodeOff* i vrednost promenljive *ButtonIn* nije jednaka true, odnosno taster nije pritisnut, dioda se drži isključenom tako što se na pin 13 upisuje „0“. Ukoliko dođe do pritiskanja tastera dok smo u ovom stanju, vrednost promenljive *ButtonIn* će se postaviti na true, što za posledicu ima odlazak sistema u stanje *DiodeOn*.

Primer 12.2. Detekcija sekvence

- *Potrebno je realizovati Arduino sistem koji detektuje sekvencu pritiska tastera i ukoliko detektuje ispravnu sekvencu uključuje zelenu diodu dok je u suprotnom uključena crvena dioda. Sistem treba da detektuje sledeću sekvencu:*

$2 - 2 - 1 - 2 - 1$

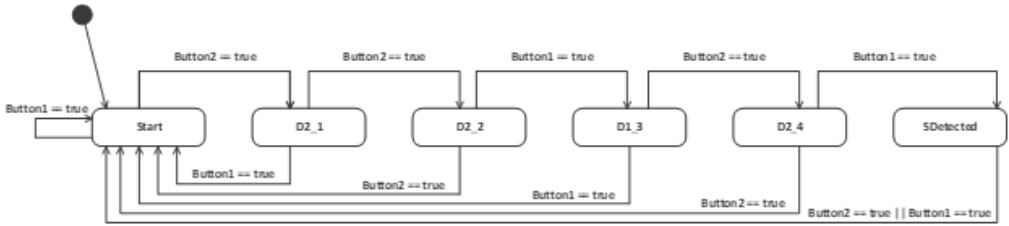
gde broj predstavlja broj tastera. Za vreme unošenja sekvence treba da bude aktivna žuta dioda. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

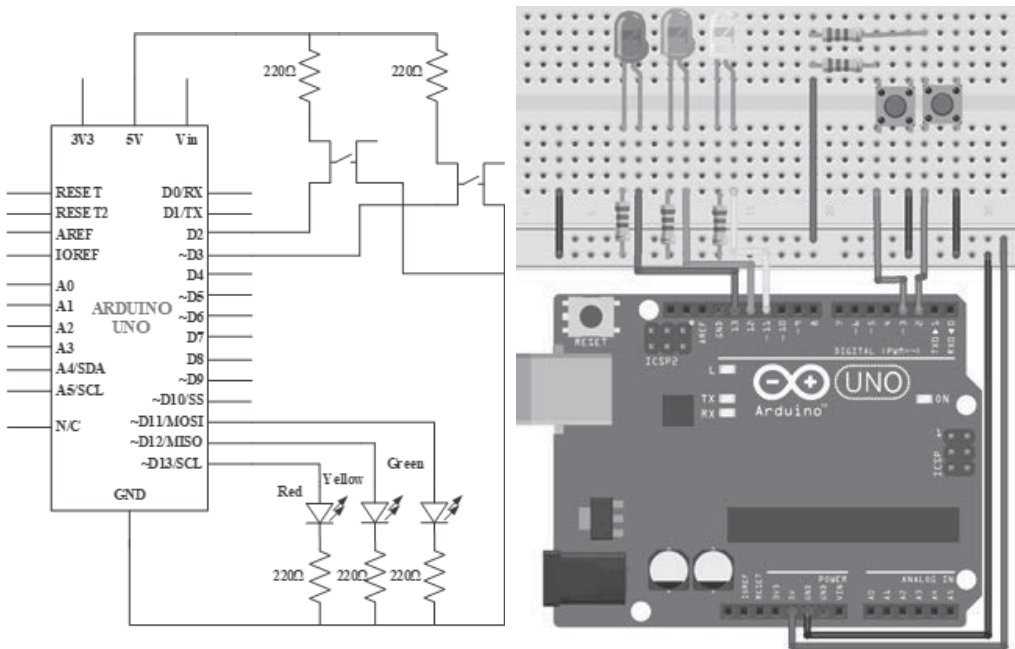
- Arduino UNO
- 5 otpornika od 220Ω
- 3 diode (crvena, žuta i zelena)
- Protobord
- 11 MM kratkospajča
- 2 tastera

Električna šema koja ilustruje način povezivanja, kao i fizička realizacija sistema u Arduino okruženju prikazani su na slici 12 - 3. Dijagram prelaza mašine stanja za ovaj sistem dat je na slici 12 - 2.



Slika 12 - 2 - Dojagram prelaza mašine stanja iz primera 12.2

Inicijalno se sistem nalazi u stanju **Start**. U ovom stanju sistem ostaje sve dok se ne pritisne taster broj 2. Kada se pritisne ovaj taster, mašina stanja prelazi u stanje **D2_1**, koje označava detekciju pritiska tastera broj 2. Iz ovog stanja se sistem vraća u stanje **Start** ukoliko korisnik pritisne taster broj 1. To znači da je došlo do prekida željene sekvence. Ukoliko u stanju **D2_1** korisnik pritisne taster 2 odlazi se u stanje **D2_2**. Ukoliko u ovom stanju korisnik pritisne taster 2, sistem se vraća u stanje **Start**. Ukoliko korisnik pritisne taster 1 dok se sistem nalazi u stanju **D2_2** nastavlja se sa ispravnom detekcijom sekvence i sistem odlazi u stanje **D1_3**. Ovom logikom se prelazi u naredno stanje ili se vraća u stanje **Start** u zavisnosti da li je pritisnut taster koji dovodi do ispunjavanja zadate sekvence.



Slika 12 - 3 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 12.2

Kod koji realizuje navedenu funkcionalnost je sledeći:

```
const int RedDiodePinNo = 13;
const int GreenDiodePinNo = 12;
const int YellowDiodePinNo = 11;
const int Button1PinNo = 2;
const int Button2PinNo = 3;
volatile boolean Button1 = false;
volatile boolean Button2 = false;
typedef enum {
    Start,
    D2_1,
    D2_2,
    D1_3,
    D2_4,
    SDetected
}MachineStates;
MachineStates CurrentState;
void Button1ISR(){
    Button1 = true;
}
void Button2ISR(){
    Button2 = true;
}
void setup() {
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT );
    pinMode(YellowDiodePinNo, OUTPUT);
    pinMode(Button1PinNo, INPUT);
    pinMode(Button2PinNo, INPUT);

    attachInterrupt(digitalPinToInterrupt(Button1PinNo), Button1ISR, FALLING);
    attachInterrupt(digitalPinToInterrupt(Button2PinNo), Button2ISR, FALLING);

    CurrentState = Start;
}

void loop() {
    switch(CurrentState){
        case Start:
            digitalWrite(RedDiodePinNo,HIGH);
            if(Button2 == true){
                Button2 = false;
                digitalWrite(RedDiodePinNo,LOW);
                CurrentState = D2_1;
            }
            break;
        case D2_1:
            if(Button2 == true){
```

```

        Button2 = false;
        CurrentState = D2_2;
    }
    else{
        if(Button1 == true){
            Button1 = false;
            digitalWrite(YellowDiodePinNo, LOW);
            CurrentState = Start;
        }
        else{
            digitalWrite(YellowDiodePinNo, HIGH);
        }
    }
}
break;
case D2_2:
    if(Button1 == true){
        Button1 = false;
        CurrentState = D1_3;
    }
    else{
        if(Button2 == true){
            Button2 = false;
            digitalWrite(YellowDiodePinNo, LOW);
            CurrentState = Start;
        }
        else{
            digitalWrite(YellowDiodePinNo, HIGH);
        }
    }
}
break;
case D1_3:
    if(Button2 == true){
        Button2 = false;
        CurrentState = D2_4;
    }
    else{
        if(Button1 == true){
            Button1 = false;
            digitalWrite(YellowDiodePinNo, LOW);
            CurrentState = Start;
        }
        else{
            digitalWrite(YellowDiodePinNo, HIGH);
        }
    }
}
break;
case D2_4:
    if(Button1 == true){
        Button1 = false;
        digitalWrite(YellowDiodePinNo, LOW);
    }

```

```

        CurrentState = SDetected;
    }
    else{
        if(Button2 == true){
            Button2 = false;
            digitalWrite(YellowDiodePinNo, LOW);
            CurrentState = Start;
        }
        else{
            digitalWrite(YellowDiodePinNo, HIGH);
        }
    }
}
break;
case SDetected:
    if((Button1 == true) || (Button2 == true) )
    {
        Button2 = false;
        Button1 = false;
        digitalWrite(GreenDiodePinNo, LOW);
        CurrentState = Start;
    }
    else{
        digitalWrite(GreenDiodePinNo, HIGH);
    }
}
break;
}
}

```

Primer 12.3. Sekvenca treperenja diode

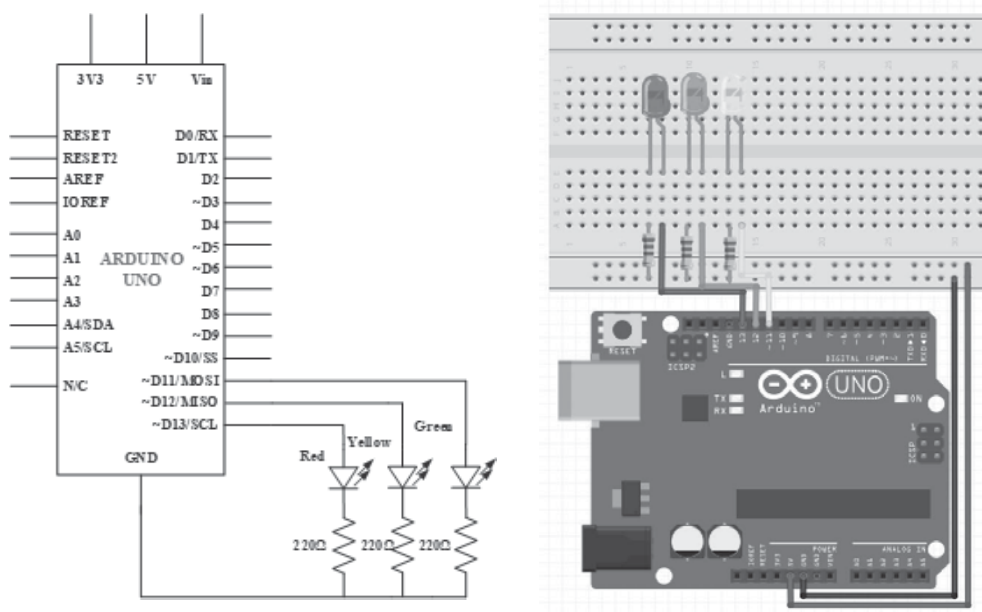
- *Realizovati Arduino sistem koji se sastoji od 3 diode (crvena, zelena i žuta) koje treba da se uključe i isključe na sledeći način: Prilikom pokretanja sistema crvena dioda treba da treperi 10 puta, dok druge dve diode ostaju ugašene. Perioda treperenja je 200ms. Nakon toga se crvena dioda deaktivira, a zatim zelena dioda menja stanje (treperi) sa istom periodom i isti broj puta. Na kraju se zelena dioda deaktivira, a zatim žuta dioda menja stanje sa istom periodom i isti broj puta kao i prethodne dve. Kada žuta dioda završi ciklus, deaktivira se a nakon toga se uključe sve tri diode i istovremeno menjaju stanje (trepere) 10 puta sa periodom od 100ms. Kada se završi ovaj ciklus sve tri diode se deaktiviraju i ponovo počinje da treperi crvena dioda i ciklus se ponavlja. Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.*

Rešenje:

Komponente koje su neophodne za realizaciju ovog sistema su:

- Arduino UNO
- 3 otpornika od 220Ω
- 3 diode (crvena, zelena, žuta)
- Protobord
- 6 MM kratkospajача

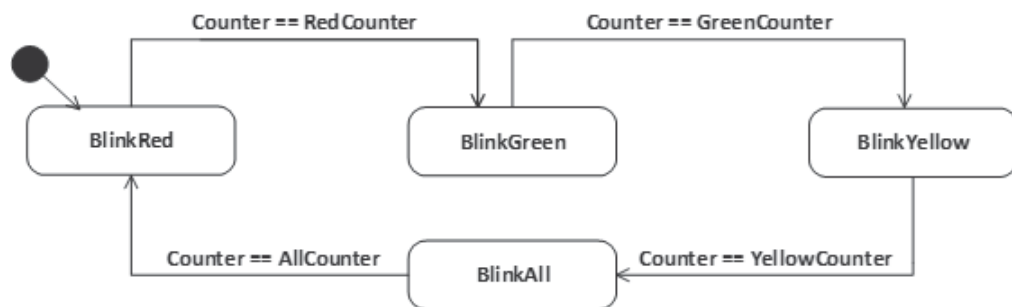
Električna šema sistema koja ilustruje način povezivanja komponenti i izgled fizičke realizacije sistema u Arduino okruženju prikazani su na slici 12 - 4.



Slika 12 - 4 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 12.3

Za realizaciju sistema korišćen je model mašine stanja. U sistemu postoji tajmer, čija je inicijalna perioda 200ms. Svaki put kada istekne perioda tajmera vrednost promenljive Counter se uveća za 1 i promeni se stanje jedne ili više dioda. Koja dioda će promeniti stanje zavisi od trenutnog stanja u kome se sistem nalazi. Dijagram prelaza mašine stanja koja implementira funkcionalnost sistema dat je na slici 12 - 5.

Inicijalno je sistem u stanju *BlinkRed* u kome treperi crvena dioda. Svaki put kada istekne perioda tajmera, unutar ovog stanja se vrši provera da li je vrednost brojača *Counter* dostigla potreban broj *RedCounter* koji predstavlja broj promena stanja crvene diode. Ukoliko ovaj broj nije dostignut, ostaje se u trenutnom stanju gde se menja stanje crvene diode. Ukoliko je ovaj broj dostignut, resetuje se vrednost brojača Counter i prelazi se u naredno stanje.



Slika 12 - 5 - Dijagram prelaza mašine stanja iz primera 12.3

Logika unutar stanja **BlinkRed**, **BlinkGreen** i **BlinkYellow** je ista izuzev toga što se pred ulazak u **BlinkAll** stanje menja perioda tajmerskog prekida sa 200ms na 100ms. Kod koji implementira opisanu funkcionalnost je:

```

#include <TimerOne.h>
const int RedDiodePinNo = 11;
const int GreenDiodePinNo = 12;
const int YellowDiodePinNo = 13;
int RedDiodeStatus = LOW;
int GreenDiodeStatus = LOW;
int YellowDiodeStatus = LOW;
const int RedCount = 10;
const int GreenCount = 10;
const int YellowCount = 10;
const int AllCount = 10;
volatile int Counter = 0;
unsigned long DiodeBlinkPeriod = 100000;
unsigned long DiodeAllBlinkPeriod = 50000;
volatile boolean TimerFlag = false;
//Machine states
typedef enum{
    BlinkRed,
    BlinkGreen,
    BlinkYellow,
    BlinkAll
}States;
States CurrentState;
void TimerISR(){
    TimerFlag = true;
}
//System is initialized here
void setup() {
    //Init diode pins as output pins
    pinMode(RedDiodePinNo, OUTPUT);
    pinMode(GreenDiodePinNo, OUTPUT);

```

```

pinMode(YellowDiodePinNo, OUTPUT);
CurrentState = BlinkRed;
Timer1.initialize(DiodeBlinkPeriod);
Timer1.attachInterrupt(TimerISR);
}
void loop(){
    switch(CurrentState){
        case BlinkRed:
            if(Counter == RedCount){
                Counter = 0;
                CurrentState = BlinkGreen;
                digitalWrite(RedDiodePinNo, LOW);

            }
            else{
                if(TimerFlag == true){
                    TimerFlag = false;
                    Counter++;
                    RedDiodeStatus = !RedDiodeStatus;
                    digitalWrite(RedDiodePinNo, RedDiodeStatus);
                }
            }
            break;
        case BlinkGreen:
            if(Counter == GreenCount ){
                Counter = 0;
                CurrentState = BlinkYellow;
                digitalWrite(GreenDiodePinNo, LOW);
            }
            else{
                if(TimerFlag == true){
                    TimerFlag = false;
                    Counter++;
                    GreenDiodeStatus = !GreenDiodeStatus;
                    digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
                }
            }
            break;
        case BlinkYellow:
            if(Counter == YellowCount ){
                Counter = 0;
                CurrentState = BlinkAll;
                digitalWrite(YellowDiodePinNo, LOW);
                Timer1.setPeriod(DiodeAllBlinkPeriod);
            }
            else{
                if(TimerFlag == true){
                    TimerFlag = false;
                    Counter++;
                    YellowDiodeStatus = !YellowDiodeStatus;

```

```

        digitalWrite(YellowDiodePinNo, YellowDiodeStatus);
    }
}
break;
case BlinkAll:
    if(Counter == AllCount ){
        Counter = 0;
        CurrentState = BlinkRed;
        digitalWrite(RedDiodePinNo, LOW);
        digitalWrite(GreenDiodePinNo, LOW);
        digitalWrite(YellowDiodePinNo, LOW);
        Timer1.setPeriod(DiodeBlinkPeriod);
    }
    else{
        if(TimerFlag == true){
            TimerFlag = false;
            Counter++;
            RedDiodeStatus = !RedDiodeStatus;
            GreenDiodeStatus = !GreenDiodeStatus;
            YellowDiodeStatus = !YellowDiodeStatus;
            digitalWrite(RedDiodePinNo, RedDiodeStatus);
            digitalWrite(GreenDiodePinNo, GreenDiodeStatus);
            digitalWrite(YellowDiodePinNo, YellowDiodeStatus);
        }
    }
    break;
}
}

```

Primer 12.4. - Semafor

- *Neophodno je realizovati Arduino sistem koji simulira ponašanje semafora. Semafor se sastoji od tastera i dva tipa semafora. Jedan je semafor za vozila, koji ima tri svetla (crveno, žuto i zeleno), dok je drugi semafor za pešake koji ima 2 svetla (crveno i zeleno). Inicijalno, nakon uključenja semafora, uključeno je zeleno svetlo za vozila i crveno za pešake. U ovom stanju sistem ostaje minimum 10 sekundi. Nakon isteka 10 sekundi, usled pritiska na taster, može se uključiti zeleno svetlo za pešake. Zeleno svetlo za pešake se ne uključuje odmah nakon pritiska tastera, već je potrebno da nakon zelenog svetla za vozila bude uključeno žuto svetlo za vozila u trajanju od 3 sekunde, a zatim se uključuje crveno svetlo za vozila. Nakon što se uključi crveno svetlo za vozila, posle 2 sekunde se uključuje zeleno svetlo za pešake i traje 10 sekundi. Nakon isteka vremenskog intervala od 10 sekundi, uključuje se crveno svetlo za pešake. Posle vremenskog intervala od 1 sekunde, uključuje se žuto svetlo za vozila u trajanju od 3 sekunde, a zatim se uključuje zeleno svetlo za vozila. Sada se ceo ciklus ponavlja. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnosti praktično potvrditi uspešnost realizacije.*

Rešenje:

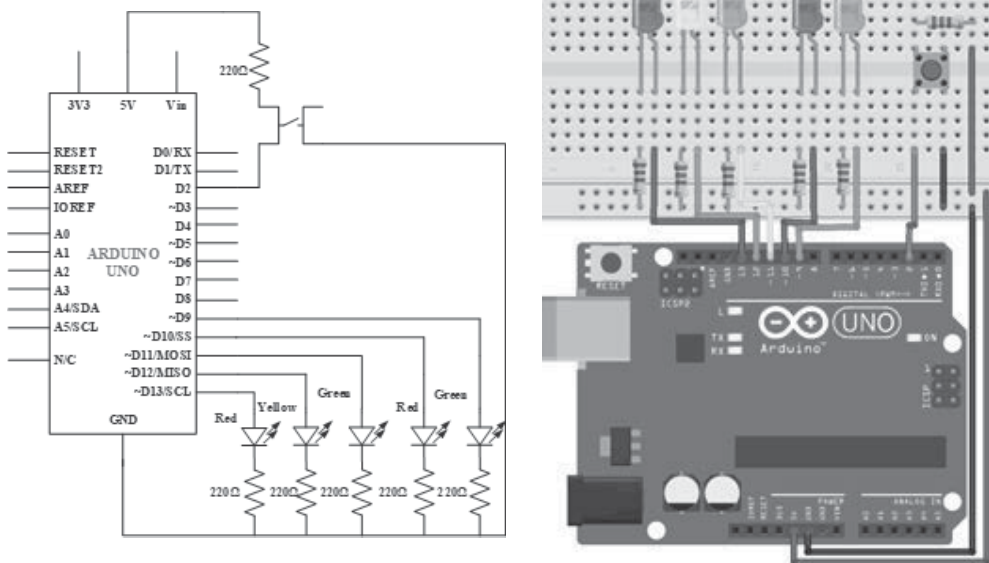
Za realizaciju sistema neophodne su sledeće komponente:

- Arduino UNO
- 6 Otpornika od 220Ω
- 5 Dioda (2 crvene, 2 zelene ,1 žuta)
- Protobord
- 10 MM kratkospajča
- 1 taster

Električna šema koja ilustruje način povezivanja navedenih komponenti kao i fizička realizacija sistema u Arduino okruženju prikazani su na slici 12 - 6.

Pre nego što krenemo u realizaciju koda sistema, neophodno je idejno osmisliti rešenje kroz mašinu stanja. Ako detaljnije analiziramo sistem, u cilju pronalaženja stanja u kojima se sistem može naći, primetićemo da postoje sledeće kombinacije dioda koje mogu biti aktivne u posmatranom trenutku:

- Zeleno za vozila, crveno za pešake (GV_RP)
- Žuto za vozila, crveno za pešake (YW_RP)
- Crveno za vozila, crveno za pešake (RV_RP)
- Crveno za vozila, zeleno za pešake (RV_GP)



Slika 12 - 6 - Električna šema (levo) i fizička realizacija (desno) sistema iz primera 12.4

Dakle, u sistemu postoje 4 stanja. U zagradi, pored značenja stanja, data je skraćenica koja će označavati dato stanje u nastavku rešenja.

Prilikom uključanja semafora, on se nalazi u stanju GV_RP, koje predstavlja inicijalno stanje. Ukoliko istekne perioda od 10 sekundi, i ako je pritisnut taster, prelazi se u stanje YV_RP u kome se ostaje 3 sekunde. Nakon isteka periode od 3 sekunde, sistem prelazi u stanje RV_RP u kome ostaje 1 sekund, a nakon isteka ove periode sistem prelazi u stanje RV_GP. Informaciju o tome da je istekla određena perioda čuvamo u promenljivoj *TimeReqSatisfied* dok informaciju o tome da je pritisnut taster čuvamo u promenljivoj *ButtonPressed*. Za sada postoji dovoljna količina informacija da se sastavi jedan deo dijagrama prelaza mašine stanja koji je prikazan na slici 12 - 7.



Slika 12 - 7 - Prvi deo dijagrama prelaza mašine stanja iz primera 12.4

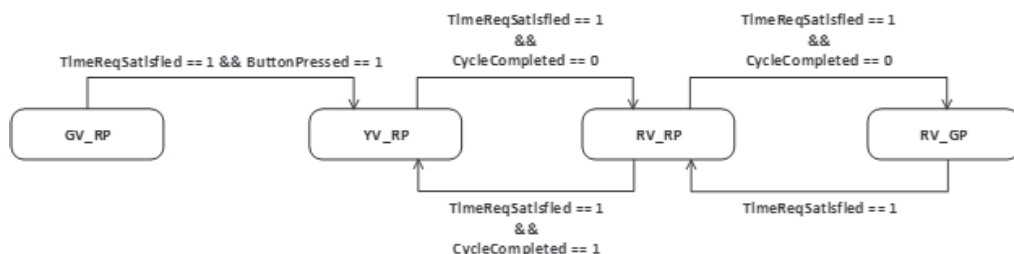


Slika 12 - 8- Drugi deo dijagrama prelaza mašine stanja iz primera 12.4

U Stanju RV_GP sistem ostaje 10 sekundi nakon čega se vraća u stanje RV_RP, kao što je prikazano na slici 12 - 8.

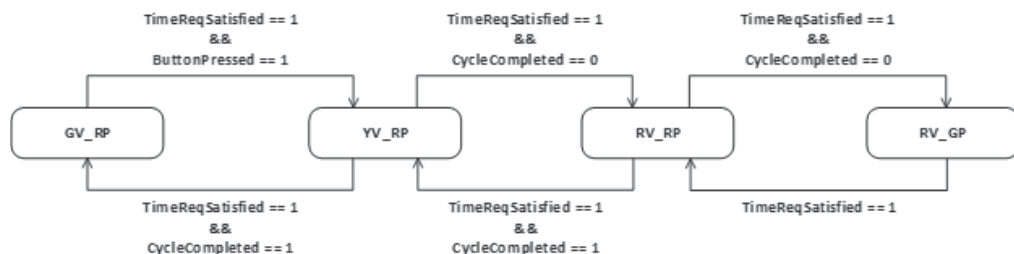
Nakon vraćanja u stanje RV_RP sistem treba u njemu da ostane 1sekund, i nakon toga bi trebao da se vrati u stanje YR_RP. Međutim, postavlja se pitanje na osnovu čega sistem zna da treba da ode u stanje YR_RP, a ne u stanje RV_GP?

Odgovor na ovo pitanje je da sistem ne može na osnovu informacija koje posedujemo proceniti u kom „smeru“ treba da ide, jer iz ovog stanja se može izaći na dva načina. Zato je potrebno da uvedemo dodatnu promenljivu (*flag*) koja će nositi informaciju o tome da li se sada iz stanja RV_RP vraćamo u RV_GP ili YV_RP. Ta promenljiva nosi naziv *CycleCompleted* i ima dve vrednosti 0 i 1. Kada je vrednost ove promenljive 0, idemo u smeru od zelenog svetla za vozila do zelenog svetla za pešake, dok se u suprotnom sistem kreće u suprotnom smeru. Nakon ove modifikacije, treći deo dijagrama prelaza izgleda kao na slici 12 - 9.



Slika 12 - 9 - Treći deo dijagrama prelaza mašine stanja iz primera 12.4

Sada, kada se mašina stanja nađe u stanju RV_RP ima dovoljno informacija da „zna“ odakle je došla u to stanje i „kuda treba da ide“. Sledeći dosadašnje ideje neophodno je završiti dijagram prelaza. Nakon toga se dobija kompletan dijagram prelaza kao na slici 12- 10.



Slika 12 - 10 - Konačan dijagram prelaza mašine stanja iz primera 12.4

Nakon implementacije ideje, koju smo do sada izneli, dobijamo sledeći kod:

```

#include <TimerOne.h>
const int V_RedDiodePinNo = 13;
const int V_YellowDiodePinNo = 12;
const int V_GreenDiodePinNo = 11;
const int P_RedDiodePinNo = 10;
const int P_GreenDiodePinNo = 9;
const int ButtonPinNo = 2;
volatile boolean ButtonPressed = false;
volatile boolean TimeReqSatisfied = false;
boolean CycleCompleted;
const unsigned long GreenPeriod = 10000000;
const unsigned long YellowPeriod = 3000000;
const unsigned long V_RedPeriod = 1000000;
    
```

```

const unsigned long P_RedPeriod = 2000000;
typedef enum {
    GV_RP,
    YV_RP,
    RV_RP,
    RV_GP
} MachineStates;
MachineStates CurrentState;
void ButtonISR(){
    ButtonPressed = true;
}
void TimerISR(){
    TimeReqSatisfied = true;
}
void setup() {
    pinMode(V_RedDiodePinNo, OUTPUT);
    pinMode(V_GreenDiodePinNo, OUTPUT );
    pinMode(V_YellowDiodePinNo, OUTPUT);
    pinMode(P_RedDiodePinNo, OUTPUT);
    pinMode(P_GreenDiodePinNo, OUTPUT );
    attachInterrupt(digitalPinToInterrupt(ButtonPinNo), ButtonISR, FALLING);
    Timer1.initialize(GreenPeriod);
    Timer1.attachInterrupt(TimerISR);

    CurrentState = GV_RP;
    CycleCompleted = false;
}
void loop() {
    switch(CurrentState){
        case GV_RP:
            digitalWrite(V_RedDiodePinNo, LOW);
            digitalWrite(V_GreenDiodePinNo, HIGH );
            digitalWrite(V_YellowDiodePinNo, LOW);
            digitalWrite(P_RedDiodePinNo,HIGH);
            digitalWrite(P_GreenDiodePinNo, LOW );
            if((TimeReqSatisfied == true) && (ButtonPressed == true)){
                TimeReqSatisfied = false;
                ButtonPressed = false;
                CycleCompleted = false;
                CurrentState = YV_RP;
                Timer1.setPeriod(YellowPeriod);
            }
            break;
        case YV_RP:
            digitalWrite(V_RedDiodePinNo, LOW);
            digitalWrite(V_GreenDiodePinNo, LOW );
            digitalWrite(V_YellowDiodePinNo, HIGH);
            digitalWrite(P_RedDiodePinNo,HIGH);
            digitalWrite(P_GreenDiodePinNo, LOW );
            if((TimeReqSatisfied == true)){

```

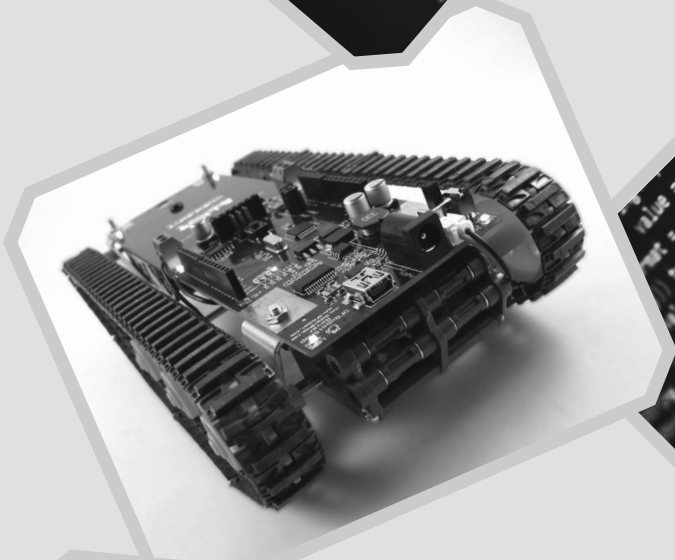
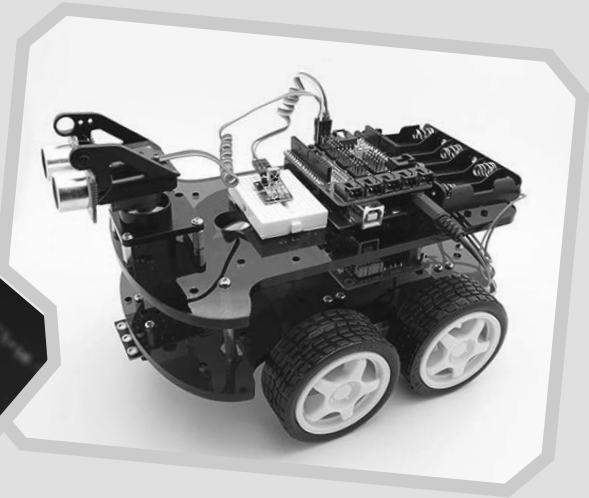
```

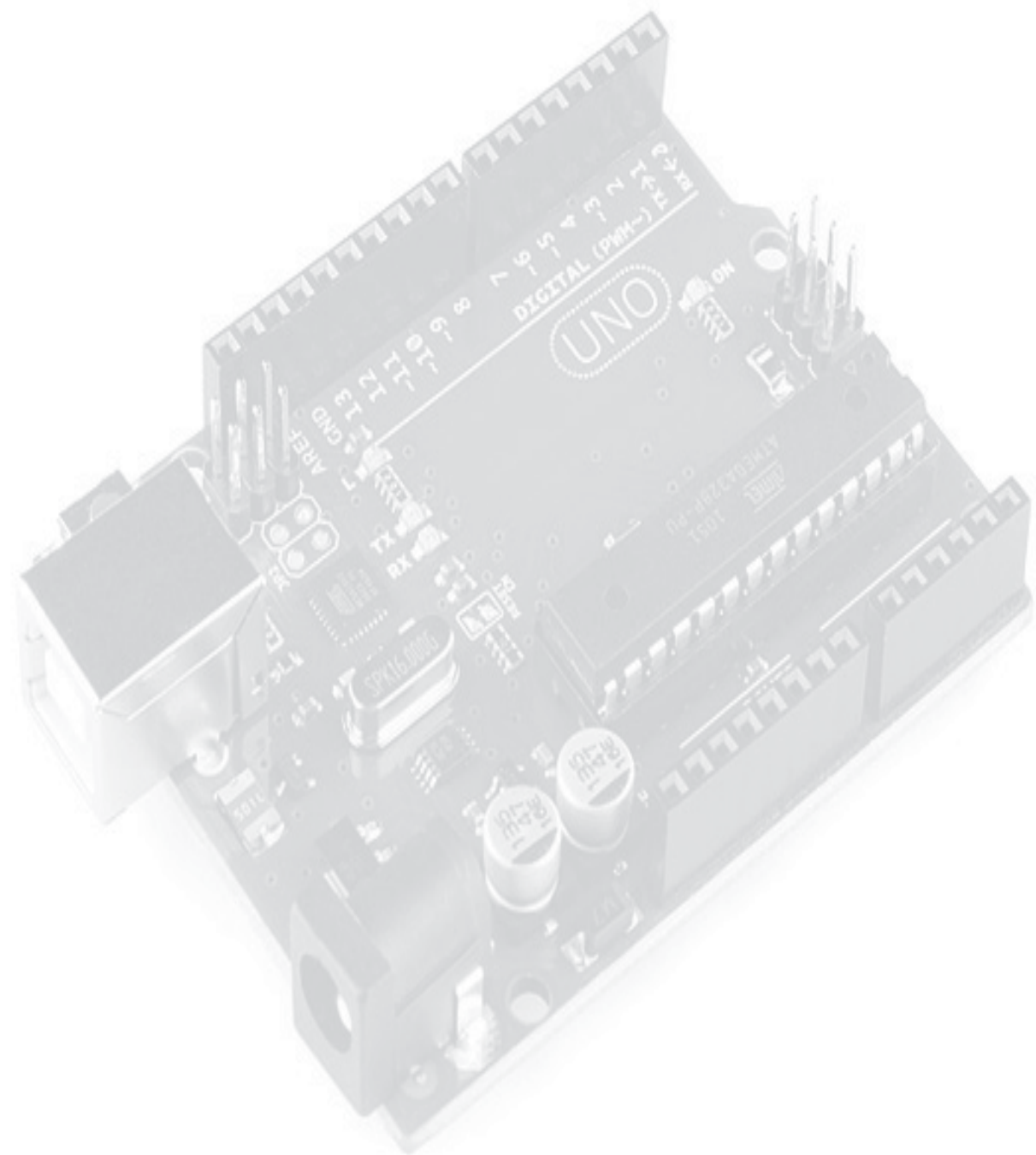
        TimeReqSatisfied = false;
        if(CycleCompleted == false){
            CurrentState = RV_RP;
            Timer1.setPeriod(V_RedPeriod);
        }
        else{
            CurrentState = GV_RP;
            Timer1.setPeriod(GreenPeriod);
        }
    }
    break;
case RV_RP:
    digitalWrite(V_RedDiodePinNo, HIGH);
    digitalWrite(V_GreenDiodePinNo, LOW );
    digitalWrite(V_YellowDiodePinNo, LOW);
    digitalWrite(P_RedDiodePinNo, HIGH);
    digitalWrite(P_GreenDiodePinNo, LOW );
    if((TimeReqSatisfied == true)){
        TimeReqSatisfied = false;
        if(CycleCompleted == false){
            CurrentState = RV_GP;
            Timer1.setPeriod(GreenPeriod);
        }
        else{
            CurrentState = YV_RP;
            Timer1.setPeriod(YellowPeriod);
        }
    }
    break;
case RV_GP:
    digitalWrite(V_RedDiodePinNo, HIGH);
    digitalWrite(V_GreenDiodePinNo, LOW );
    digitalWrite(V_YellowDiodePinNo, LOW);
    digitalWrite(P_RedDiodePinNo, LOW);
    digitalWrite(P_GreenDiodePinNo, HIGH );
    if((TimeReqSatisfied == true)){
        TimeReqSatisfied = false;
        CycleCompleted = true;
        CurrentState = RV_RP;
        Timer1.setPeriod(P_RedPeriod);
    }
    break;
}
}
}

```

12.1. ZADACI ZA SAMOSTALNI RAD

1. Primenom mašine stanja rešiti primer 10.3
2. Primenom mašine stanja rešiti primer 7.3
3. Potrebno je realizovati Arduino sistem koji implementira proširenu funkcionalnost semafora iz primera 12.4. Proširenje funkcionalnosti se ogleda u dodavanju još jednog tastera kojim se vrši prelazak semafora iz aktivnog u neaktivno stanje i obrnuto. Kada je semafor aktivan on funkcioniše na način opisan u primeru 12.4. Pritiskom na dodatni taster, vrši se deaktivacija semafora. Kada je semafor deaktiviran, treba da treperi žuto sveto za vozila sa periodom od 0.3 sekunde, dok se semafor za pešake isključuje. Napisati kod za Arduino Uno hardversku platformu i praktično potvrditi uspešnost realizacije.
4. Potrebno je realizovati Arduino Uno sistem koji realizuje igricu „Zaobiđi prepreku“. Sistem se sastoji od LCD displeja i dva tastera. Inicijalno se na početku prvog reda LCD displeja iscrtava čovečuljak koji ne sme da udari u prepreku. Prepreke predstavljaju ispunjeni kvadratići koji čovečuljku idu u susret. Tokom trajanja igrice, čovečuljak je pozicioniran na početku prvog ili drugog reda, a promena reda se vrši pritiskom na odgovarajući taster. Prepreke se generišu na suprotnoj strani LCD displeja (posmatrano u odnosu na čovečuljka) i kreću se ka čovečuljku. Prepreke mogu biti generisane u proizvoljnom redu i proizvoljnom trenutku vremena. Nakon isteka vremenskog intervala od 30 sekundi, brzina kretanja prepreka se uvećava. Igrica traje sve dok se ne izgube 2 života. Napisati kod za Arduino Uno hardversku platformu i praktično potvrditi uspešnost realizacije.





13. REALIZACIJA KORISNIČKIH BIBLIOTEKA

Većina biblioteka koje su korišćene kroz ovaj priručnik su preuzete sa interneta i korišćene su u cilju pojednostavljenja rada sa pojedinim modulima. Za većinu modula postoje već gotove biblioteke, koje se takođe mogu preuzeti sa interneta, a koje nisu korišćene kroz ovaj priručnik. Međutim, moduli se unapređuju, interfejsi se menjaju i usled toga se menjaju i biblioteke. Projektant Arduino sistema mora posedovati dovoljno znanja da se prilagodi ovim promenama i da ponekad, usled pojednostavljenja sistema koji realizuje, napravi i svoje biblioteke. Cilj ovog poglavlja je da čitaoca upozna sa načinom kreiranja biblioteka.

Da bi mogli da kreiramo biblioteke za Arduino module, neophodno je podsetiti se fajlovske strukture koda pisanog na programskom jeziku C++ i dobro razumeti hardversku realizaciju modula za koji kreiramo biblioteku.

Svaki kod pisan u programskom jeziku C++ može se pisati u jednom fajlu sa ekstenzijom *.cpp*. Ovaj fajl se naziva *source* fajl. Međutim, kada se pišu složeniji programi, broj linija koda *source* fajla postaje prevelik, što smanjuje čitljivost i razumljivost samog koda. Radi bolje čitljivosti se iz *source* fajla mogu izdvojiti deklaracije funkcija i pojedinih promenljivih i smestiti u *header* fajl.

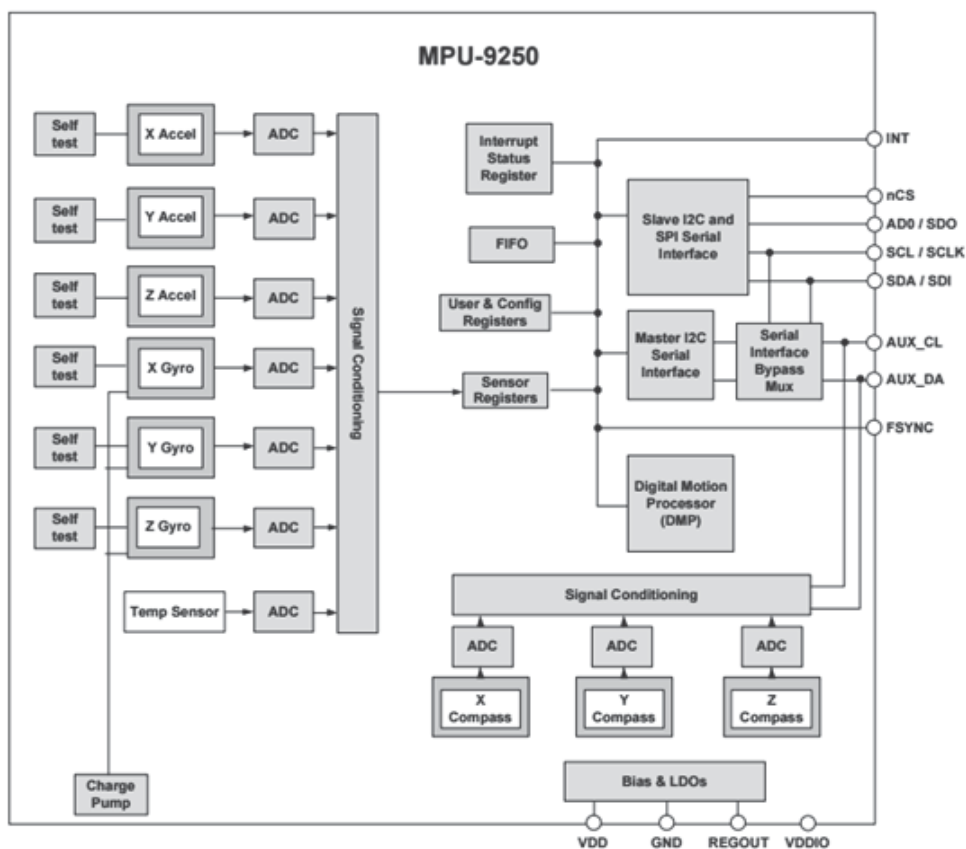
Header fajlovi imaju ekstenziju *.h*. Svaki *header* fajl ima odgovarajući *source* fajl u kome se nalaze definicije funkcija i deklaracije promenljivih koje su, najšire posmatrano, vidljive samo u tom fajlu. Ako se neka promenljiva deklarise kao globalna unutar nekog C++ fajla, ona nije vidljiva izvan tog C++ fajla, već predstavlja globalnu promenljivu samo za taj C++ fajl.

Program može sadržati i veći broj *header* fajlova. Za funkcije koje su deklarisanе unutar *header* fajla potrebno je da postoje odgovarajući *.cpp* fajlovi u kojima su te funkcije definisane. Da bi kompajler, u trenutku prevođenja, mogao da nađe deklaracije funkcija neophodno je na početku svakog *.cpp* fajla navesti *header* fajl u kome je funkcija deklarisanа. Ovo se postiže direktivom *#include* koja kao argument prima naziv fajla.

Nakon što smo se podsetili strukture fajlova, neophodno je da se upoznamo sa hardverskom realizacijom modula za koji kreiramo Arduino biblioteku. U tu svrhu se koristi dokument koji se naziva *DataSheet* modula (nekada se koristi i naziv *UserGuide*). Cilj ovog dokumenta je da korisniku sistema omogući uvid u hardversku realizaciju korišćene komponente i često se može naći na web stranici proizvođača modula. Kroz ovo poglavlje koristi se modul *MPU-9250/6500* koji na sebi poseduje akcelerometar, žiroskop, senzor temperature i magnetometar. *DataSheet* ovog modula se može naći na internetu [16].

13.1. HARDVERSKA REALIZACIJA MPU-9250 MODULA

Sistem se sastoji od tri senzora ubrzanja (žiroskopa i akcelerometra), koji su raspoređeni duž tri ose, jednog senzora temperature i tri senzora za detekciju magnetnog polja koji su takođe raspoređeni duž tri ose. Međutim, u ovom poglavlju se samo obrađuje mogućnost ovog modula da se koristi kao akcelerometar. Da bi bilo moguće realizovati biblioteku koja će abstrahovati pristup akcelerometru ovog modula, najpre je neophodno prostudirati hardversku realizaciju modula.



Slika 13 - 1 - Blok šema modula MPU_9250 [16]

Na osnovu slike 13 - 1 se može bolje razumeti kako zapravo izgleda sistem iznutra. Postoje 3 senzora ubrzanja koja su raspoređena duž X, Y i Z ose. Ovi senzori na svom izlazu daju veličinu koja je srazmerna ubrzanju koje deluje na njih.

Ubrzanje se inače izražava u m/s^2 ili u veličini koja se označava sa g , a iznosi $9.81 m/s^2$. (ubrzanje zemljine teže) Na osnovu ovoga, može se zaključiti da ukoliko se senzor postavi na apsolutno ravnu površinu, na izlazu senzora X i Y će biti 0 dok će na izlazu senzora Z biti vrednost napona srazmerna vrednosti g . Senzori ubrzanja koji se koriste u ovom modulu imaju rezoluciju od $\pm 2g$ do $\pm 8g$ koja se može konfigurisati u zavisnosti od potreba sistema koji se realizuje. Izlazi ovih senzora se vode na analogno digitalne konvertore koji imaju rezoluciju od 16 bita. Rezultati konverzije se smeštaju u odgovarajuće registre kojima se može pristupiti putem I2C magistrale.

Da bi bilo moguće pristupiti ovom modulu, i čitati izmerene vrednosti, putem I2C magistrale, najpre je neophodno znati adresu uređaja kojem pristupamo, a potom i adrese registara unutar kojih se smeštaju podaci. Adresa uređaja služi za njegovu identifikaciju u slučaju da u sistemu postoji više senzora. Registrima se može pristupiti u cilju čitanja nekih podataka sa senzora (npr. merene veličine), ali takođe i u cilju podešavanja njihovih vrednosti (tzv. konfiguracija senzora).

Nakon što identifikujemo adresu uređaja na I2C magistrali neophodno je u *DataSheetu* pronaći adrese registara koji su nam neophodni da bi uspešno pročitali vrednosti od interesa. U primeru koji će biti dat nešto kasnije, od interesa su ubrzanja duž x, y i z ose. Zbog toga su relevantni registri koji sadrže rezultate analogno digitalne konverzije signala sa akcelerometra, kao i registri koji služe za njihovu konfiguraciju dati u tabeli 13 - 1.

Tabela 13 - 1 - Adrese registara od interesa unutar modula MPU_9250

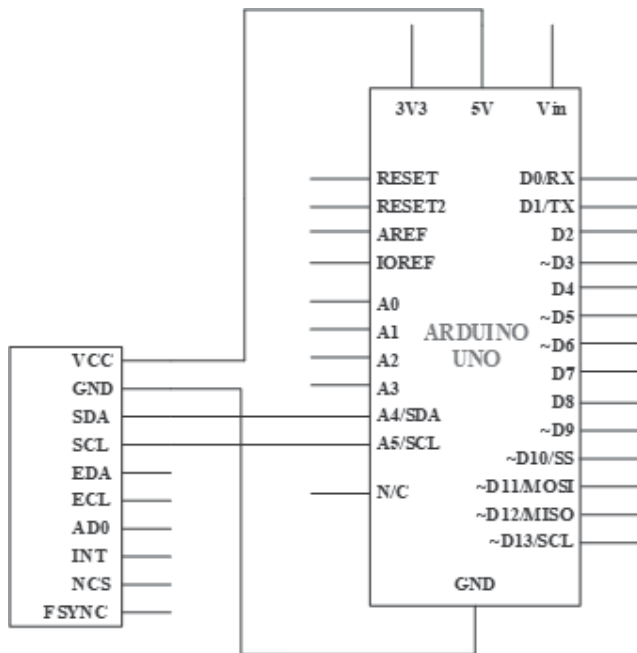
Naziv registra	Adresa	Značenje registra
/	0x68	Adresa uređaja
ACCEL_XOUT_H	0x3B	Viši bajt podatka 16tobitnog AD konvertora sa senzora ubrzanja na X osi
ACCEL_XOUT_L	0x3C	Niži bajt podatka 16tobitnog AD konvertora sa senzora ubrzanja na X osi
ACCEL_YOUT_H	0x3D	Viši bajt podatka 16tobitnog AD konvertora sa senzora ubrzanja na Y osi
ACCEL_YOUT_L	0x3E	Niži bajt podatka 16tobitnog AD konvertora sa senzora ubrzanja na Y osi
ACCEL_ZOUT_H	0x3F	Viši bajt podatka 16tobitnog AD konvertora sa senzora ubrzanja na Z osi
ACCEL_ZOUT_L	0x40	Niži bajt podatka 16tobitnog AD konvertora sa senzora ubrzanja na X osi
ACCEL_CONFIG	0x1C	Konfiguracioni registar za konfiguraciju akcelerometra

Konfiguracioni registra ACCEL_CONFIG se koristi za podešavanje rezolucije senzora, tako što se menjaju vrednosti bita na poziciji 4 i 3 i to na sledeći način:

Tabela 13 - 2 - Značenje bita 3 i 4 u ACCEL_CONF registru

Rezolucija	Vrednosti bita 4 i 3 respektivno
$\pm 2g$	00
$\pm 4g$	01
$\pm 8g$	10
$\pm 16g$	11

Kada smo proučili hardversku realizaciju i upoznali se sa registrima koji su nam neophodni, potrebno je povezati Arduino hardversku platformu i modul putem I2C magistrale, kao što je prikazano na slici 13 - 2.



Slika 13 - 2 - Povezivanja modula MPU_9250 i Arduina

Nakon što se modul poveže sa Arduinoom putem I2C magistrale, iz DataSheet-a je neophodno proučiti protokol na osnovu koga se vrši komunikacija modula i Arduina. Pod protokolom podrazumevamo niz koraka koji se dešavaju na komunikacionoj magistrali u cilju sprovođenja neke akcije (konkretno, u našem slučaju, čitanja iz registra ili upis u registar).

Da bi upisali sadržaj u registar neophodno je proći kroz sledeće korake:

- Adresirati periferiju u modu upisa
- Upisati adresu registra u koji želimo da upišemo sadržaj
- Upisati sadržaj
- Završiti prenos i osloboditi magistralu

Da bi pročitali sadržaj iz registra neophodno je uraditi sledeće:

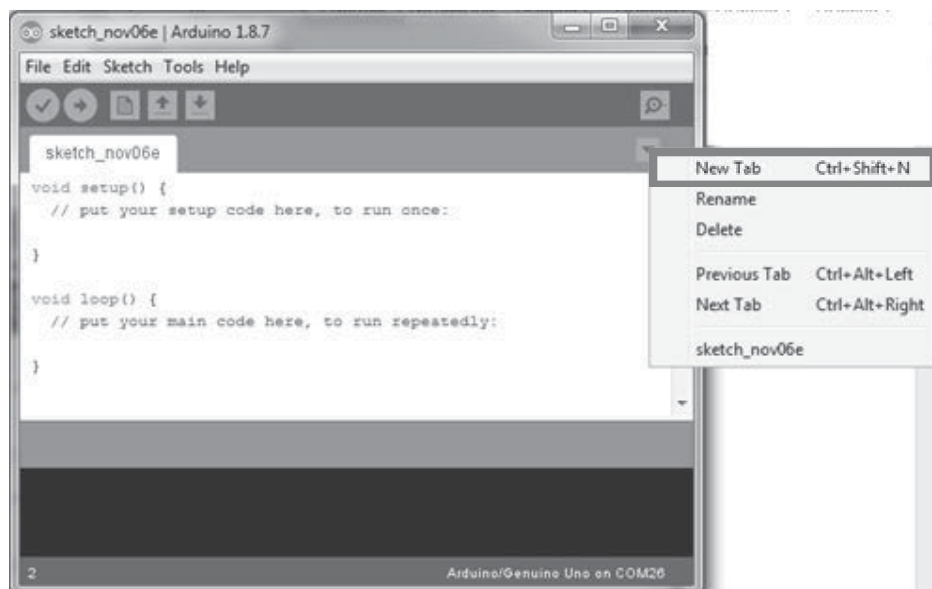
- Adresirati periferiju u modu upisa
- Upisati sadržaj iz koga želimo da pročitalo
- Završiti prenos
- Adresirati periferiju u cilju čitanja

Na bazi prethodnih razmatranja i informacija prikupljenih iz DataSheet-a moguće je kreirati biblioteku za rad sa opisanim modulom.

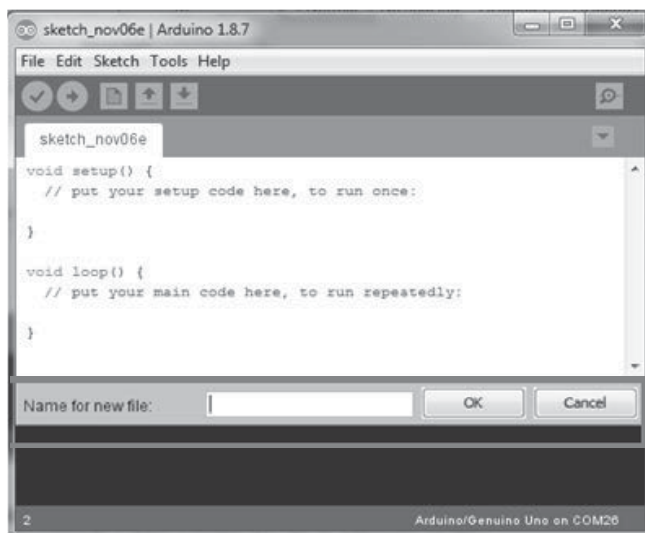
13.2. KREIRANJE I UKLJUČIVANJE BIBLIOTEKE

Cilj kreirane biblioteke je da abstrahuje pristup hardveru. Biblioteka predstavlja skup *.h* i *.c* (*.cpp*) fajlova koji implementiraju funkcije za očitavanje senzora ili upis konfiguracije na senzor. Ovi fajlovi se kreiraju na sledeći način:

Nakon što udjemo u Arduino IDE softversko okruženje kreiramo novi Arduino fajl, potrebno je kliknuti na srelicu na dole kao što je prikazano na slici 13 - 3.



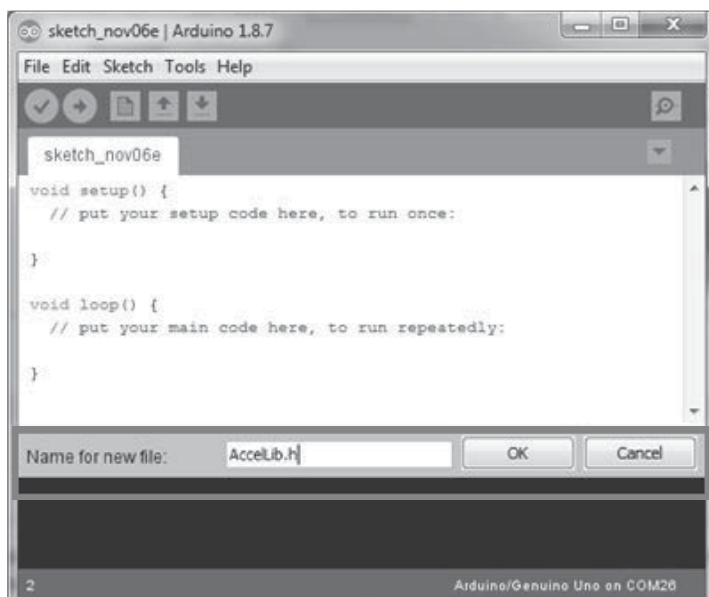
Slika 13 - 3 - Kreiranje novog fajla biblioteke



Slika 13 - 4 - Prozor za unos naziva fajla

Nakon što se otvori padajući meni, potrebno je kliknuti na New Tab nakon čega se otvara mali pravougaonik na žutoj pozadini kao što je prikazano na slici 13 - 4.

U pravougaonik je potrebno uneti naziv fajla sa ekstenzijom .h kao što je prikazano na slici 13 - 5 i nakon toga potvrditi unos klikom na dugme „OK“ .



Slika 13 - 5 - Potvrda naziva fajla



Slika 13 - 6 - Otvoren novi .h fajl

Nakon što smo potvrdili unos otvara se novi, prazan fajl, gde u kartici, na vrhu fajla, piše naziv fajla (Slika 13 - 6). Sadržaj header fajla koji ćemo razmatrati je sledeći:

```
#ifndef AccelLib_h
#define AccelLib_h
#include "Arduino.h"
#include "Wire.h"
//Define registers addresses
#define ACCEL_XOUT_H 0x3B
#define ACCEL_XOUT_L 0x3C
#define ACCEL_YOUT_H 0x3D
#define ACCEL_YOUT_L 0x3E
#define ACCEL_ZOUT_H 0x3F
#define ACCEL_ZOUT_L 0x40
#define ACCEL_CONFIG 0x1c
//define class
class MPU_9250{
public:
    // Init object of this class by Module Address
    MPU_9250(const uint8_t Address);
    // Read Acceleraton on X,Y and Z axes
    void ReadAccel(int16_t* X,int16_t* Y,int16_t* Z);
```

```
// Set Acceleration sensor resolution
// Possible options are 2,4,8,16
void SetAccelRes(uint8_t const Resolution);
private:
    uint8_t Module_Address;
};
```

Svaki header fajl treba da počinje sa predprocesorskom direktivom *#ifndef* koja sprečava višestruko uključivanje header fajlova. Nakon nje, ukoliko u fajlu ne postoji definisan header fajl sa istim imenom, sledi definisanje naziva heder fajla tako da ostatak sistema zna da heder fajl pod definisanim imenom već postoji.

Unutar ovog fajla se uključuje header fajl *Arduino.h* koji implementira osnovne Arduino funkcionalnosti. Ovo je obavezno ukoliko se definišu zasebni header fajlovi. Nakon *Arduino.h* fajla se uključuje *Wire.h* biblioteka koja implementira funkcije za korišćenje I2C protokola. Nakon toga, sledi definisanje adresa registara koje se mogu koristiti, kako u ovoj biblioteci, tako i u ostatku sistema ukoliko se uključi ovaj fajl.

Nakon što smo definisali adrese registara ovog modula, definiše se klasa *MPU_9250* unutar koje se deklarišu funkcije i promenljive koje će predstavljati interfejs ka modulu. Funkcija *MPU_9250* predstavlja konstruktor i kao argument prima *Address* koji predstavlja adresu modula. Dakle, ako želimo da koristimo funkcionalnosti ovog modula, najpre je neophodno kreirati objekat tipa *MPU_9250*, kome kao argument prosleđujemo *Addressu* modula.

Funkcija *ReadAccel* služi za čitanje ubrzanja sa sve tri ose X,Y i Z. Ova funkcija kao argument prima tri pokazivača na tip *int16_t* koji predstavljaju promenljive u koje će se smestiti pročitane vrednosti ubrzanja. Tip *int16_t* je intidžerski tip veličine dva bajta.

Funkcija *SetAccelRes* služi za podešavanje rezolucije akcelerometra. Kao argument prima vrednost *Resolution*. Ova vrednost predstavlja moguće vrednosti rezolucije senzora i to su 2, 4, 8 i 16. Ukoliko korisnik prosledi neku vrednost koja ne pripada ovom skupu, vrednost rezolucije će biti podešena na vrednost 2.

Ovim smo završili sa kodom koji se nalazi u header fajlu. Sada je potrebno sve funkcije, koje su deklarisanе u okviru header fajla, definisati u *.cpp* fajlu. Ovaj fajl se kreira na isti način kao i header fajl, samo je sada ekstenzija fajla *.cpp*. Kod kojim se definiše funkcionalnost deklarisanih funkcija iz header fajla je sledeći:

```
#include "AccelLib.h"
MPU_9250::MPU_9250(const uint8_t Address){
    Module_Address = Address;
    Wire.begin();
}
void MPU_9250::ReadAccel(int16_t* X,int16_t* Y,int16_t* Z){
    //Begin transmission to the Module_Address
    Wire.beginTransaction(Module_Address);
```

```

//Init transmission for read
Wire.write(ACCEL_XOUT_H);
//And transmission
Wire.endTransmission();
//Read 6 bytes of data
Wire.requestFrom(Module_Address,6);
while(Wire.available()){
    *X = Wire.read()<<8;
    *X |= Wire.read();
    *Y = Wire.read()<<8;
    *Y |= Wire.read();
    *Z = Wire.read()<<8;
    *Z |= Wire.read();
}
}
void MPU_9250::SetAccelRes(const uint8_t Resolution){
    uint8_t BinRes;
    switch(Resolution){
        case 2:
            BinRes = 0x00;
            break;
        case 4:
            BinRes = 0x08;
            break;
        case 8:
            BinRes = 0x10;
            break;
        case 16:
            BinRes = 0x18;
            break;
        default:
            BinRes = 0x00;
            break;
    }
    Wire.beginTransaction(Module_Address);
    Wire.write(ACCEL_CONFIG);
    Wire.write(BinRes);
    Wire.endTransmission();
}

```

Da bi započeli proces na I2C magistrali koristimo funkciju

Wire.beginTransmission(address)

Address – sedmobitna adresa uređaja

Ova funkcija je praćena funkcijom

Wire.write(data)

data – Adresa registra

koja služi da se upiše adresa registra iz koga čitamo ili upisujemo

Nakon ove funkcije sledi funkcija

Wire.endTransmission()

koja završava ciklus upisa ili čitanja.

Ukoliko želimo da pročitamo sadržaj iz prethodno adresiranog registra, koristimo funkciju

Wire.requestFrom(DeviceAddr, NoOfBytes);

DeviceAddr – Adresa uređaja iz koga čitamo

NoOfBytes – Broj bajtova koji želimo da pročitamo iz uređaja

Zadatak ove funkcije je da od uređaja zahteva NoOfBytes bajtova. Nakon što uređaj primi te bajtova, smešta ih u prihvatne registre. Da bi pročitali bajtove iz prihvatnih registara koristimo funkciju

Wire.read();

Unutar funkcije MPU_9250 vršimo dodeljivanje adrese prosleđene kao argument internoj promenljivoj *Module_Address*. Nakon toga funkcijom *Wire.Begin()* započinjemo prenos. Prilikom svake I2C komunikacije neophodno je da ova funkcija bude pozvana kao prva.

Unutar funkcije *ReadAccel* se vrši čitanje sa senzora i to tako što se realizuje niz koraka definisanih protokolom. Najpre, funkcijom *Wire.beginTransmission()* postavimo adresu uređaja *Module_Address*. Nakon toga upišemo adresu prvog registra iz koga želimo da pročitamo niz bajtova. To se postiže pomoću funkcije *Wire.write()* koja u ovom slučaju postavlja adresu registra *ACCEL_XOUT_H*. Ovom funkcijom su u pomoćne promenljive klase *Write* za sada smeštene dve vrednosti: adresa uređaja i adresa registara. Sada je potrebno te adrese „pustiti“ na I2C magistralu i to se postiže pomoću funkcije *Wire.endTransmission()*.

Nakon što smo adresirali uređaj, funkcijom *Wire.requestFrom()* zahtevamo od uređaja da nam pošalje 6 bajtova podatka. Nakon toga funkcijom *Wire.read()* čitamo tih 6 bajtova. Ovi bajtovi su organizovani tako da pvo ide viši pa niži bajt sa očitanoog senzora pa je zbog toga neophodno prvi bajt pomeriti za 8 mesta u levo.

Podešavanje rezolucije se postiže pomoću funkcije *SetAccelRes* koja na osnovu vrednosti argumenta *Resolution* vrši dodelu odgovarajuće vrednosti sadržaju promenljive

BinRes koja ima za zadatak da na poziciji bita 3 i 4 registra *ACCEL_CONF* podesi bite tako da se dobije odgovarajuća rezolucija. Nakon toga, vrši se upis vrednosti *BinRes* u registar *ACCEL_CONF* po protokolu komunikacije za upis na I2C magistralu. Analizom *.cpp* fajla smo završili analizu sadržaja fajlova biblioteke.

Sledeći korak je stvaranje korisničke biblioteke od kreiranih fajlova. U folderu u kome je kreiran projekat postoje 2 fajla kao što je prikazano na slici 13 - 7.

	AccelLib	11/6/2018 07:43	C++ Source File	2 KB
	AccelLib	11/6/2018 06:08	C/C++ Header File	1 KB

Slika 13 - 7 - Sadržaj foldera u kome su kreirani fajlovi za biblioteku

Ova dva fajla staviti u jednu arhivu sa ekstenzijom *.zip* i toj arhivi dati naziv *AccelLib*. Na taj način smo dobili biblioteku koja se može importovati u bilo koji *Arduino* projekat i pojednostaviti korišćenje modula *MPU_9250*. Kada imamo biblioteku u *.zip* fomratu, njena integracija u sistem se vrši na standardan način opisan u poglavlju 2.2

13.3. PRIMERI

Primeri koji slede imaju za cilj da ilustruju način korišćenja prethodno opisane biblioteke

Primer 13.1. – Očitavanje ubrzanja duž X, Y i Z ose primenom MPU_9250 modula

- *Realizovati Arduino sistem koji se sastoji od MPU_9250 modula sa koga očitavamo ubrzanje duž 3 ose koordinatnog sistema. Rezultate očitavanja potrebno je ispisivati na serijsku konzolu. Napisati Arduino kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 4 MM kratkospajča
- MPU_9250 modul

Električna šema koja ilustruje način povezivanja ovog modula data je na slici 13 - 2, dok je kod koji implementira traženu funkcionalnost je sledeći:

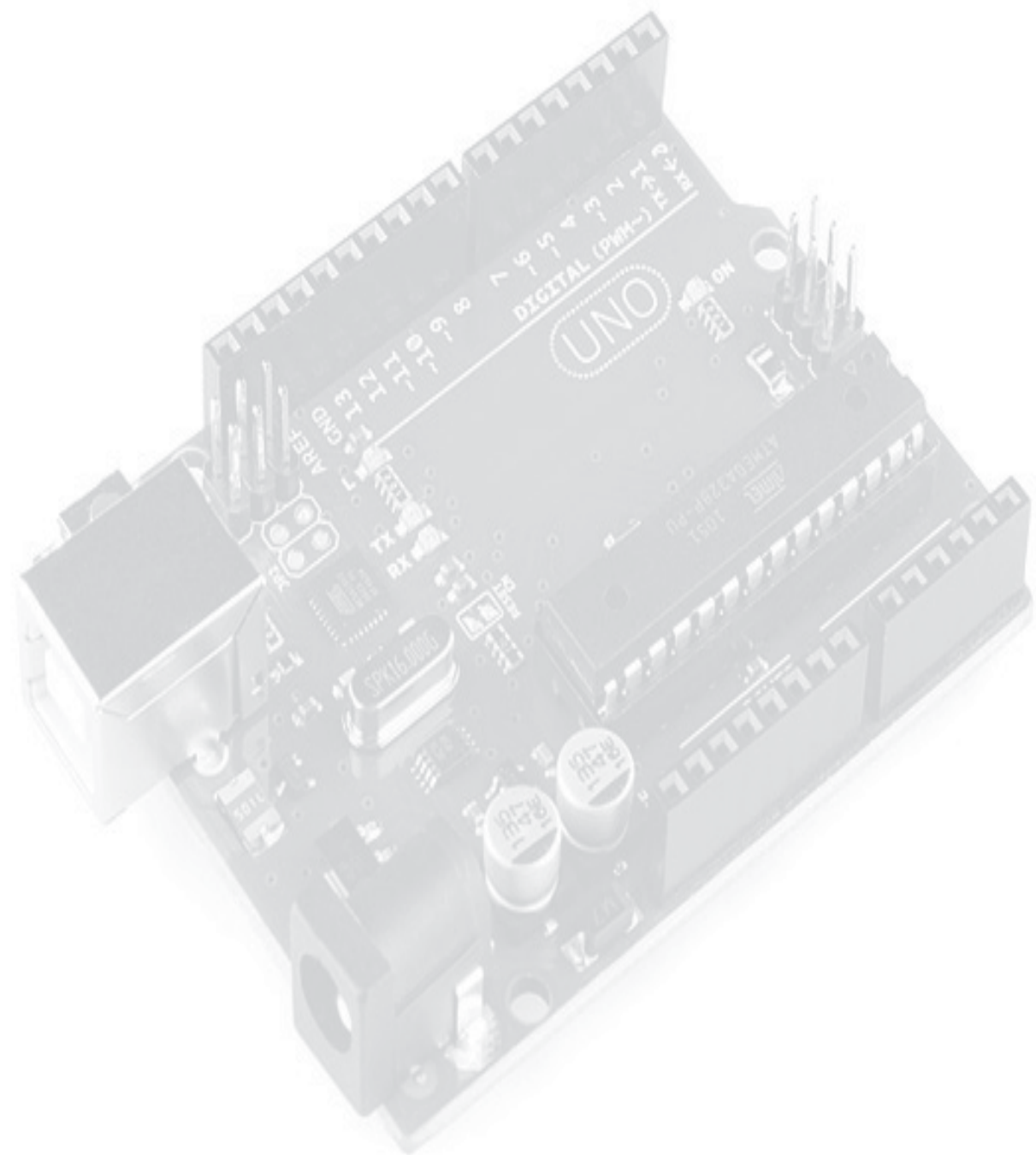
```
#include <AccelLib.h>
#define ModuleAddress 0x68
MPU_9250 Module(ModuleAddress);
int16_t X;
int16_t Y;
int16_t Z;
void setup() {
    Serial.begin(9600);
}
void loop() {
    Module.SetAccelRes(2);
    Module.ReadAccel(&X,&Y,&Z);
    Serial.print("X:");
    Serial.println(X);
    Serial.print("Y:");
    Serial.println(Y);
    Serial.print("Z:");
    Serial.println(Z);
    Serial.println("*****");
}
```

Iz prikazanog koda može se primetiti da je uvođenje biblioteke pojednostavilo korišćenje modula i povećalo njegovu čitljivost i razumljivost. Sada rad sa modulom postaje jednostavan i nije potrebno dublje zalaženje u konkretnu hardversku realizaciju modula.

13.4. ZADACI ZA SAMOSTALNI RAD

1. Realizovati Arduino sistem koji se sastoji od modula MPU_9250, a koji služi za detekciju potresa. Ukoliko se detektuje potres uključuje se dioda. Napisati kod za Arduino hardversku platformu koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije. Potres se može detektovati tako što se uzmu vrednosti ubrzanja sa sva tri senzora, kvadriraju se i saberu. Rezultat ovog izraza se zatim poredi sa nekim predefinisanim pragom i ukoliko je vrednost veća od praga dioda se uključuje.
2. Napraviti biblioteku za korišćenje četvorocifrenog sedmosegmentnog displeja iz primera 9.2. Biblioteka treba da implementira funkcije koje pojednostavljaju korišćenje ovog displeja. Pod tim se podrazumeva da postoje funkcije za inicijalizaciju sva četiri displeja u celini, za uključivanje pojedinacnog displeja kao i za ispis vrednosti na pojedinačni displej. Napraviti biblioteku za Arduino IDE okruženje koja implementira traženu funkcionalnost i praktično potvrditi uspešnost realizovane biblioteke. Implementirati algoritam za multipleksiranje displeja korišćenjem kreirane biblioteke.

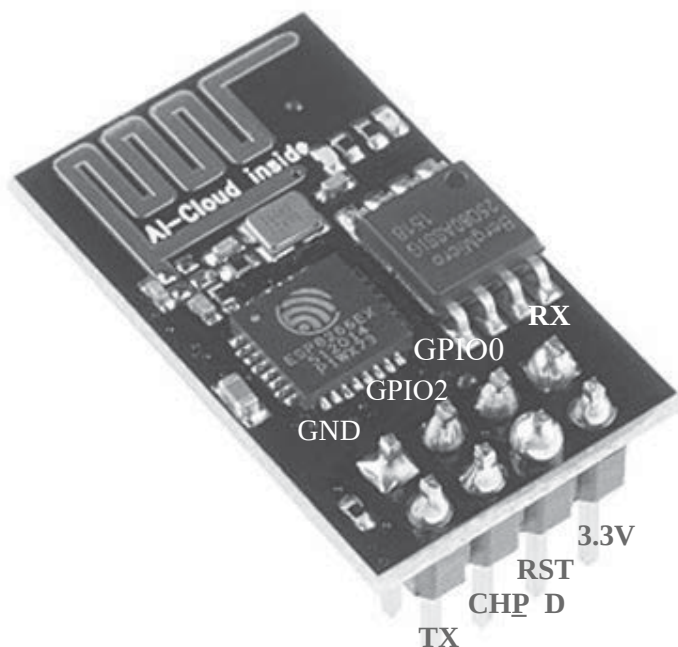
3. Napraviti korisničku biblioteku za korišćenje digitalnog barometra iz primera 10.10. Biblioteka treba da realizuje funkcije za inicijalizaciju modula, čitanje temperature i pritiska kao i očitavanje relativne visine. Napisati Arduino kod, koji poziva realizovanu biblioteku i demonstrira ispravnost njenih funkcija.
4. Napraviti korisničku biblioteku koja pojednostavljuje korišćenje funkcionalnosti ultrazvučnog modula iz primera 10.1. Potrebno je da biblioteka realizuje funkciju za konfiguraciju modula kao i funkciju za očitavanje razdaljine. Napisati kod za Arduino Uno hardversku platformu i demonstrirati ispravnost realizovane biblioteke.



14. WIFI KOMUNIKACIJA

Ovo poglavlje ima za cilj da čitaoca upozna sa modulom za wifi komunikaciju ESP8266. Ovaj modul omogućava povezivanje Arduina na internet i tako daje osnovu za kreiranje interneta stvari (eng. *IoT*)

ESP8266 modul, sa oznakama pinova, prikazan je na slici 14 - 1.



Slika 14 - 1 - ESP8266 Wifi modul

Podešavanje, konfiguracija i korišćenje funkcionalnosti ovog modula je moguće ostvariti slanjem stringova posebnih formata koje ovaj uređaj može da tumači. Ti stringovi se po standardu nazivaju AT komande i imaju sledeći format:

komanda=argumenti\r\n

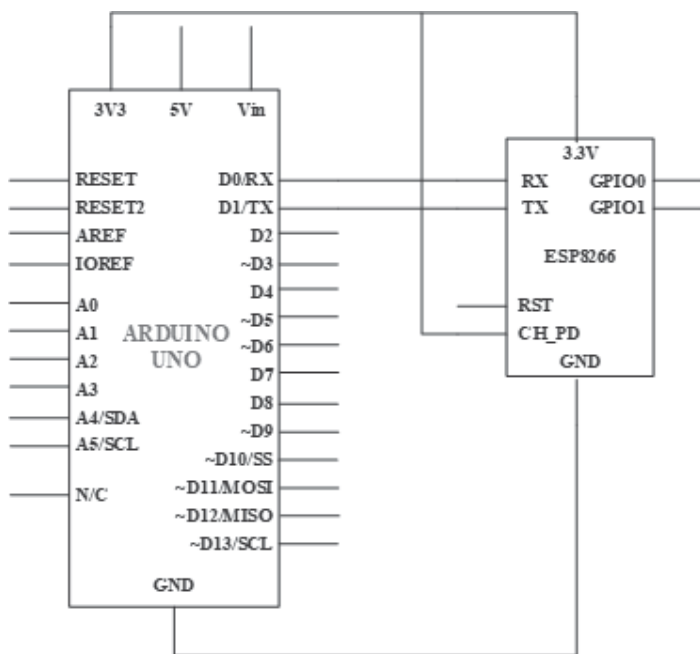
komanda predstavlja string čija vrednost pripada konačnom skupu vrednosti koje uređaj može da razume. Svaka AT komanda počinje sa ključnom reči AT nakon koje sledi znak +, a zatim se nadovezuje odgovarajuća podkomanda koju uređaj treba da razume i da u skladu sa njom preduzme odgovarajuću akciju.

Iza naziva komande mogu, a i ne moraju, da slede *argumenti* koji nose dodatne informacije. Da li iza neke komande slede argumenti ili ne, može se videti iz dokumentacije AT komandi za taj uređaj. Na web stranici [20] mogu se naći detaljni opisi svih komandi na koje ovaj uređaj može da odgovori kao, i to da li primaju argumente i ako primaju, koji je očekivani format tih argumenata.

Obavezno je svaku AT komandu završiti sa dva karaktera `\r\n`, koje uređaj tumači kao kraj komande.

Kada uređaj primi neku AT komandu on na nju odgovara. Struktura odgovora zavisi od poslate AT komande. Međutim, svaki odgovor na AT komandu završava se sa jednom od dve ključne reči OK ili ERROR. Odgovor se završava sa OK kada uređaj prepozna AT komandu dok se u suprotnom odgovor završava sa ERROR. U dokumentaciji [20] se može naći struktura odgovora na svaku od komandi.

Da bi se bolje razumeo sam princip komunikacije putem AT komandi, u nastavku sledi primer koji ilustruje način konfiguracije uređaja kao AP (*Access point*) i konfiguracija uređaja kao internet servera. Da bi bilo moguće ostvariti komunikaciju sa uređajem, najpre je neophodno povezati uređaj sa Arduinoom, kao što je prikazano na slici 14 - 2.



Slika 14 - 2 - Način povezivanja Arduina i ESP8266 modula



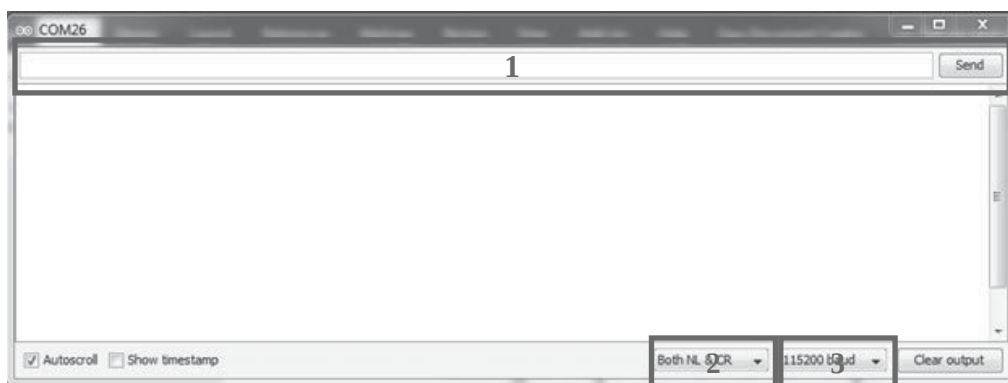
Treba voditi računa o tome da se UART veza sa ovim uređajem ostvaruje tako što se RX linija Arduino hardverske platforme povezuje sa RX linijom ESP8266 modula a TX linija Arduino hardverske platforme sa TX linijom ESP8266 modula.

14.1. KONFIGURACIJA MODULA KAO ACCESSPOINT-A I USPOSTAVLJANJE SERVERA

Ovo poglavlje ima za cilj da čitaoca upozna sa načinom korišćenja AT komandi. AT komande koje ovaj uređaj podržava date su u dokumentu [17]. Međutim, u ovom delu se koriste samo neke od njih. Korišćene AT komande će biti detaljno objašnjene u nastavku.

Nakon što se ostvari hardverska veza između Arduino hardverske platforme i ESP8266 modula neophodno je testirati da li modul funkcioniše, tj. da li je modul dobro povezan sa Arduino. Da bi mogli da uradimo ovo testiranje, neophodno je otvoriti Arduino IDE softversko okruženje i otvoriti novi Arduino projekat. Na Arduino treba spustiti prazan projekat, a zatim otvoriti serijski monitor (za detalje videte poglavlje 10).

Serijski monitor se koristi za slanje AT komandi ka uređaju i čitanje sadržaja odgovora koji se dobijaju od uređaja. Nakon što se otvori serijski monitor, dobija se prozor kao na slici 14 - 3.



Slika 14 - 3 – Arduino IDE serial monitor-delovi

1. Ovaj pravougaonik služi za unos AT komandi koje se šalju ka uređaju
2. Neophodno je podesiti da se svaki tekst unesen u pravougaonik završava sa `\r\n`
3. Podrazumevana brzina komunikacije sa uređajem je 115200

Nakon što je podešeno radno orkuženje *Serial Monitor*-a neophodno je poslati prvu AT komandu koja testira uspešnost ostvarene veze. Ukoliko se kao odgovor na ovu komandu dobije OK, to znači da je uređaj spreman za rad, dok u suprotnom treba nastaviti sa slanjem ovih komandi sve dok se ne dobije odgovor OK. Tipičan primer jedne ovakve sekvence je sledeći:

1. Poslali smo prvu AT komandu i dobili odgovor ERROR što znači da treba ponovo ponoviti AT komandu
2. Ponovo dobijamo isti odgovor, pa treba nastaviti sa slanjem
3. Uređaj je spreman jer smo dobili odgovor OK



Slika 14 - 4 - Arduino IDE Serial monitor posle slanja komande AT

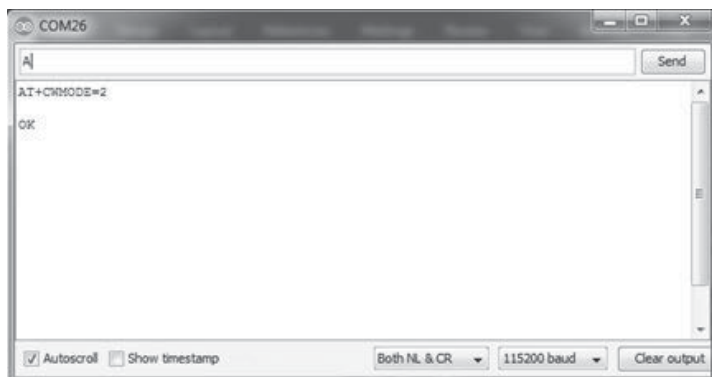


ERROR se obično javlja usled toga što se pogreši u nazivu komande, ili se komande previše brzo šalju jedna za drugom. Da bi ovo izbegli neophodno je ubaciti neko kašnjenje između sekvencijalnih slanja komandi.

Nakon što smo dobili potvrdu od modula, možemo zaključiti da je dobro povezan. Sada je neophodno konfigurisati modul da bi mogao da se koristi kao Wifi AP. Pošto uređaj može da se koristi kao wifi client i AP, potrebno je uređaju poslati AT komandu koja ga postavlja u AP mod. To se postiže sledećom komandom.

AT+CWMODE = mode

mode – uzima jednu od tri vrednosti (1,2,3) koje označavaju različite režime rada. Mi koristimo mogućnost uređaja da radi kao AP što se postiže brojem 2. Dakle, potrebno je poslati ovu komandu nakon koje dobijamo situaciju kao na slici 14 - 5.



Slika 14 - 5 - Izgled Arduino IDE Serial monitora nakon slanja komande za podešavanje uređaja kao AP

Od modula dobijamo odgovor OK čime modul potvrđuje da je konfigurisan u ovom modu. Sada je potrebno konfigurisati naziv i password našeg Wifi APa. To se postiže sledećom komandom

AT+CWSAP=ssid, pwd, ch, ecn

ssid – string čiji sadržaj predstavlja naziv APa

pwd – string, minimalne dužine 8 karaktera, čiji sadržaj predstavlja password APa

ch – broj kanala wifi (staviti 5)

ecn – broj koji označava enkripciju koja se primenjuje prilikom tumačenja password-a (staviti 3)

Sada je potrebno i ovu komandu poslati ka uređaju nakon čega dobijamo izgled odgovora kao na slici 14 - 6.



Slika 14 - 6 - Izgled Arduino IDE Serial monitor-a nakon podešavanja SSIDa i password-a



Slika 14 - 7 - Izgled Arduino IDE Serial monitor-a nakon podešavanja broja klijenata

Ovim smo završili konfiguraciju modula kao wifi APa. Sada je neophodno konfigurirati naš uređaj kao Web Server sa koga ćemo moći da pročitate neke podatke. Najpre je potrebno dozvoliti da više uređaja može istovremeno (hardverska ograničenja ESP modula dozvoljavaju da to budu maksimalno 4) da pristupi našem serveru. To se postiže pomoću komande

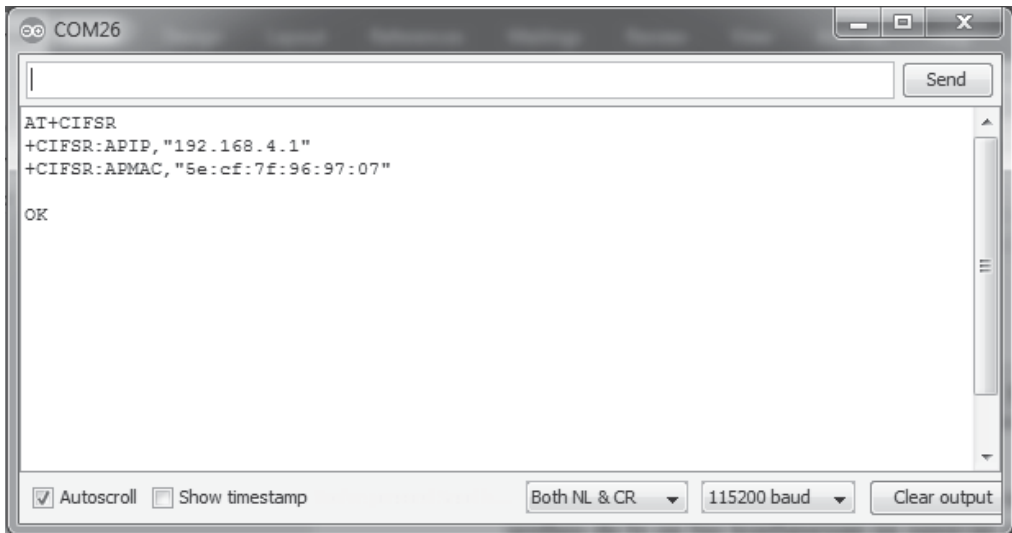
AT+CIPMUX = mode

Web server je uređaj koji treba da odgovori na zahtev klijenta. Klijent-a u ovoj komunikaciji predstavlja neki uređaj (mobilni telefon, tablet, PC) koji serveru šalje zahtev za podacima. Server nakon što detektuje da je primio zahtev, klijentu odgovara slanjem odgovarajućeg sadržaja. Međutim, da bi bilo moguće pristupiti uređaju kao serveru, neophodno je otvoriti odgovarajući serverski port na uređaju. To se postiže komandom:

AT+CIPSERVER=mode,port

mode – Ako je vrednost ovog argumenta 0, to uređaju govori da treba da izbriše server. Ako je vrednost ovog argumenta 1, to uređaju govori da treba da kreira server na portu 80
port – vrednost porta u opsegu od 0-65000.

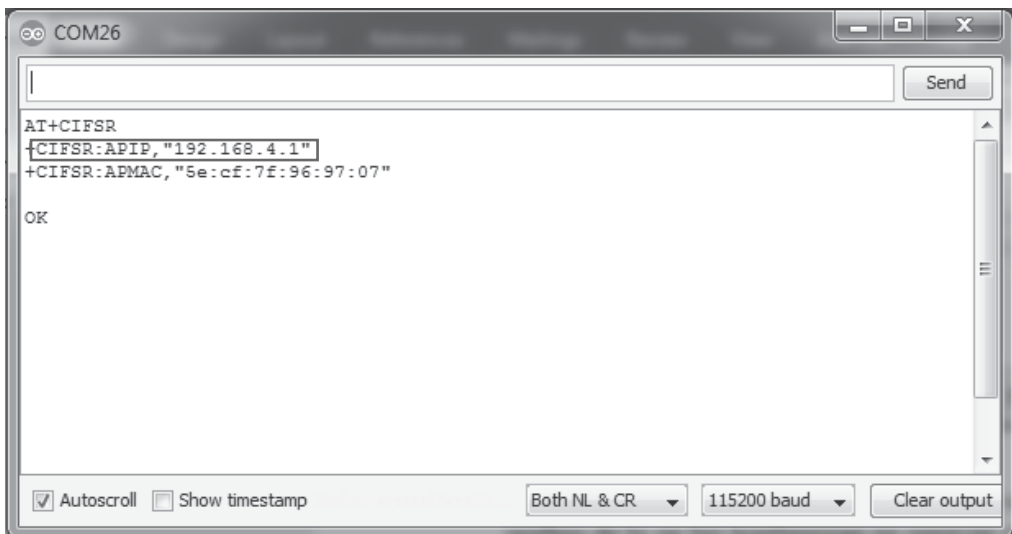
Nakon slanja ove komande prozor serijskog interfejsa ima izgled kao na slici 14 - 8.



Slika 14 - 8 - Izgled Arduino IDE Serial monitor-a nakon otvaranja servera na portu 80

Kada se povežemo sa uređajem u cilju očitavanja podataka sa servera, neophodno je dobiti IP adresu uređaja kako bi korisnik znao kako da pristupi serveru na mreži. IP adresu koja je dodeljena serveru možemo pročitati pomoću komande

AT+CIFSR



Slika 14 - 9 - Provera podešenog IP-a servera

Nakon slanja komande dobijamo odgovor koji sadži IP adresu servera, što je prikazano na slici 14 - 9.

Na osnovu dobijenog odgovora može se videti da je adresa našeg servera 192.168.4.1. Sada je potrebno izvršiti povezivanje na AP koji bi trebao da je vidljiv u dostupnim mrežama, kao što je prikazano na slici 14 - 10.



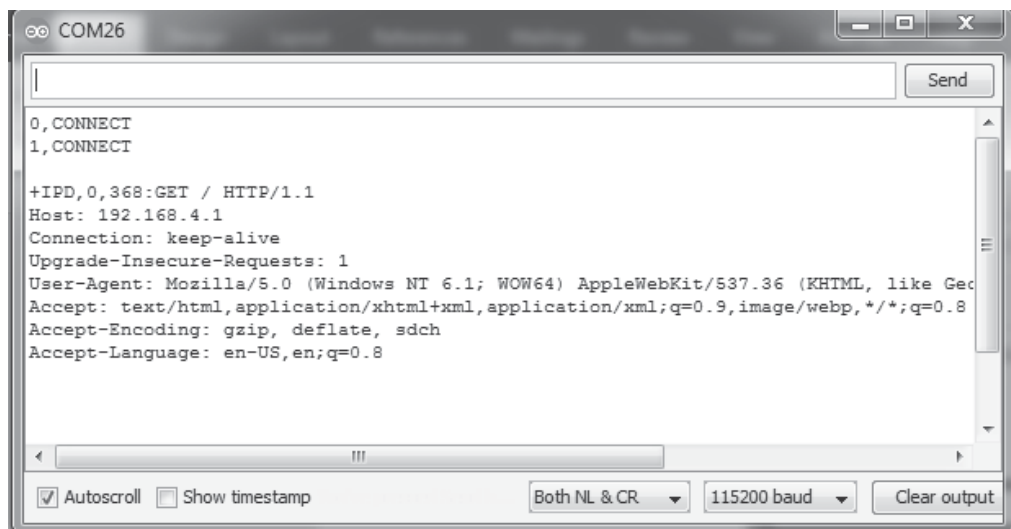
Slika 14 - 10 - Konfigurisani AP. Na listi vidljiv AP na klijentskoj strani

Nakon povezivanja uređaja sa AP neophodno je otvoriti Web browser i u navigacioni deo web browsera treba uneti sledeći URL:

<http://192.168.4.1:80>

Kada ovaj URL unesemo u Web Browser, on šalje zahtev serveru, na adresi 192:168.4.1. portu 80, da mu dostavi sadržaj. Pošto trenutno na serveru ne postoji sadržaj, neophodno je napraviti Arduino program koji će usled detektovanja klijentskog zahteva poslati sadržaj klijentu.

Potrebno je primetiti da usled slanja zahteva od strane Web Browsersa ka serveru, naš AP šalje određene informacije koje je potrebno detektovati kao zahtev klijenta. Te informacije se ispisuju u serijskoj konzoli i prikazane su na slici 14 - 11. Ova poruka se može softverski obraditi u cilju detekcije pristiglog zahteva na server što je i urađeno u primeru 14.1.



Slika 14 - 11 - Izgled Arduino IDE Serial monitor-a nakon zahteva klijenta

14.2. PRIMERI

Primeri koji slede imaju za cilj da čitaoca upoznaju sa načinom korišćenja Wifi modula ESP8266. Da bi razumevanje primera bilo kompletno, neophodno je da čitalac poseduje određena predznanja iz arhitekture računarskih mreža kao i osnovna znanja iz HTML programiranja.

Primer 14.1. - Implementacija servera na ESP8266 modulu

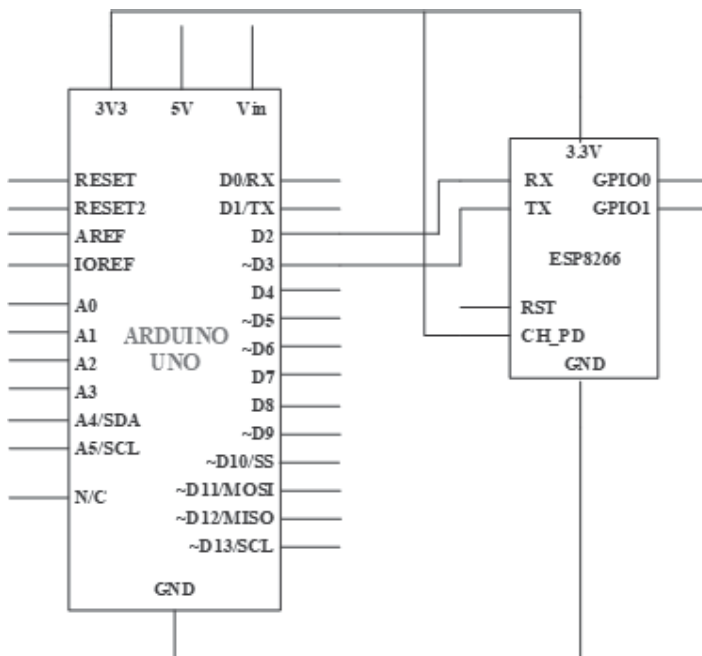
- *Neophodno je realizovati Arduino sistem koji se sastoji od Wifi modula ESP8266 koji predstavlja server na kome se nalazi HTML stranica koja se prosleđuje klijentu prilikom pristupa adresi servera. Sadržaj stranice treba da bude takav da ispisuje tekst „Internet stvari“ formatiran u okviru HTML elementa paragraf (p). Napisati Arduino kod koji implementira traženu funkcionalnost i praktično potvrditi uspešnost realizacije.*

Rešenje:

Za realizaciju ovog sistema neophodne su sledeće komponente:

- Arduino UNO
- 5 MŽ i 1 MM kratkospajač
- Protobord
- ESP8266 modul

Električna šema koja ilustruje način povezivanja prikazana je na slici 14 - 12.



Slika 14 - 12 - Električna šema sistema iz primera 14.1

Slika 14 - 12 se razlikuje od slike 14 - 2 po pinovima na koje su povezani RX i TX pinovi ESP8266 modula. Za Arduino hardversku platformu postoji biblioteka koja može simulirati UART magistralu na bilo koja dva pina. Ta biblioteka se naziva SoftwareSerial i predstavlja deo Arduin IDE razvojnog okruženja. Kod rešenja je:

```
#include <SoftwareSerial.h>
const int RxPinNo = 2;
const int TxPinNo = 3;
//Create Virtual UART interface
SoftwareSerial vUART(RxPinNo,TxPinNo);
//Send AT command data via vUART interface
String sendData(String command, const int timeout){
String response = "";
```

```
// send the read character to the esp8266
vUART.print(command);
long int time = millis();
while( (time+timeout) > millis()){
while(vUART.available()){

char c = vUART.read();
// read the next character.
response+=c;
    }
}
return response;
}

void setup() {
    Serial.begin(115200);
    vUART.begin(115200);
    //Send AT command to configure our module as AP server
    //Reset module
    sendData("AT+RST\r\n",2000);
    //Send AT command to configure our module as AP server
    //Reset module
    sendData("AT+RST\r\n",2000);
    //Configure AP name and password
    sendData("AT+CWSAP=\"IoT1\", \"1234test\",5,3\r\n",2000);
    //Configure as access point
    sendData("AT+CWMODE=2\r\n",2000);
    //Configure for multiple connections
    sendData("AT+CIPMUX=1\r\n",2000);
    //Configure server on port 89
    sendData("AT+CIPSERVER=1,80\r\n",2000);
}

void loop() {
    if(vUART.available()){
        //If client request page, server will receive +IPD AT command
        if(vUART.find("+IPD,")){
            delay(1000);
            //Get ID by substracting 48 to get number from char
            int connectionId = vUART.read()-48;
            //Web page content
            String webpage = "<h1>Internet stvari</h1><p>Dobro dosli
na prvu stranicu</p>";
            //Create AT command which open server connectio
            String cipSend = "AT+CIPSEND=";
            cipSend += connectionId;
            cipSend += ",";
            cipSend +=webpage.length();
            cipSend += "\r\n";
            sendData(cipSend,1000);
            sendData(webpage,1000);
            //Add extra content to the page
        }
    }
}
```

```

webpage="<button>Dummy</button>";
cipSend = "AT+CIPSEND=";
cipSend += connectionId;
cipSend += ",";
cipSend +=webpage.length();
cipSend +=webpage.length();
cipSend += "\r\n";
sendData(cipSend,1000);
sendData(webpage,1000);
//Close the server connection
String closeCommand = "AT+CIPCLOSE=";
closeCommand+=connectionId; // append connection id
closeCommand+="\r\n";
sendData(closeCommand,3000);
    }
}
}

```

U funkciji *setup* izvršena je konfiguracija modula slanjem odgovarajućih AT komandi. Redosled slanja AT komandi je isti kao redosled slanja komandi sproveden u poglavlju 14.1. Komande se ka ESP8266 modulu šalju posredstvom funkcije *sendData* koja ima zadatak da pošalje prosleđenu komandu, a da nakon slanja komande, sačeka odgovor i prosledi ga kao povratnu vrednost.

Nakon što se inicijalizuje uređaj posredstvom AT komandi ulazi se u glavnu programsku petlju u kojoj se čeka da klijent zatraži zahtev od servera. To se prepoznaje na osnovu poruke koju prima server. Ova poruka identična je poruci sa slike 14 - 12, što znači da treba detektovati odgovor koji počinje sa +IPD nakon koga sledi ID klijenta. ID klijenta je neophodan za dalju komunikaciju pa je potrebno ekstrahovati tu informaciju, što se postiže konvertovanjem prvog sledećeg karaktera u nizu u celobrojnu vrednost koja se smešta u promenljivu *connectionId*.

AT komanda za slanje sadržaja ka klijentu je

AP+CIPSEND = id, length

id – jedinstveni broj klijenta

length – dužina podatka, maksimalno 2048 bajtova

Nakon slanja ove komande potrebno je poslati sadržaj. Komanda je smeštena u string *cipSend* dok je sadržaj smešten u string *WebPage*. Nakon poslatog sadržaja potrebno je zatvoriti konekciju slanjem komande

AP+CIPCLOSE = id

14.3. ZADACI ZA SAMOSTALNI RAD

1. Potrebno je realizovati *IoT* sistem koji se sastoji od Arduino hardverske platforme, Wifi modula ESP8266 i diode. Sistem treba da, nakon što detektuje zahtev od klijenta, promeni stanje LE diode. Inicijalno je dioda uključena. Napisati Arduino kod koji realizuje zahtevanu funkcionalnost i praktično potvrditi uspešnost realizacije.
2. Realizovati IoT sistem koji se sastoji od temperaturnog senzora LM35, Arduino hardverske platforme i Wifi modula ESP8266. Sistem treba da, nakon pristiglog zahteva sa klijenta (Web Browser-a), klijentu prosledi informaciju o očitanoj temperaturi. Informacija se prosleđuje u okviru HTML elementa `p` (paragrafa). Napisati kod za Arduino hardversku platformu i praktično potvrditi uspešnost realizacije.

15. LITERATURA

- [1] https://en.wikipedia.org/wiki/Open-source_software
- [2] <https://learn.sparkfun.com/tutorials/what-is-an-arduino>
- [3] <https://circuitdigest.com/arduino-projects> , <https://create.arduino.cc/projecthub> , <https://www.hackster.io/arduino/projects>
- [4] <https://en.wikipedia.org/wiki/Arduino>
- [5] <https://www.arduino.cc/en/Main/Products>
- [6] <https://www.arduino.cc>
- [7] <https://www.arduino.cc/reference/en/language/variables/constants/constants/>
- [8] <http://www.ti.com/lit/ds/symlink/lm35.pdf>
- [9] <https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf>
- [10] <https://www.arduino.cc/documents/datasheets/LCD-WH1602B-TMI-ET%23.pdf>
- [11] https://en.wikipedia.org/wiki/Pulse-width_modulation
- [12] https://en.wikipedia.org/wiki/Embedded_system
- [13] [https://en.wikipedia.org/wiki/Duplex_\(telecommunications\)](https://en.wikipedia.org/wiki/Duplex_(telecommunications))
- [14] https://en.wikipedia.org/wiki/Serial_Peripheral_Interface
- [15] <https://en.wikipedia.org/wiki/I%C2%B2C>
- [16] <http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-1.1.pdf>
- [17] <https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/>
- [18] https://play.google.com/store/apps/details?id=jp.side2.apps.btterm&hl=sr_LATN
- [19] https://github.com/sparkfun/BMP180_Breakout_Arduino_Library/archive/master.zip
- [20] <https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/>
- [21] <https://www.fluke.com/en-us/learn/best-practices/measurement-basics>
- [22] https://www.pjrc.com/teensy/td_libs_TimerOne.html

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004.78:004.35
004.383/.384

ТАНАСКОВИЋ, Марко, 1986-

Internet stvari : Arduino praktikum / Marko Tanasković. - Beograd : Univerzitet
Singidunum, 2020 (Beograd : Caligraph). - V, 229 str. : ilustr. ; 24 cm

Tiraž 400. - Bibliografija: str. 229.

ISBN 978-86-7912-724-2

a) Интернет ствари b) Микроконтролери -- Примена

COBISS.SR-ID 283003916

© 2020.

Sva prava zadržana. Nijedan deo ove publikacije ne može biti reprodukovan u bilo kom vidu i putem bilo kog medija, u delovima ili celini bez prethodne pismene saglasnosti izdavača.



Marko Tanasković

INTERNET STVARI

INTERNET OF THINGS

IOT

Ovaj praktikum je namenjen pre svega studentima softverskog inženjerstva, kao podloga za izradu laboratorijskih vežbi za predmet internet stvari korišćenjem Arduino razvojnog sistema. Međutim mogu je koristiti svi oni koje zanimaju oblasti programiranja mikrokontrolera i interneta stvari. Priručnik je pisan tako da ga mogu koristiti i oni koji imaju početna znanja iz oblasti elektronike i programiranja mikrokontrolera, ali i oni koji tek prave svoje prve korake u ovim oblastima.

Priručnik obrađuje teme koje su važne za oblast interneta stvari polazeći od poglavlja koja daju osnove Arduino sistema, objašnjavaju kako se fizički sklapaju električne šeme, preko poglavlja koja objašnjavaju kako se prave jednostavne aplikacije koristeći digitalne i analogne ulaze, pa sve do poglavlja u kojima su predstavljeni složeniji projekti koji podrazumevaju rad sa modulima, komunikaciju sa drugim uređajima ili izlazak na internet uz pomoć wi-fija. Poglavlja su organizovana tako da su osnovne stvari i najjednostavniji problemi pokriveni u prvim poglavljima, dok težina problema koji se obrađuju raste kroz svako sledeće poglavlje.