

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6721

**BIBLIOTEKA ZA OSTVARENJE IZBORNIKA ZA MALE
TEKSTUALNE LCD ZASLONE**

Ana Andrašek

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6721

**BIBLIOTEKA ZA OSTVARENJE IZBORNIKA ZA MALE
TEKSTUALNE LCD ZASLONE**

Ana Andrašek

Zagreb, lipanj 2020.

**SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6721

Pristupnica: **Ana Andrašek (0036502018)**

Studij: Računarstvo

Modul: Računalno inženjerstvo

Mentor: doc. dr. sc. Leonardo Jelenković

Zadatak: **Biblioteka za ostvarenje izbornika za male tekstualne LCD zaslone**

Opis zadatka:

Proučiti uobičajene mogućnosti tekstualnih LCD zaslona koji se spajaju na mikroračunala, a koji omogućuju i prikaz poruka i upravljanje radom sustava kroz načelo izbornika. Osmisliti i ostvariti biblioteku za podršku ostvarenja upravljanja takvim LCD zaslonima koji, uz sam zaslon, imaju i nekoliko tipaka za podršku navigacije kroz izbornik, pregled i promjene vrijednosti. Ispitati zahtjeve koje ostvarena biblioteka traži od sustava te ograničenja u njenoj primjeni.

Rok za predaju rada: 12. lipnja 2020.

SADRŽAJ

1.	Uvod	1
2.	LCD ZASLON	2
3.	UREĐAJI POTREBNI ZA RAD S BIBLIOTEKOM.....	5
3.1.	LCD ZASLON	5
3.2.	Arduino	8
3.3.	Povezivanje Arduina i LCD zaslona.....	10
3.4.	Arduino IDE.....	13
4.	OSTVARENJE BIBLIOTEKE ZA IZBORNIK NA LCD ZASLONU.....	15
4.1.	Konfiguracijska datoteka	15
4.2.	Implementacija	16
5.	PRIMJERI KORIŠTENJA BIBLIOTEKE	19
5.1.	Primjer osnovnog izbornika	19
5.2.	Primjer s višestrukim i jednostrukim odabirom	21
5.3.	Primjer promjene cjelobrojnih i decimalnih vrijednosti	23
5.4.	Primjer promjene osvjetljenja LCD zaslona	24
6.	PROBLEMI I MOGUĆA POBOLJŠANJA	26
7.	Zaključak	27
	Literatura	28
	Sažetak	29
	Summary	29

1. UVOD

Početna motivacija ovog rada bila je napraviti općenitu biblioteku koja bi se koristila za male tekstualne LCD zaslone (engl. *Liquid Crystal Display*) i koju bi korisnik mogao uključiti u svoj Arduino program. Svrha biblioteke bila bi jednostavan izbornik. Korisnik bi po svojoj želji i potrebama mogao napraviti vrlo jednostavno izbornik po kojem bi se kretao, ali i mijenjao vrijednosti nekih varijabli pomoću gumbi. Biblioteka bi trebala biti napravljena na nekoj općoj razini, odnosno trebala bi se moći koristiti za različite tekstualne LCD zaslone, neovisno o njihovoj veličini (broju redaka i stupaca). Također, trebalo bi uzeti u obzir i mogući različiti način spajanja gumbi.

Ovaj rad podijeljen je u sedam poglavlja. U samom uvodu, odnosno prvom poglavlju opisuje se početna ideja i motivacija za izradu ovog rada.

U drugom poglavlju uvod je proširen nekim informacijama o LCD zaslonima. Navedene su različite vrste zaslona te je opisan njihov način rada. Također, u tom poglavlju dati su neki primjeri korištenja izbornika na LCD zaslonima u stvarnim sustavima.

U trećem poglavlju navedeni su konkretni uređaji potrebni za ovaj rad te dodatne informacije o njima. Opisani su i LCD modul i mikroračunalo¹ te njihova međusobna povezanost. Osim sklopoljva (engl. *hardware*), opisana je i korištena programska podrška (engl. *softver*).

U četvrtom poglavlju ukratko je opisana implementacija biblioteke te „konfiguracijska“ datoteka za korisnike.

Zatim, u petom poglavlju navedeni su konkretni primjeri, odnosno konkretan rad s bibliotekom. Primjeri su opisani, ali i vizualno je demonstriran njihov rad.

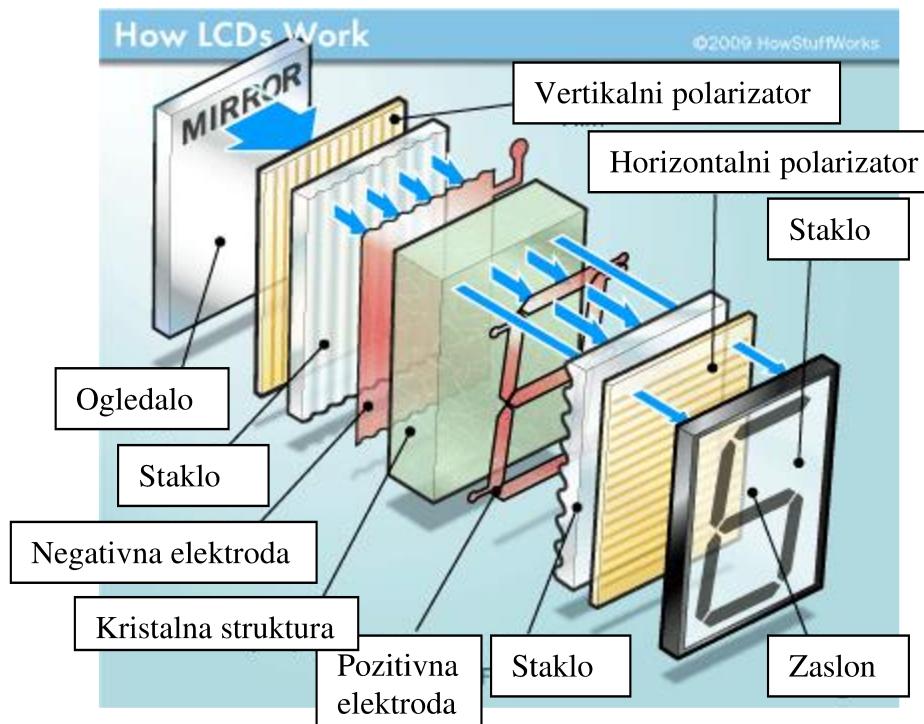
U šestom poglavlju opisani su problemi i ograničenja ove biblioteke, ali i moguća poboljšanja.

¹ „Računalo“ na jednom čipu. Sadrži procesor, memoriju i ulazno–izlazne periferne uređaje. Koriste se u ugradbenim računalnim sustavima.

2. LCD ZASLON

LCD zaslon vrsta je ravnog zaslona koji koristi tekuće kristale u svom radu. LCD zasloni troše manje energije od LED zaslona pošto LED emitira svjetlost, dok LCD radi na principu njegova blokiranja. Ukratko, LCD zasloni koriste bijelo pozadinsko osvjetljenje (engl. *backlight*) koje prolazi kroz polarizator² nakon čega svjetlost oscilira na jednak način. Nakon toga dolazi sloj „tekućeg kristala“, odnosno kristalna struktura koja mijenja polarizaciju svjetlosti koja kroz nju prolazi. Odnosno, u stanju isključenosti ne mijenja polarizaciju, a kada je u stanju uključenosti mijenja postotak polarizacije jer nakon kristala svjetlost dolazi na drugi polarizator koji je zaokrenut za 90 stupnjeva od prvog polarizatora. Dakle, u stanju isključenosti drugi polarizator blokira svjetlost, a u stanju uključenosti kroz njega prolazi svjetlost te tu svjetlost mi vidimo na zaslonu. Kako bismo prikazali boju, nakon toga postoje tri filtera u boji (crveni, zeleni i plavi) koji blokiraju sve ostale boje osim njih samih. Ovo objašnjenje prikazano je na slici Slika 2.1.

LCD zasloni razlikuju se po kristalnoj strukturi. Tako postoji kristalna struktura koja može zakretati (engl. *Twisted Nematic*, TN) polarizaciju za 90 stupnjeva ili ju može uskladiti s prvim polarizatorom. Međutim, ovo nije najpreciznije pa za posljedicu može imati lošu boju i kontrast. Zatim postoje i LCD zasloni u kojemu se kristali poravnavaju vertikalno ili horizontalno (engl. *Vertical Alignment*, VA), no oni imaju sporije vrijeme odziva od zakretnih kristala. Najviše se primjenjuju LCD zasloni koji imaju pozitivnu i negativnu elektrodu pomoću kojih se mijenja stanje strukture kristala (engl. *in - plane switching*, IPS). Upravo je ova vrsta prikazana na slici Slika 2.1. [11].



Slika 2.1. Princip rada LCD zaslona [10]

² Optički filter kroz koji prolazi svjetlost nakon čega ona oscilira na jednak način.

Svrha LCD zaslona je nešto prikazati, vizualizirati. Dakle, koriste se u projektima u kojima želimo na izlaz poslati jasno vidljivu informaciju. Za ovaj rad korišten je tekstualni LCD zaslon, no osim njega postoje i grafički, ali o njima u ovom radu neće biti govora. Tekstualni LCD zasloni mogu se međusobno razlikovati. Tako postoje LCD zasloni sa različitim brojem redaka i stupaca, npr. LCD 1602 (16 stupaca i 2 retka) i LCD 2004 (20 stupaca i 4 retka). Također, osim osnovnih LCD zaslona koji imaju plavu pozadinu i bijela slova, postoje i LCD zasloni, takozvani RGB (engl. *Red Green Blue*) kod kojih se boja pozadine može mijenjati u bilo koju boju. Također, LCD zasloni se i po boji slova mogu razlikovati, primjerice mogu imati bijela ili crna slova. Primjeri različitih LCD zaslona prikazani su na slici Slika 2.2.

Što se tiče gumbi, postoje takozvani štitovi (engl. *KeyPad Shield*) koji su zapravo tiskane pločice (engl. *printed circuit board*, PCB) na kojima su i LCD modul i gumbi. Na njima su gumbi najčešće spojeni na jedan analogni pin. Naravno, gumbi ne trebaju biti na ovaj način povezani, već mogu biti i samostalni, odnosno ne trebaju biti na tiskanoj pločici već mogu biti direktno povezani na bilo koji izlazni pin mikroračunala pa tako i na digitalne pinove.



Slika 2.2. LCD 1602 sa plavom pozadinom i bijelim slovima (gornja slika) i LCD 2004 sa crvenom pozadinom i crnim slovima (donja slika) [12,13]

S obzirom da je u ovom radu napravljena biblioteka za izbornik na LCD zaslonu, u dalnjem tekstu navedeni su neki primjeri korištenja izbornika u stvarnim sustavima. LCD zaslon može se koristiti za pregled ili za konfiguraciju. Primjerice, možemo „šetati“ po izborniku i njegovim podizbornicima i pregledavati vrijednosti nekih varijabla ili pak neke poruke.

Primjerice mjerimo temperaturu i vlagu staklenika pomoću senzora spojenog na Arduino te želimo vrijednosti dobivene od senzora prikazati u izborniku. To možemo tako da u glavnom izborniku napravimo elemente „temperatura“ i „vlažnost“ te odabirom pojedinog elementa prikazujemo njihove vrijednosti na više različitih načine, točnije u različitim mjernim jedinicama (temperaturu možemo prikazati u stupnjima Celzijevih, stupnjima Farenheita i u Kelvinima). Svaki put kada se vrijednost temperature ili vlažnosti promijeni, element za prikaz određene vrijednosti koja se promijenila ažurirati će se. Ovo je bio primjer kako

koristiti izbornik za prikaz, a sada slijedi primjer kako bismo ga mogli koristiti za konfiguraciju. Pošto izbornik napravljen u ovom radu ima mogućnost povećavanja ili smanjenja vrijednosti cjelobrojnih i decimalnih elemenata izbornika pomoću gumbi, možemo kroz izbornik mijenjati vrijednost temperature. Izmijenjeni iznos temperature možemo poslati grijачima ili ventilatorima. Također, pošto izbornik ima i mogućnost višestrukog izbora od ponuđenih (engl. *checkbox*) kao i mogućnost jednog izbora od ponuđenih (engl. *radio*) možemo napraviti popis određenih temperatura te svaki put iznova izabirati jednu od njih i poslati ju grijачima ili ventilatorima. Još jedna ideja je dodati rasvjetu te odabirati želimo li ju ili ne želimo uključiti [14].

Drugi primjer je koristiti izbornik za postavljanje nekog alarma. Možemo napraviti popis narednih nekoliko godina, popis svih mjeseci u godini i cjelobrojni element koji će označavati dan, a moći ćemo ga mijenjati. Također, možemo još napraviti na sličan način i elemente za odabir sata i minute kada bi se alarm trebao upaliti.

Izbornik možemo koristiti i za IoT (engl. *Internet of things*) projekte. Ovo su bili samo neki od mnogobrojnih primjera u kojima napravljeni izbornik može biti koristan.

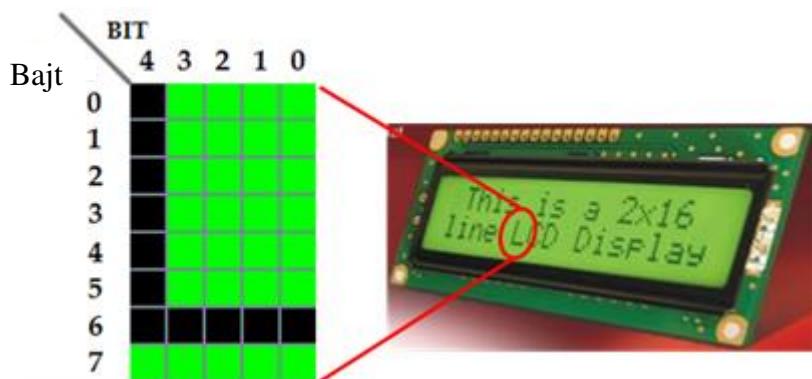
3. UREĐAJI POTREBNI ZA RAD S BIBLIOTEKOM

U ovom radu korišten je tekstualni LCD zaslon na kojemu se prikazuje izbornik, a kretanje po izborniku ostvareno je pomoću gumbi. LCD koji sam koristila je LCD 1602 koji se nalazi na tiskanoj pločici na kojoj su i gumbi. Na pločici je šest gumba: gumb za odabir (engl. *select*), za lijevo (engl. *left*), gore (engl. *up*), dolje (engl. *down*), desno (engl. *right*) i gumb za ponovno pokretanje (engl. *reset*). Osim LCD zaslona trebala sam i mikroračunalo za koje sam koristila Arduino Uno prvotno jer je LCD kompatibilan za njega, ali i jer je s njim vrlo jednostavno raditi na ovakvim projektima.

3.1. LCD ZASLON

LCD 1602 korišten u ovom radu Vellemanov³ je tekstualni LCD modul koji se nalazi na tiskanoj pločici na kojoj su osim njega i šest gumbi, potenciometar⁴ i led dioda (engl. *Light-Emitting Diode*, LED). Izgled LCD zaslona prikazan je na slici Slika 3.2. LCD je kompatibilan za ova mikroračunala: Arduino Uno, Mega, Diecimila, Duemilanove i Freeduino. Cijena ove pločice je 65 kn. Oznaka 1602 govori kako LCD ima 16 stupaca i 2 reda te prema tome u jednom trenutku može prikazivati najviše 32 znaka. Svaki znak prikazan je u matrici piksela veličine 5 stupaca i 8 redova, a moguće je napraviti i vlastiti znak unutar te matrice [2].

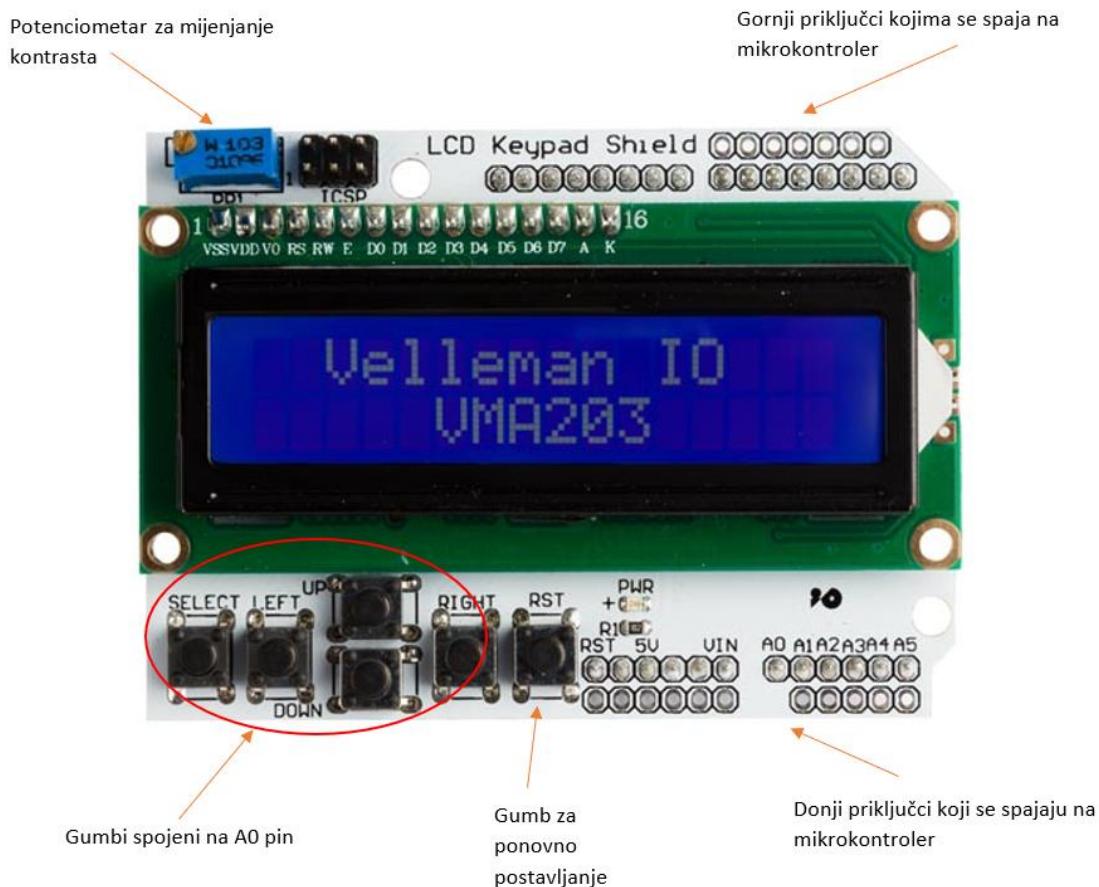
Vlastiti znakovi spremaju se u CGRAM (engl. *Character Generator Random Access Memory*) kojeg ovaj LCD, kao i svi tipa HD44780 imaju. Ova memorija veličine je 64 bajta te je prema tome moguće napraviti najviše 8 vlastitih znakova pošto jedna matrica veličine 5*8 piksela sadrži jedan znak. Iako se sastoji od 5 piksela, svaki redak matrice računalu je predviđen kao jedan bajt, a pošto redaka ima 8 to je ukupno 8*1 bajt, odnosno 8*8 bitova. Dakle, za prikaz jednog znaka potrebno je 64 bita, a pošto memorija ima 64 bajta, moguće je spremiti 8 takvih znakova. Kako će znak izgledati određuje se stavljanjem određenih bitova u 1 ili 0 čime piksele, na mjestima tih bitova u matrici, palimo ili ostavljamo ugašenima [1]. Kako to vizualno izgleda prikazano je na slici Slika 3.1.



Slika 3.1. Prikaz znaka u matrici 5*8 piksela [1]

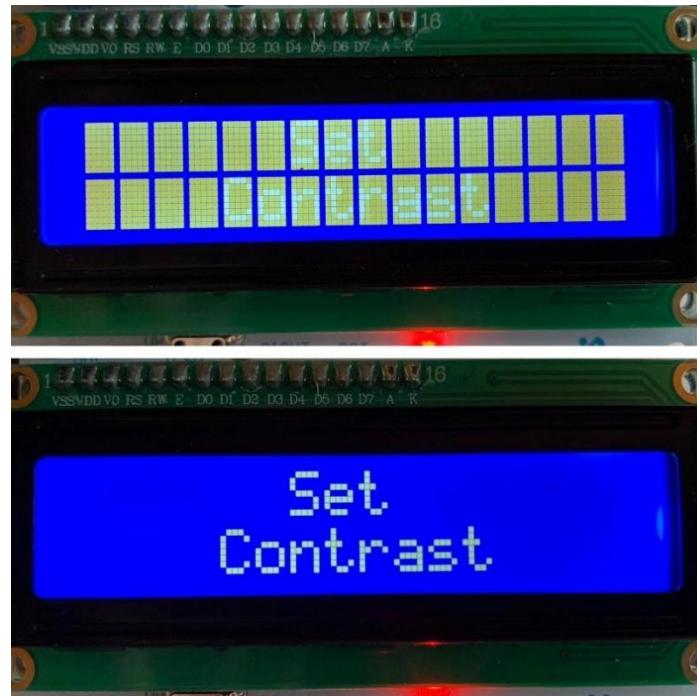
³ Velleman je kompanija koja proizvodi i distribuira elektroniku. Osnovana je 1975. godine u Belgiji. [15]

⁴ Potenciometar je promjenljivi otpornik s kliznim, linearnim ili okretnim spojem te s linearnom ili logaritamskom promjenom otpora. [16]



Slika 3.2. Tiskana pločica s LCD modulom i gumbima [3]

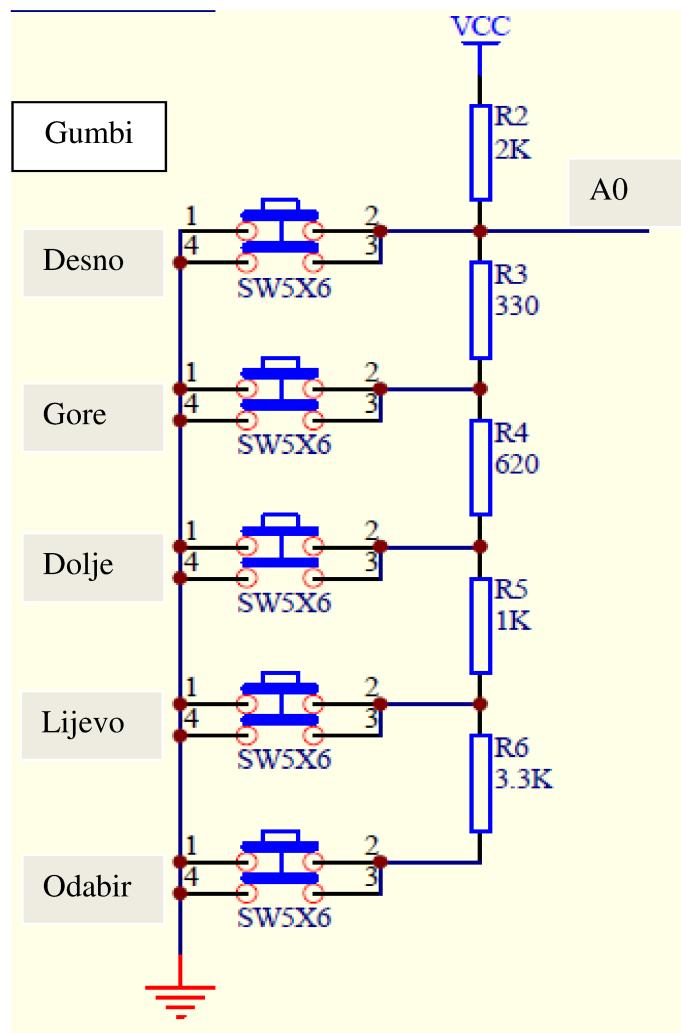
LCD ima LED bijelo pozadinsko osvjetljenje (engl. *backlight*), a pozadina zaslona je u plavoj boji te ima mogućnost ručnog mijenjanja kontrasta između znakova i pozadine pomoću potenciometra. Ukoliko potenciometar okrećemo obrnuto od smjera kazaljke na satu povećat ćemo kontrast znakova koji će biti jače izoštreni, a ukoliko okrećemo u smjeru kazaljke povećava se kontrast pozadine kada sve točkice piksela u matrici postaju sve jače vidljivije te se sve manje raspoznaće koji su pikseli trenutno aktivni, a koji nisu, odnosno znakovi postaju sve manje izoštreni. Kako koji kontrast izgleda možemo pogledati na slici Slika 3.3.



Slika 3.3. Prikaz različitih kontrasta dobivenih okretanjem potenciometra

Napon napajanja je 5 VDC (engl. *Voltage direct current*) što pokazuje ledica koja svijetli crveno ukoliko je LCD napajan.

Pet upravljačkih gumbi: odabir, lijevo, gore, dolje i desno spojeni su na jedan analogni ulazni pin A0 te se pomoću analogno-digitalnog pretvarača (engl. *Analog-to-Digital Converter*, ADC) dobije broj koji označava razinu napona pomoću kojega znamo koji gumb je trenutno aktivан. Broj koji označava razinu napona šalje se mikroračunalu na obradu, a dužnost programera je da na temelju tog broja odredi aktivan gumb. Koji broj označava koji gumb znati ćemo ako znamo kako su gumbi spojeni što vidimo na slici Slika 3.4. Tada znamo unutar kojeg intervala brojeva se nalaze koji gumbi, odnosno za svaki gumb znamo njegovu minimalnu i maksimalnu razinu napona na kojoj se može nalaziti [2].



Slika 3.4. Shema gumbi [2]

3.2. Arduino

Arduino je mikroračunalo koje se povezuje sa računalom putem univerzalne serijske sabirnice (engl. *Universal Serial Bus*, USB) kako bi se mogao programirati. Također, Arduino ima i druge utičnice na koje možemo priključiti mnoge druge komponente, kao što su LCD zasloni, senzori, zvučnici, mikrofoni, itd. Arduino sklopovlje je otvoreno (engl. *open*), odnosno slobodno dostupno svima. Jedino je zaštićeno ime Arduino zbog čega pločice zasnovane na Arduinu nose slične nazive poput Freeduino, Seeeduino, Boarduino, itd. Prvi Arduino nastao je 2005. godine, a od tada do danas razvile su se mnoge vrste Arduino pločica od kojih su neke od najpopularnijih: Arduino Uno, Arduino Leonardo, Arduino Mega, Arduino Due, itd. Za ovaj rad korišten je *Arduino Uno R3 SMD Edition* koji možemo vidjeti na slici Slika 3.5. U dalnjem tekstu ovu vrstu Arduino pločice referencirati ću samo sa Arduino iako će se misliti na *Arduino Uno R3 SMD Edition*. Ostale vrste Arduino pločica mogu se razlikovati po nekim komponentama. U nastavku će biti opisane najvažnije komponente Arduino pločice.

Arduino se temelji na mikrokontroleru ATmega328 koji je postavljen na sredinu pločice u čipu sa 32 pina. Čip sadrži procesor (engl. *Central Processing Unit*, CPU), 2 KB radne memorije (engl. *Static Random Access Memory*, SRAM) za skladištenje podataka, 1 KB trajne električno izbrisive programibilne ispisne memorije⁵ (engl. *Electricaly Erasable Programmable Read-Only Memory*, EEPROM), 32 KB brze memorije (engl. *Flash Memory*) u koju se pohranjuje program, ulazne i izlazne priključke te serijsko sučelje za podatke (engl. *Universal Asynchronous Receiver-Transmitter*, UART).

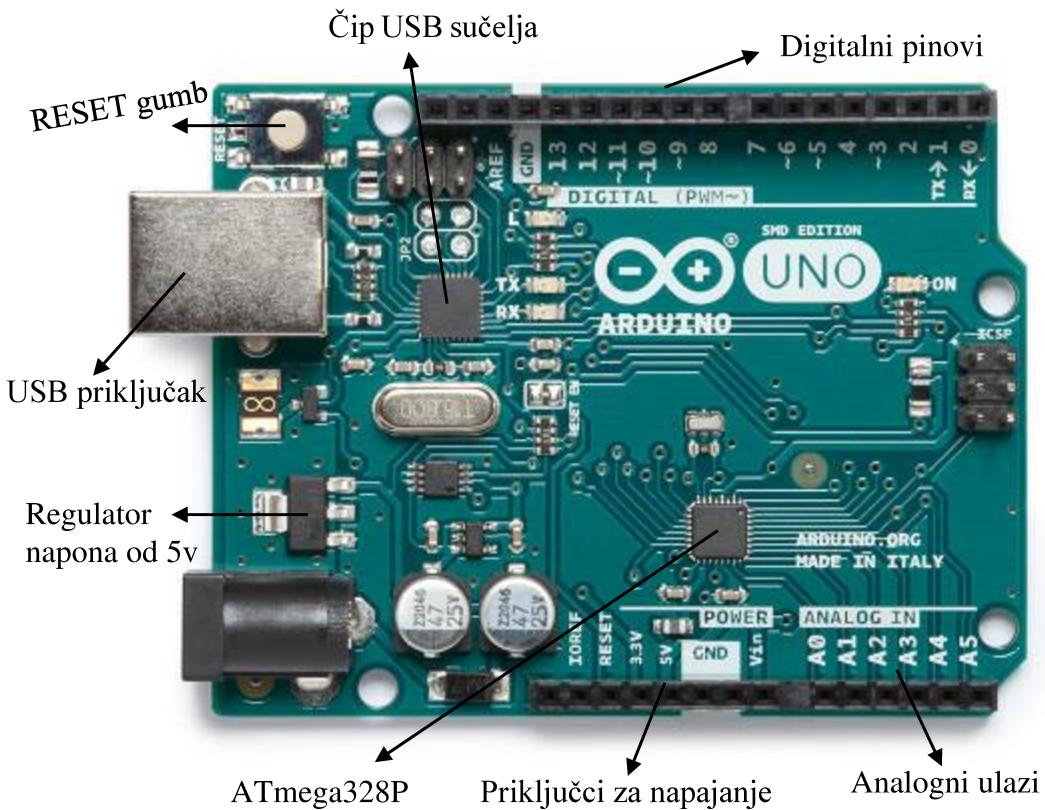
Što se tiče napajanja pločice, Arduino se može napajati putem USB priključka ili vanjskim napajanjem preko utičnice ili baterije. Vanjsko napajanje spaja se na pretvarač napona od 5 volti (engl. *AC/DC adapter*) koji pretvara svaki napon od 7 do 12 volti u konstantan napon od 5 volti. Arduino u teoriji može raditi na vanjskom napajanju od 6 do 20 volti, no ako ga se napaja s manje od 7 volti, 5V pin može napajati s manje od 5 volti pa Arduino može imati problema u radu. Također, ako ga se napaja s više od 12 volti, regulator napona može pregrijati i oštetiti Arduino pločicu.

Pinovi za napajanje su pin ulaznog napona (Vin) kada koristimo vanjski izvor napajanja, 5V pin, 3.3V pin, pin za uzemljenje (GND, 0V) i IOREF pin koji pruža referentnu vrijednost napona s kojom mikrokontroler radi. Među ovim pinovima nalazi se i pin *Reset* koji ima istu ulogu kao i gumb *Reset* koji služi za ponovno pokretanje programa Arduino pločice.

Pored pinova za napajanje nalaze se pinovi označeni sa A0-A5 koji predstavljaju analogne ulaze. Oni mijere napon od 0 do 5 volti te se kasnije izmjerena vrijednost može koristiti u programu. Svaki od njih pruža 10 bita razlučivosti, odnosno 1024 različitih vrijednosti. Kroz njih prolazi veoma slaba struja prema zemlji jer su njihovi unutrašnji otpori vrlo veliki. Iako su ovo analogni ulazi, mogu se koristiti i kao digitalni ulazi ili izlazi u smislu da možemo promatrati da li je neki pin uključen ili isključen.

Sa suprotne strane pločice nalaze se digitalni pinovi kojih ima 14, a mogu se koristiti i kao ulazi i kao izlazi. Pinovi 0 i 1 su pin za primanje (engl. *Receiving*, RX) i za prijenos (engl. *Transmitting*, TX). Oni se koriste kod serijske veze sa računalom za primanje i prijenos podataka. Pinovi 2 i 3 mogu se koristiti za vanjske prekide (engl. *Interrupt*), a pin 13 je spojen na ledicu koju regulira [8,9].

⁵ ROM koji se koristi za pohranjivanje male količine podataka čiji se podaci mogu električki brisati i ponovo programirati. Ciklusi čitanja i pisanja se izvode vrlo sporo u usporedbi sa RAM memorijom. [17]

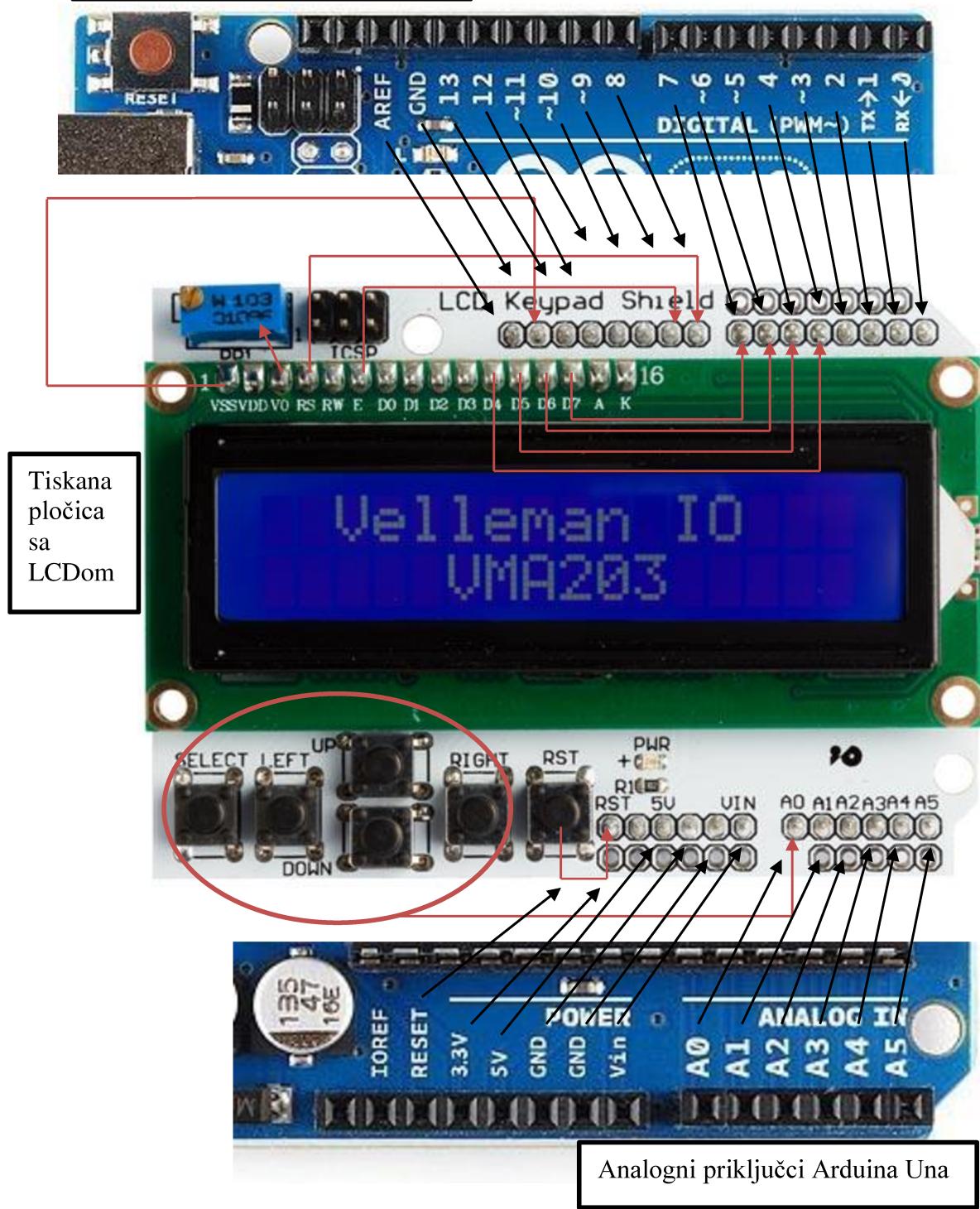


Slika 3.5. Arduino pločica [8]

3.3. Povezivanje Arduina i LCD zaslona

Pomoću ulaza, odnosno pet gumbi (uz šesti gumb za ponovno postavljanje (engl. *reset*)) upravlja se mikrokontrolerom, odnosno Arduinom, koji obrađene podatke šalje na izlaz, tj. LCD zaslon. Ostaje pitanje kako se gumbi i LCD zaslon spajaju sa Arduinom. Na slici Slika 3.6. prikazano je kako su spojeni neki važniji pinovi s LCD modula na tiskanu pločicu te s tiskane pločice na Arduino. Opis slike, odnosno spajanja te značenje pojedinih pinova opisano je u tablici Tablica 3.1. Pinovi koji nemaju nikakvu ulogu ili koji nisu spojeni, nisu ni u tablici. Ovaj LCD zaslon koristi 4 bita za prijenos podataka (D7-D4 pinovi), no može se koristiti i 8 bitova (D7-D0 pinovi).

Digitalni priključci Arduina Una



Slika 3.6. Spajanje tiskane pločice s LCD modulom i s Arduinom Unom

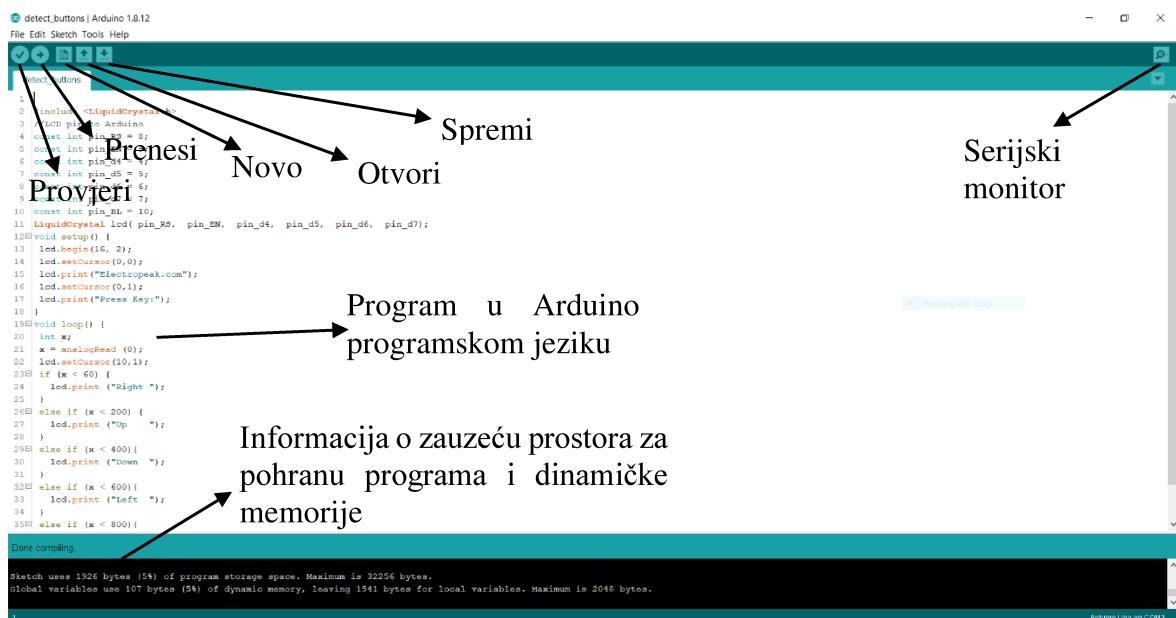
Tablica 3.1. Spajanje tiskane pločice i Arduina Una. Pinovi se počinju gledati od gornjih (digitalnih) pinova Arduina sa lijeve strane. [2]

Arduino Uno pinovi	LCD pinovi, pinovi tiskane pločice	Uloga pina
GND	VSS	Uzemljenje; negativno napajanje
Digitalni pin 10	Pin za pozadinsko osvjetljenje (engl. <i>Backlight</i>)	Omogućava ili onemogućava pozadinsko osvjetljenje, a može i mijenjati svjetlinu ekrana
Digitalni pin 9	E (engl. <i>Enable</i>)	Omogućuje čitanje/pisanje podataka
Digitalni pin 8	RS (engl. <i>Register Select</i>) pin	Odabire da li će se podatak na sabirnici podataka gledati kao instrukcija ili podatak
Digitalni pin 7	D7	Pin sabirnice podataka za prijenos podataka između Arduina i LCD zaslona
Digitalni pin 6	D6	Pin sabirnice podataka za prijenos podataka između Arduina i LCD zaslona
Digitalni pin 5	D5	Pin sabirnice podataka za prijenos podataka između Arduina i LCD zaslona
Digitalni pin 4	D4	Pin sabirnice podataka za prijenos podataka između Arduina i LCD zaslona
RESET	RST	Za ponovno postavljanje, aktivira se pritiskom gumba RST
5V (VCC)	5V	Pozitivno napajanje, +5V
GND (VSS)	GND	Negativno napajanje, uzemljenje
Analogni ulaz A0	A0	Pin na koji su spojeni svi gumbi

3.4. Arduino IDE

Arduino IDE (engl. *Integrated Development Environment*) razvojno je okruženje otvorenog koda pomoću kojeg se programira Arduino mikrokontroler. Radi na Windowsima, Mac operacijskom sustavu i Linuxu. Može se instalirati na računalo te koristiti bez pristupa Internetu ili se može koristiti *Arduino Web Editor* ukoliko smo na mreži (engl. *online*). Trenutno najnovija verzija je Arduino 1.8.12 izšla 13.02.2020. godine. Kompatibilan je sa bilo kojom Arduino pločicom. Kodove u ovom razvojnog okruženju pišemo u Arduino programskom jeziku koji je zapravo programski jezik C++ sa nekim dodatnim funkcijama, klasama, itd. te se taj kod vrlo lagano prenosi na pločicu [4].

Kako je za ovaj rad korišten Arduino Uno, u nastavku će biti opisano kako se on programira pomoću Arduino IDE-a, no slično vrijedi i za ostale. Kako bismo Arduino mogli programirati moramo ga spojiti na računalo. Arduino spajamo USB A/B kabelom s računalom s kojega Arduino automatski crpi napajanje što možemo primijetiti po tome što se zelena ledica (PWR) upalila, isto kao i crvena LCD ledica. Nakon toga treba instalirati upravljačke programe (engl. *drivers*) što će se automatski učiniti. Sada je vrijeme za napisati Arduino skicu (engl. *sketch*), odnosno program, a kako napisati program za LCD zaslon biti će objašnjeno kasnije. Nakon što imamo napisani program moramo odabratи Arduino pločicu koju koristimo u Alati (engl. *Tools*) → Pločica (engl. *Board*) te u popisu pronađemo Arduino Uno. Osim Arduino pločice, moramo odabratи i priključak (engl. *port*) na koji je uređaj spojen u Alati → Priključak → Serijski priključci. Sada možemo prevesti (engl. *compile*) program pritiskom na Provjeri (engl. *Verify*). Sada će prevodilac vratiti greške ili ukoliko je sve u redu napisati će koliko prostora (u bajtovima) program zauzima te koliko prostora zauzimaju globalne varijable u dinamičkoj memoriji. Kako Arduino IDE izgleda te gdje se nalaze spomenuti alati, prikazano je na slici Slika 3.7. Ukoliko smo zadovoljni svime do sada, možemo napokon provjeriti kako naš program radi na Arduino pločici. Kako bismo program prenijeli na pločicu, stisnuti ćemo Prenesi (engl. *Upload*). Sada možemo vidjeti kako ledice RX (engl. *Receiving*) i TX (engl. *Transmitting*) trepere pošto se ti pinovi koriste tijekom serijskog prijenosa podataka. Nakon što je prijenos završio Arduino program bi trebao izvršavati svoju funkciju [5].



Slika 3.7. Arduino IDE

Serijski monitor koristi se za interakciju korisnika s Arduino pločicom te uvelike pomaže u otklanjanju pogrešaka pošto Arduino zasad još uvijek nema program za ispravljanje pogrešaka (engl. *debugger*) [6].

Ostaje još pitanje kako napisati program u Arduinu za LCD. Arduino, kao i mnogi drugi programski jezici, sadrži biblioteke kojima proširuje svoju funkcionalnost. Neke osnovne biblioteke već su dodane u Arduino IDE, no uvijek možemo dodati nove ili pak stvoriti vlastite što je upravo napravljeno u ovome radu. Svaka biblioteka sastoji se od barem dvije datoteke, a to su zaglavna datoteka (engl. *header file*) i izvorna datoteka (engl. *source file*). Zaglavna datoteka sastoji se od definicija, odnosno sadrži popis javnih i privatnih metoda i varijabla klase koja mora biti istog imena kao i sama datoteka. S druge strane, izvorna datoteka sadrži „pravi“ kod, odnosno napisane funkcije koje su u zaglavnoj datoteci samo definirane. Biblioteka mora imati i tekstualnu datoteku naziva `keywords.txt` u kojoj su definirane ključne riječi, točnije klase i funkcije, kako bi u Arduino IDE-u one bile obojane pošto Arduino IDE ovo ne radi automatski. Također, svaka biblioteka sadrži i primjere korištenja koje možemo pronaći u Datoteka (engl. *File*) → Primjeri (engl. *Examples*) [6]. Najjednostavniji način za dodati biblioteku u Arduino IDE je kopirati ju u `../Arduino/libraries` te ponovno pokrenuti Arduino IDE. Sada bi biblioteka trebala biti dodana što se može provjeriti otvaranjem Datoteka → Primjeri → Ime_biblioteke. Biblioteka koja se najčešće koristi za tekstualne LCD zaslone je `LiquidCrystal`. Ova biblioteka kompatibilna je za sve LCD Hitachi HD44780 zaslone, a radi i sa 4 i sa 8 bita za prijenos podataka [7].

Program započinje uključivanjem biblioteka, konkretno `LiquidCrystal` biblioteke. Nakon toga definirane su određene varijable, instance klasa, itd. koje trebaju biti dostupne cijelom programu. Ovdje se pomoću konstruktora klase `LiquidCrystal` stvara `lcd`, a konstruktoru su poslani brojevi pinova potrebni za stvaranje instance. Sada dolazi funkcija postavljanja (engl. *setup*) koju svaki Arduino program mora imati. Ona se izvršava samo jednom i to prilikom pokretanja te se u njoj uglavnom inicijaliziraju varijable, odnosno definiraju početna stanja. U slučaju s LCD zaslonom ovdje se poziva funkcija započni (engl. *begin*) kako bi se definirali broj redaka i stupaca LCD zaslona. Još je jedna funkcija koju sadrži svaki Arduino program, a to je funkcija petlje (engl. *loop*) koja se izvodi beskonačno na Arduino mikrokontroleru. Ovdje se primjerice može koristiti funkcija za postavljanje kursora (engl. *setCursor*) koja postavlja cursor na određeno mjesto na zaslonu te se pomoću funkcije za ispis (engl. *print*) na zaslonu nešto i prikazuje. Kako bi izgledao ovaj jednostavni Arduino program prikazano je u kodu Isječak koda 3.1.

Isječak koda 3.1. Primjer jednostavnog Arduino programa

```
#include <LiquidCrystal.h>
const int RS = 8;
const int EN = 9;
const int d4 = 4;
const int d5 = 5;
const int d6 = 6;
const int d7 = 7;
LiquidCrystal lcd( RS, EN, d4, d5, d6, d7);

void setup() {
    lcd.begin(16, 2);
}
void loop() {
    lcd.setCursor(0, 1);
    lcd.print("hello, world!");
}
```

4. OSTVARENJE BIBLIOTEKE ZA IZBORNIK NA LCD ZASLONU

U ovome poglavlju opisana je detaljnije biblioteka pomoću koje je izrađen izbornik za LCD zaslone. Biblioteka je napisala u C++ programskom jeziku i slijedi upute Arduina za pisanje vlastitih biblioteka koje se potom dodaju u Arduino IDE. Dakle, biblioteka sadrži zagлавnu datoteku `MenuPack.h` koja se sastoji od definicija, odnosno sadrži popis javnih i privatnih metoda i varijabla klase `MenuPack`. Sve funkcije koje bi korisnik mogao koristiti u svome programu su javne. Također, biblioteka ima i izvornu datoteku `MenuPack.cpp` koja sadrži „pravi“ kod, odnosno napisane funkcije koje su u zaglavnoj datoteci samo definirane.

Biblioteka ima i tekstualnu datoteku naziva `keywords.txt` u kojoj su definirane ključne riječi, točnije klase i funkcije, kako bi u Arduino IDE-u one bile obojane. Također, biblioteka sadrži i 6 primjera koja su detaljnije opisana u poglavlju 5 [18].

Ostale datoteke koje biblioteka sadrži napravljene su kako bi cijela biblioteka bila preglednija te kako osnovne dvije datoteke, `MenuPack.h` i `MenuPack.cpp`, ne bi bile prenatrpane. Sve datoteke će u nastavku biti detaljnije objašnjene.

4.1. Konfiguracijska datoteka

Konfiguracijska datoteka je zapravo obična zagлавna datoteka koju korisnik mora otvoriti te definirati specifične parametre za svoj LCD zaslon. Parametri će već biti definirani za LCD koji je korišten u ovome radu, ali ukoliko korisnik ima drugačije parametre ili ih uopće nema, treba ih izmijeniti ili izbrisati njihove vrijednosti. Sve informacije kako popuniti konfiguracijsku datoteku da program radi ispravno, nalaze se u komentarima u samoj datoteci kako bi se korisnik lakše snalazio.

Na početku datoteke je inicijalizacija gumbi, odnosno korisnik mora odrediti koliko gumbi ima (minimalno je potrebno imati dva gumba za kretanje po izborniku), da li su gumbi digitalni ili analogni te na kojem pinu su spojeni. Ukoliko gumb za odabir ne postoji neće se moći ulaziti u podizbornike, a ukoliko gumb za vraćanje u roditeljski izbornik ne postoji moći će se napraviti element u izborniku koji će imati tu funkciju. Također, gumb za desno, isto kao i gumb za vraćanje, nije nužan jer se može napraviti element u izborniku koji će imati funkciju gumba za desno, odnosno „skoka“ na vrh izbornika.

Ako su gumbi koje koristimo analogni i spojeni na jedan pin, za svaki gumb postoje intervali koji označavaju koji gumb je aktivan te se ti intervali, odnosno minimalne i maksimalne vrijednosti za svaki takav gumb trebaju upisati.

Nakon inicijalizacije gumbi, slijedi odabir znaka kojeg želimo koristiti kako bi znali na kojem elementu se trenutno nalazimo te znak koji će se pojaviti ukoliko je neki element odabran u izborniku s mogućnošću višestrukog ili jednostrukog odabira.

Posljednja stvar koja se mora definirati je broj redaka i stupaca LCD zaslona kojeg koristimo.

Konfiguracijska datoteka ostvarena je kao definicija makro konstanti, a u Isječak koda 4.1. možemo vidjeti kako ona izgleda na dijelu odabira znakova.

Isječak koda 4.1. Odabir znakova u konfiguracijskoj datoteci

```
//an arrow that indicates which element we are currently on
#define SELECTED_ARROW ">"

//arrow indicating which element is marked in checkbox or radio
#define RADIO_CHECKBOX_ARROW "x "
```

4.2. Implementacija

Ranije je već spomenuto kako se biblioteka sastoji od dvije glavne datoteke: zaglavne datoteke `MenuPack.h` i izvorne datoteke `MenuPack.cpp`. Međutim, one će biti kasnije objašnjene jer je prvo važnije objasniti kako su gumbi ostvareni.

Funkcionalnost gumbi objedinjena je klasom `Buttons`, a važna je i enumeracija `Button` koja sadrži nazine pet gumbi koliko ih je moguće maksimalno imati. U enumeraciji je dodan i `BT_NONE` koji predstavlja da ni jedan gumb nije trenutno aktivran. Klasa sadrži dva polja (engl. *arrays*), od kojih jedno predstavlja analogne, a drugo digitalne pinove na koje mogu biti spojeni gumbi. Konstruktorom klase `Buttons` ta se polja inicijaliziraju na vrijednost 100 iz razloga jer taj pin nikada neće postojati, a pomoću funkcije `check_the_existence_of_buttons` provjerava se koji gumbi postoje, koji su analogni, a koji digitalni (pomoću podataka iz konfiguracijske datoteke) te na temelju toga se popunjavaju prethodno spomenuta dva polja. Funkcija `detectButton` vraća trenutno aktivran gumb koji je lakše saznati ukoliko su gumbi digitalni pošto će tada svi biti spojeni na različite pinove te u tom slučaju samo treba pregledati da li postoje vrijednosti različite od 100 u polju koje predstavlja digitalne pinove. Ukoliko su gumbi pak spojeni na analogni pin, svi će biti spojeni na isti pin, ali će se dobivati različite vrijednosti iz nekog intervala vrijednosti za svaki gumb. Iz ovog razloga poziva se dodatna funkcija `analog_detect_btn` koja pomoću minimalnih i maksimalnih vrijednosti za svaki gumb, danih u konfiguracijskoj datoteci, raspoznaće koji gumb je aktivran. Funkcije pomoću kojih se određuje da li je neki gumb aktivran ili nije su `analogRead` i `digitalRead` Arduino programskog jezika.

Nakon objašnjenja implementacije gumbi, važno je objasniti kako radi klasa `MenuPack` koja predstavlja objedinjenje svega što izbornik sadrži. Sastoji se od mnogo funkcija od kojih se neke pozivaju samo unutar klase te su one privatne, a ostale koje možemo pozivati iz Arduino programa su javne. Prije objašnjenja funkcija važno je reći da postoje strukture `lcd`, `menu` i `menu_element` te enumeracija `MenuItemType` kojom su definirane sve moguće vrste elemenata. Struktura `lcd` sadrži broj redaka i stupaca LCD zaslona. Struktura `menu` prikazana je u Isječak koda 4.2.

Isječak koda 4.2. Prikaz strukture izbornika

```
struct menu {
    String arrow; //an arrow to the element we are on

    // at the beginning we are on the first element
    int current_element = 0;
    int current_row = 0; //current row on LCD
    int max_menu_elements;
    menu_element **menuElements; // array of menu elements

    // if menu is a sub menu then there is a parent in the higher
    // menu so it can go back to it
    menu_element *parent;
    String radio_checkbox_arrow; //if menu is radio or checkbox

    // whether one value in the radio is selected (if the menu is
    // radio)
    bool radio_selected = false;
};
```

Struktura `menu_element` i enumeracija `MenuItemType` prikazane su u Isječak koda 4.3.

Isječak koda 4.3. Prikaz strukture elementa

```
enum MenuElementType {
    ME_NAME, // Menu name plus description
    ME_MENU, // a basic member of the menu with selected arrow
    ME_PARENT, // clicking on it enters the submenu
    ME_BACK, // by clicking on it we return to the previous menu
    ME_RADIO, // radio element (only one value can be selected)
    ME_CHECKBOX, // checkbox element (more values can be selected)
    ME_INT, // int value which can change
    ME_FLOAT, // float value which can change
    ME_TOP // returns to the top of the menu
};

union change_value { // only one struct is in memory
    struct change_int_value {
        int value; // current value
        int new_value; // changed value
        int min; // minimum value
        int max; // maximum value
        int increment;
    } int_value;
    struct change_float_value {
        float value;
        float new_value;
        float min;
        float max;
        float increment;
    } float_value;
};

struct menu_element {
    String name;
    MenuElementType type;
    // a function that is performed after selecting element
    void (*actionFunctionFirst)();

    // a function that is performed in a loop after selecting an
    // element
    void (*actionFunctionSecond)();

    // used only if the element is a parent; true if you want
    // to return from sub to the previous menu
    bool back = false;

    // if the item is inside the radio or checkbox indicates whether
    // it is selected
    bool radio_checkbox_selected = false;

    change_value value;
    // whether the element is in a state of change
    bool change_value = false;
    // a function that is performed while changing an element
    void (*actionFunctionWhenChangeValue)();
    bool action_function_when_change_value_exist = false;
};
```

Kao što je već napomenuto, klasa `MenuPack` sadrži mnoge funkcije. Njihova implementacija neće biti detaljno opisana, već će biti opisana samo funkcionalnost nekih važnijih funkcija. Njihova implementacija može se naći u kodovima danim uz ovaj rad.

Od izrazitog je značaja funkcija `checkButtons` kojom se provjerava da li je neki i koji je gumb pritisnut. Ukoliko funkcija prepozna da je neki gumb aktivan, pozvati će se funkcija `menu_event` koja će pronaći aktivan gumb te izvesti određenu funkciju za svaki gumb. Koja je funkcionalnost kojeg gumba opisana je u pseudokodu Isječak koda 4.4.

Isječak koda 4.4. Pseudokod funkcije `menu_event`

```

switch (button) {
    case SELECT:
        switch (current_element_type) {
            case PARENT:
                enter the sub menu and performs functions in the loop
                if (current_element_back active) break loop;
            case BACK:
                set parent_element_back to true
            case CHECKBOX:
                if (element is unselected) select it
                else unselect it
            case RADIO:
                if (no element is selected in menu) select it
                if (current_element is selected) unselect it
                else no change
            case INT and FLOAT:
                if (changing state not active) activate it
                else value set to the changed value (new_value)
            case TOP:
                reset menu (go to the top of menu)
        }
    case BACK:
        if ((INT or FLOAT type) and changing state is active)
            changing state set to not active
        else set parent_element_back to true
    case UP:
        if ((INT or FLOAT type) and changing state is active)
            increment value
        else go to the higher element in menu
    case DOWN:
        if ((INT or FLOAT type) and changing state is active)
            decrement value
        else go to the lower element in menu
    case RIGHT:
        reset menu (go to the top of menu)
}

```

Važno je napomenuti kako prilikom mijenjanja cijelobrojnih i decimalnih vrijednosti pritiskom prvi puta na gumb za odabir ulazimo u stanje za mijenjanje u kojemu vrijednost mijenjamo gumbima za gore i dolje te ukoliko želimo izabrati novu vrijednost moramo stisnuti ponovno gumb za odabir jer će se inače vrijednost vratiti na staru nepromijenjenu.

Još jedna važnija funkcija je `updateBuffer`. Ona se poziva svaki put kada želimo promijeniti ono što želimo ispisati na LCD zaslon. Dakle, poziva se kada prvi put uđemo u izbornik, kada se krećemo po izborniku, ulazimo u podizbornike, izlazimo iz njih, povećavamo ili smanjujemo vrijednosti, itd. Uglavnom, kada god LCD zaslon treba registrirati neku promjenu.

5. PRIMJERI KORIŠTENJA BIBLIOTEKE

Biblioteke pisane za Arduino moraju imati i neke primjere kako bi budućim korisnicima bilo lakše koristiti tu biblioteku. Za svoju biblioteku napravila sam 6 primjera koja ispituju sve funkcionalnosti biblioteke.

5.1. Primjer osnovnog izbornika

Ovaj se primjer nalazi u datoteci `MenuPack/examples/sub_menu_test/sub_menu_test.ino`. Njime se želi pokazati kako napraviti osnovni izbornik i kako dodati podizbornik. Također, pokazuje se na koji način se pomoću elementa vraćamo u prethodni izbornik i kako pomoću elementa možemo „skočiti“ na vrh izbornika.

Program započinje uključivanjem svih potrebnih biblioteka, a to su: Arduino, LiquidCrystal i MenuPack. Nakon toga deklarirani su pokazivači (engl. *pointers*) na klase LiquidCrystal i MenuPack kako bi se funkcije tih klasa mogle koristiti dalje u programu. Također, deklariramo i pokazivače `main_menu` i `sub_menu` na strukturu `menu` jer u programu postoje glavni izbornik i podizbornik.

U `setup` funkciji najvažniji je dio programa, a to je stvaranje i oblikovanje izbornika. Prvo je stvoren osnovni izbornik funkcijom `makeMenu` sa najviše 6 elemenata. Nakon toga radimo element vrste `ME_NAME` koji će predstavljati naslov glavnog izbornika te ostale elemente vrste `ME_MENU` koji predstavljaju obične elemente izbornika kojima je jedino svojstvo da se po njima može „šetati“. U glavni izbornik moramo dodati i element vrste `ME_PARENT` koji predstavlja roditelja na koji će se moći vratiti kada izađemo iz podizbornika. Tom elementu pridružujemo `actionFunctionFirst` koja se izvršava kada prvi puta uđemo u podizbornik te `actionFunctionSecond` koja se izvršava u petlji sve dok ne izađemo iz podizbornika. Važno je napomenuti da je uvijek potrebno prvo napraviti cijeli element, odnosno pridružiti mu vrijednosti koje trebamo, a tek onda taj element dodati u izbornik. Na kraju ne smijemo zaboraviti dodati roditeljski element podizborniku kako bi podizbornik znao gdje se vratiti izlaskom iz njega. Zadnji element koji je dodan u glavni izbornik je vrste `ME_TOP` i njegovim odabirom vraćamo se na naslov glavnog izbornika. Podizbornik koji je stvoren ima 4 elementa od kojih su svi `ME_MENU` jedino je zadnji `ME_BACK` čijim odabirom izlazimo iz podizbornika.

Na kraju `setup` funkcije mora se pozvati funkcija `print` koja vraća `buffer` prvi puta kada uđemo u neki izbornik.

U `loop` funkciji samo pozivamo `print`, s tim da kao parametar šaljemo `bool false` koji označava da se funkcija `print` ne poziva prvi puta.

Funkcija `print` mora biti omotana novom funkcijom koja koristi metode klase LiquidCrystal i brine o tome da se `buffer` ispravno ispisuje na LCD zaslon. Na slici Slika 5.1. možemo vidjeti vrh izbornika gdje vidimo naslov i prvi element te sredinu izbornika gdje vidimo roditeljski element čijim odabirom ulazimo u podizbornik. Također, kako izgleda ovaj kod možemo vidjeti u Isječak koda 5.1.

Isječak koda 5.1. Primjer osnovnog izbornika

```
#include "Arduino.h"
#include <LiquidCrystal.h>
#include <MenuPack.h>
```

```

// definition of pins for LCD
#define pin_RS 8
#define pin_EN 9
#define pin_d4 4
#define pin_d5 5
#define pin_d6 6
#define pin_d7 7

LiquidCrystal *my_lcd = new LiquidCrystal(pin_RS, pin_EN, pin_d4,
                                         pin_d5, pin_d6, pin_d7);
MenuPack *menu_pack = new MenuPack;
menu *main_menu = new menu;
menu *sub_menu = new menu;

void setup() {
    my_lcd->begin(16, 2); // the number of columns and rows of LCD
    main_menu = menu_pack->makeMenu(6); // main menu with 6 items
    menu_element *element = new menu_element;
    // adding a menu title
    element = menu_pack->makeMenuElement("Main menu", ME_NAME);
    menu_pack->addElementToMenu(element, main_menu);
    element = menu_pack->makeMenuElement("Configuration", ME_MENU);
    menu_pack->addElementToMenu(element, main_menu);

    sub_menu = menu_pack->makeMenu(4); // sub menu with 4 items
    element = menu_pack->makeMenuElement("Low", ME_MENU);
    menu_pack->addElementToMenu(element, sub_menu);
    element = menu_pack->makeMenuElement("Medium", ME_MENU);
    menu_pack->addElementToMenu(element, sub_menu);
    element = menu_pack->makeMenuElement("High", ME_MENU);
    menu_pack->addElementToMenu(element, sub_menu);
    element = menu_pack->makeMenuElement("Back", ME_BACK);
    menu_pack->addElementToMenu(element, sub_menu);

    // adding parent element (of sub menu) to main menu
    element = menu_pack->makeMenuElement("LCD Contrast", ME_PARENT);
    menu_pack->setActionFunctionFirst(element,
                                      [](){firstPrint(sub_menu);});
    menu_pack->setActionFunctionSecond(element,
                                      [](){loopPrint(sub_menu);});
    menu_pack->addElementToMenu(element, main_menu);
    menu_pack->addParentElement(element, sub_menu);

    element = menu_pack->makeMenuElement("Brightnesses", ME_MENU);
    menu_pack->addElementToMenu(element, main_menu);
    element = menu_pack->makeMenuElement("Dark mode", ME_MENU);
    menu_pack->addElementToMenu(element, main_menu);
    element = menu_pack->makeMenuElement("On top", ME_TOP);
    menu_pack->addElementToMenu(element, main_menu);

    firstPrint(main_menu); // print menu on LCD
}

void loop()
{
    loopPrint(main_menu); // print menu in loop
}

void firstPrint(menu *menu) {
    printEachRow(menu_pack->print(menu, true));
}

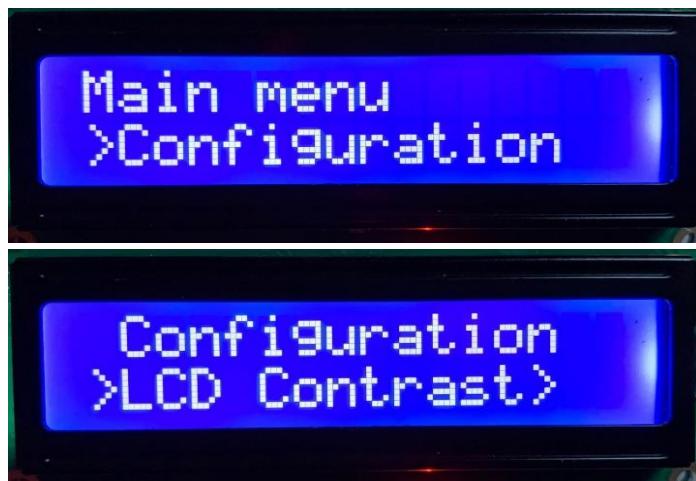
```

```

void loopPrint(menu *menu) {
    if(menu_pack->checkButtons(menu)) { // find the currently active button
        printEachRow(menu_pack->print(menu, false)); // print a buffer
    }
    delay(100); // I put it on because the loop was spinning too fast
                // so when it cleared the screen and re-wrote you could see
                // flickering.
                // With this delay there is no problem when checking
                // buttons.
}

void printEachRow(String buffer) {
    my_lcd->clear(); // clear LCD
    my_lcd->setCursor(0,0);
    my_lcd->print(buffer);
    if(buffer.length()>16){ // because we have 2 rows and 16 columns
        my_lcd->setCursor(0,1); // set cursor to second row
        my_lcd->print(buffer.substring(16));
    }
}

```



Slika 5.1. Izgled izbornika na LCD zaslonu; vrh izbornika (gornja slika) i sredina izbornika (donja slika)

5.2. Primjer s višestrukim i jednostrukim odabirom

Prvo ćemo pogledati primjer s višestrukim odabirom (engl. *checkbox*). On se nalazi u datoteci `MenuPack/examples/checkbox_test/checkbox_test.ino`. Ovim primjerom želi se demonstrirati rad izbornika u kojem postoji mogućnost višestrukog odabira elemenata te pokazati na koji način možemo kasnije koristiti odabrane elemente.

U glavnom izborniku postoje dva podizbornika, a to su `checkbox` u kojemu su elementi vrste `ME_CHECKBOX` i `selected_checkbox` u kojemu su prikazani trenutno odabrani elementi iz izbornika `checkbox`.

Dakle, u izborniku `checkbox` odabiremo jedan ili više elemenata te svaki put kada odaberemo neke druge elemente, izbornik `selected_checkbox` mora se ažurirati na nove

odabrane elemente što je zapravo najbitniji dio programa u ovome primjeru. Ključni dio nalazi se u funkciji `actionFunctionFirst` roditeljskog elementa izbornika `selected_checkbox`. Ta funkcija će se izvesti svaki puta kada uđemo u izbornik `selected_checkbox`, a na koji način je ona ostvarena možemo pogledati u Isječak koda 5.2.

Isječak koda 5.2. Funkcija koja se izvodi kada uđemo u podizbornik u kojemu su prikazani odabrani elementi iz podizbornika checkbox

```
void firstPrintSelectedCheckbox(menu *selected_checkbox, menu *checkbox)
{
    //resets the menu so that by re-entering in it we would be at
    //the beginning of the menu and not where we stopped
    menu_pack->resetMenu(selected_checkbox);

    //delete all elements except the title (zero element)
    menu_pack->deleteMenuItem(selected_checkbox, 2);
    menu_pack->deleteMenuItem(selected_checkbox, 1);

    //fetch new elements that are selected in the checkbox
    String **elem = menu_pack->getSelectedFromCheckbox(checkbox);
    while(*elem) {
        menu_element *element = new menu_element;
        element = menu_pack->makeMenuItem(**elem, ME_MENU);
        //adding new element to the menu
        menu_pack->addElementToMenu(element, selected_checkbox);
        elem++;
    }

    //print menu to screen
    printEachRow(menu_pack->print(selected_checkbox, true));
}
```

U sličnom primjeru `MenuPack/examples/radio_test/radio_test.ino` razlika je jedino što u prvom podizborniku umjesto elemenata vrste `ME_CHECKBOX` imamo elemente vrste `ME_RADIO` što znači da možemo odabrati samo jedan element iz tog podizbornika. Odabrani element prikazujemo u podizborniku `selected_radio`, a funkcija koja se izvodi prilikom ulaska u taj podizbornik vrlo je slična funkciji u Isječak koda 5.2. Jedina je razlika što umjesto polja elemenata, kojeg dobijemo kada tražimo odabrane elemente, dobiti ćemo samo jedan odabrani element.

Na slici Slika 5.2. možemo vidjeti kako izgleda izbornik `checkbox` u kojemu je moguće odabrati jedan ili više elemenata i izbornik `radio` u kojemu je moguće odabrati samo jedan element.



Slika 5.2. Izbornik u kojemu je moguće odabrati jedan ili više elemenata (gornja slika) i izbornik u kojemu je moguće odabrati samo jedan element (donja slika)

5.3. Primjer promjene cjelobrojnih i decimalnih vrijednosti

Ovdje su nam značajna zapravo dva primjera koja se nalaze u datotekama `MenuPack/examples/change_value_test/change_value_test.ino` i `MenuPack/examples/change_value_one_row_test/change_value_one_row_test.ino`. Razlika između njih je što prvi primjer prikazuje vrijednost koju mijenjamo u zasebnom retku, dok drugi primjer tu vrijednost stavlja u isti redak s imenom te vrijednosti. Drugi primjer je pregledniji ukoliko nam naziv i sama vrijednost stanu u jedan redak kao što je slučaj ovdje.

Prođimo prvo kroz prvi primjer. U glavnому izborniku ponovno imamo dva podizbornika, a to su `change_value` u kojemu je jedan cjelobrojni element vrste `ME_INT` i jedan decimalni element vrste `ME_FLOAT` te podizbornik `selected_values` u kojemu prikazujemo trenutnu cjelobrojnu i trenutnu decimalnu vrijednost. Ovaj podizbornik se ažurira svaki puta kada uđemo u njega kako bi prikazivao točne trenutne vrijednosti. Ažuriranje je implementirano pomoću funkcije `actionFunctionFirst` roditeljskog elementa izbornika `selected_values`, a ostvarena je na vrlo sličan način kao i funkcija u Isječak koda 5.2.

Bitna stavka za ovaj primjer je da kada radimo element vrste `ME_INT` obavezno kao ime elementa pošaljemo prazan `String` jer na taj način vrijednost stavljamo u zaseban redak:

```
element = menu_pack->makeMenuItem("", ME_INT);
```

Također, ne smijemo zaboraviti inicijalizirati vrijednost tako da joj odredimo redom početnu vrijednost, minimalnu vrijednost, maksimalnu vrijednost i vrijednost povećanja:

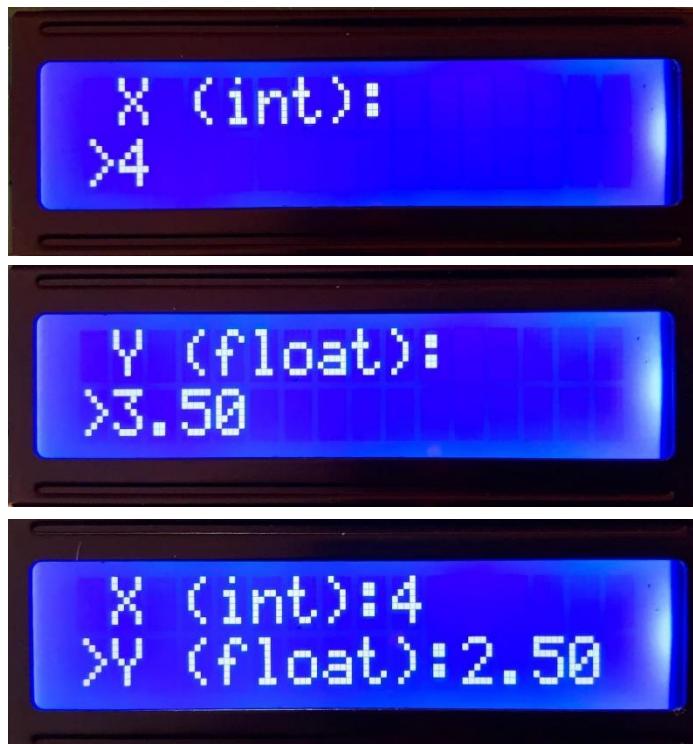
```
menu_pack->initializeValue(element, new int(1), new int(0), new int(10),
new int(1));
```

Na identičan način radimo i za element vrste `ME_FLOAT`.

U drugom primjeru kao što sam već napomenula vrijednosti su napisane u istom retku sa njihovim imenima što postižemo tako da tijekom izrade elementa umjesto praznog `Stringa` pošaljemo ime elementa koje je u prvom primjeru bilo zaseban element iznad vrijednosti. Dakle, element radimo ovako:

```
element = menu_pack->makeMenuItem("X (int):", ME_INT);
```

Kako na zaslonu izgleda prvi, a kako drugi primjer mijenjanja vrijednosti u izborniku, možemo vidjeti na slici Slika 5.3.



Slika 5.3. Mijenjanje cjelobrojne vrijednosti u prvom primjeru (prva slika); mijenjanje decimalne vrijednosti u prvom primjeru (druga slika); mijenjanje cjelobrojne i decimalne vrijednosti u drugom primjeru (treća slika)

5.4. Primjer promjene osvjetljenja LCD zaslona

Ovim primjerom koji se nalazi u `MenuPack/examples/brightnesses_test/brightnesses_test.ino` želim pokazati kako tijekom promjene cjelobrojne ili decimalne vrijednosti možemo nešto izvoditi. Ovdje mijenjamo cjelobrojnu vrijednost koja nam zapravo označava vrijednost svjetline zaslona pa kako se mijenja, mijenja se i osvjetljenošć.

Program radimo na sličan način kao i program u 5.3. Isto imamo dva podizbornika gdje prvi služi za mijenjanje vrijednosti, a drugi za prikaz trenutne vrijednosti. U prvoj imamo element vrste `ME_INT` kojeg mijenjamo. Ovaj element inicijaliziramo ovako:

```
menu_pack->initializeValue(element, new int(128), new int(0), new  
int(255), new int(32));
```

Povećanje je 32 kako bi promjene osvjetljenja bile što uočljivije.

Bitna stavka u ovom primjeru je da elementu vrste `ME_INT` moramo postaviti `actionFunctionWhenChangeValue` pošto se ta funkcija poziva prilikom mijenjanja vrijednosti. Funkcija se sastoji od samo jednog retka koda, a on izgleda ovako:

```
analogWrite(pin_BL, *(int*) (menu_pack->getNewValue(menu_pack  
->getMenuItem(change_brightness, 1))));
```

Upisujemo trenutnu vrijednost cjelobrojnog elementa na pin 10 koji je zadužen za pozadinsko osvjetljenje. Kako izgleda LCD zaslon ako mijenjamo cjelobrojnu vrijednost i u isto vrijeme se mijenja osvjetljenje možemo vidjeti na slici Slika 5.4.



Slika 5.4. Kada cjelobrojnu vrijednost promijenimo na 224 zaslon će biti iznimno osvijetljen (gornja slika), a kada vrijednost promijenimo na 0 zaslon neće biti osvijetljen (donja slika)

6. PROBLEMI I MOGUĆA POBOLJŠANJA

Ranije je već spomenuto kako biblioteka ima neka ograničenja u primjeni, jedno od njih je što se može koristiti samo na tekstualnim LCD zaslonima, točnije onim zaslonima koji imaju određeni broj redaka i stupaca. Nadalje, za rad s bibliotekom potrebno je imati minimalno dva gumba. Sljedeći problem je memorija. Zbog toga što Arduino nema puno memorije, ona se brzo može potrošiti. Ponovimo, Arduino ima *flash* memoriju od 32KB u koju pohranjuje program, 2KB statičke radne memorije za pohranjivanje podataka te 1KB trajne električno izbrisive programibilne ispisne memorije. Dakle, očito je kako je radna memorija izrazito mala što dovodi do zaključka da izbornik koji radimo mora biti ograničen. Najbolje možemo vidjeti koliko memorije zauzima tako što pogledamo napravljene primjere. Od 6 napravljenih primjera najviše memorije troši primjer u datoteci `MenuPack/examples/change_value_test/change_value_test.ino` koji zauzima 34% *flash* memorije, točnije 11274B i 5% radne memorije, točnije 117B. Iako ovaj primjer zauzima najviše memorije, to i dalje nije toliko puno. Primjer sadrži 3 izbornika i sveukupno 10 elemenata. Memorija ne smije zauzimati više od 90% jer u tom slučaju može doći do neželjenog ponašanja programa koji ne mora raditi ispravno. Iako Arduino nema puno memorije, u radu s ovom bibliotekom na tekstualnim LCD zaslonima ne bi trebalo biti velikih problema jer obično izbornici koji se koriste na takvim LCD zaslonima ni ne trebaju biti toliko veliki da bi se sva memorija zauzela. Međutim, ukoliko se zbog velike količine Stringova statička radna memorija ipak potroši, moguće je Stringove prebaciti u *flash* memoriju koristeći ovakvu deklaraciju:

```
const dataType variableName[] PROGMEM = {};
```

Biblioteka se može nadograditi tako da podržava rad i s grafičkim LCD zaslonima, kao i s okretnim gumbom (engl. *rotary push button*) čime bi se povećala njezina općenitost. Osim toga, moguće je dodati i razne druge nove funkcionalnosti čime bi se poboljšao izgled izbornika. Primjerice, traku za pomicanje (engl. *scrollbar*) sa desne strane izbornika, mogućnost automatskog pomicanja teksta (engl. *scrolling*) u lijevo ukoliko je tekst duži od jednog retka, uvođenje trake napretka (engl. *progress bar*) tijekom mijenjanja cjelobrojnih ili decimalnih vrijednosti, mogućnost stavljanja nekih posebnih znakova, kao i mnogih drugih funkcionalnosti.

7. ZAKLJUČAK

U ovom radu prikazano je ostvarenje općenite biblioteke za izradu jednostavnog izbornika na tekstualnim LCD zaslonima. Navigacija kroz izbornik ostvarena je pomoću nekoliko gumbi. Izbornik podržava pregled, promjenu i odabir vrijednosti te mogućnost kasnijeg korištenja tih vrijednosti. Biblioteka je napravljena na velikoj razini općenitosti s obzirom da je podržan rad i analognih i digitalnih gumbi, kao i tekstualnih LCD zaslona različitih veličina, točnije različitih po broju redaka i stupaca. Međutim, općenitost je i dalje moguće povećati podržavanjem rada i na grafičkim LCD zaslonima.

Moguće su daljnje nadogradnje biblioteke u smislu poboljšanja samog dizajna izbornika, primjerice dodavanje pomicne trake ili trake napretka.

LCD zaslon spaja se na mikroračunalo, a za tu primjenu korišten je Arduino Uno. Demonstrirano je spajanje LCD zaslona s Arduinom te je objašnjen rad s Arduino sklopolovljem, ali i s programskom podrškom za Arduino pločicu. Arduino se pokazao kao dobro mikroračunalo za ovu primjenu zbog kompatibilnosti u spajanju s LCD zaslonom, kao i zbog jednostavnog rada s programskom podrškom za Arduino te jednostavne provjere rada biblioteke.

LITERATURA

1. *LCD: Interfacing with PIC Microcontrollers (Part 3) – Creating Custom Character,*
<https://robu.in/lcd-interfacing-with-pic-microcontrollers-part-3-creating-custom-character/>, 16. 03. 2016.
2. *LCD and KeyPad Module (FullyAssembled PCB),*
http://www.okaauto.com/LCD_KeyPad_module-VMA203.html, 25. 11. 2018.
3. Velleman, *LCD and KeyPad shield for Arduino – User manual,*
https://www.velleman.eu/downloads/29/vma203_a4v03.pdf, 16. 01. 2019.
4. Arduino, *Getting Started with Arduino products,*
<https://www.arduino.cc/en/Guide/HomePage>, 12. 07. 2019.
5. Arduino, *Getting Started with Arduino Uno,*
<https://www.arduino.cc/en/Guide/ArduinoUno>, 16. 10. 2019.
6. Circuito team, *Everything you need to know about Arduino code,*
<https://www.circuito.io/blog/arduino-code/>, 11. 03. 2018.
7. Arduino, *LiquidCrystal Library,*
<https://www.arduino.cc/en/Reference/LiquidCrystal>, 24. 12. 2019.
8. Arduino, *Arduino Uno REV3 SMD,*
<https://store.arduino.cc/arduino-uno-rev3-smd>
9. Simon Monk, *Programming Arduino, Getting Started with Sketches,*
Mikro knjiga, drugo izdanje, 2017.
10. Xenarc Technologies, *LCD Technology,*
<https://www.xenarc.com/lcd-technology.html>
11. Sora Thompson, *Understanding Different Display Panel Types,*
<https://www.tomshardware.com/reviews/lcd-led-led-oled-panel-difference,5394.html>,
26. 08. 2018
12. Adafruit, *RGB backlight positive LCD 20x4 + extras,*
<https://www.adafruit.com/product/499>
13. Adafruit, *Standard LCD 16x2 + extras,*
<https://www.adafruit.com/product/181>
14. Arduino Greenhouse Control – Humidity and Temperature,
<https://www.instructables.com/id/Arduino-Greenhouse-Control-Humidity-and-Temperatur/>
15. Chipoteka, *Velleman,*
<https://www.chipoteka.hr/brend/velleman>
16. Leksikografski zavod Miroslav Krleža, *Potenciometar,*
<https://www.enciklopedija.hr/natuknica.aspx?id=49738>
17. Ecomputer notes, *What is EEPROM?,*
<http://ecomputernotes.com/fundamental/input-output-and-memory/eeprom>
18. Arduino, *Writing a Library for Arduino,*
<https://www.arduino.cc/en/Hacking/LibraryTutorial>

SAŽETAK

Biblioteka za ostvarenje izbornika za male tekstualne LCD zaslone

U radu su proučene mogućnosti tekstualnih LCD zaslona te je za njih napravljena biblioteka čija je svrha jednostavni izbornik. Izbornik pruža mogućnost pregleda, promjene te odabira vrijednosti. Navigacija kroz izbornik ostvarena je pomoću nekoliko gumbi. Mikroračunalo na koje se LCD zaslon spaja je Arduino Uno. Provedeni su i neki testni primjeri rada biblioteke.

Ključne riječi: LCD zaslon, izbornik, Arduino, gumbi, biblioteka

SUMMARY

Title: A library for making menus on small textual LCDs

Summary

The aim of this thesis was to study the possibilities of textual LCD screens and create a library with the purpose of making a simple menu. The menu makes it possible to view, change and select values. Navigation through the menu is accomplished by using several buttons. The microcomputer to which the LCD screen connects is the Arduino Uno. The paper also included several tests of the library's functionality.

Keywords: LCD module, menu, Arduino, buttons, library