Mehanički 7-segmentni pokaznik

Vrabelj, Matija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

7 – SEGMENTNI MEHANIČKI POKAZIVAČ

Diplomski rad

Matija Vrabelj

Osijek, 2023.

SADRŽAJ

1. UVOD

U diplomskom radu je izrađena maketa mehaničkog sedam segmentnog pokazivača. Za izradu makete korišten je mikroupravljački sustav, jedan ili više servo motora, PCB pločica te ostali popratni materijal. Svrha makete je simulirati kretanje segmenata naprijed-natrag pomoću servo motora. Segmenti su postavljeni na površinu zaslona, a iza svakog segmenta je postavljen servo motor koji se aktivira prema potrebi. Sedam segmentni mehanički pokazivač je tip zaslona koji koristi mehaničke elemente za prikaz numeričkih informacija. Sastoji se od sedam pojedinačnih segmenata raspoređenih u specifičnom uzorku kako bi se formirali različiti brojevi i znakovi. Ovi segmenti, koji su izrađeni pomoću 3D printera u ovom slučaju, obično su napravljeni od plastike ili stakla te se mogu osvijetliti izvorom svjetlosti kako bi bili čitljivi. Iako mehanički zasloni nisu danas toliko zastupljeni zbog napretka elektroničkih zaslona, imali su značajnu ulogu u povijesti računalstva i još uvijek se primjenjuju u određenim specijaliziranim područjima. Prvi mehanički zasloni datiraju iz 1870-ih i koristili su se u telegrafskim strojevima za prikazivanje poruka. Ti uređaji su koristili male metalne igle koje su se elektromehanički pomicale kako bi stvarale slova i brojeve. U današnjem svakodnevnom radu, mehanički pokazivači koriste servo motore kako bi se segmenti rotirali i stvarali različite znakove. Ovi segmenti su raspoređeni u specifičnom uzorku te se pomiču pomoću servo motora kako bi se oblikovali željeni brojevi i znakovi. Ovaj mehanizam omogućuje da svaki segment bude kontroliran neovisno, a kada je segment aktiviran, servo motor ga pomakne na određeno mjesto kako bi se formirao traženi znak. Segmenti se mogu izrađivati od različitih materijala, uključujući plastiku, staklo ili metal te su obično oblikovani kao pravokutne šipke s određenim izostavljenim uglovima kako bi se postigao željeni uzorak. Montirani su na okvir ili bazu koja im omogućuje slobodno kretanje i okretanje prema potrebi [1]. U ovom radu, segmenti su pričvršćeni na servo motore, dok je sami servo motor postavljeni na površinu zaslona. Tijekom rada u prvom poglavlju objašnjeno je što je zapravo sedam segmentni mehanički pokazivač, koje on prednosti donosi u svakodnevnom životu, koje su tehnologije upotrjebljene prilikom izrade rada, odnosno koje se komponente koriste te naravno u kojoj grani gospodarstva se može koristiti. Tijekom rada objašnjen je postupak dizajniranja, testiranja i izrade rada te je objašnjen kod korišten u radu.

1.1. Zadatak diplomskog rada

U ovom diplomskom radu potrebno je napraviti maketu mehaničkog sedam segmentnog pokazivača. Potrebno je koristiti mikroupravljački sustav, jedan ili više servo motora te modul za precizno mjerenje vremena. Prilikom izrade diplomskog rada moguće je bilo naići na razne izazove i probleme.

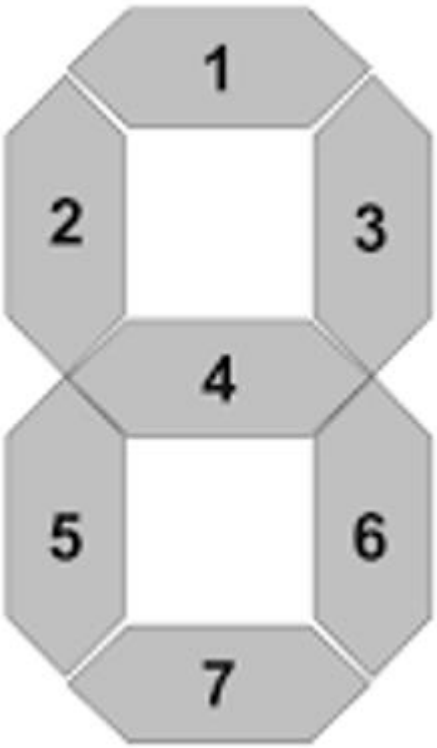
1.2. Izazovi diplomskog rada

Jedan od najvećih problema, odnosno izazova koji se mogao pojaviti jest osigurati dovoljno napajanje kako bi Arduino Mega mogao pokrenuti svih 14 servo motora. Taj problem je riješen pomoću vanjskog napajanja i PCB pločica. Kalibracija motora predstavljala je problem jer svaki servo motor nije pokazivao isti stupanj, stoga je za svaki servo motor postavljena početna i krajnja vrijednost unutar koda. Kod je dostupan u Prilogu. Kod postavljanja segmenata na ruku servo motora također je trebalo prilagoditi kako će se pričvrstiti u tome trenutku jer SG90 se može okretati samo za 180 stupnjeva i ako je krivo pričvršćen neće se moći pravilno okretati, odnosno neće se htjeti u pravu stranu okretat.

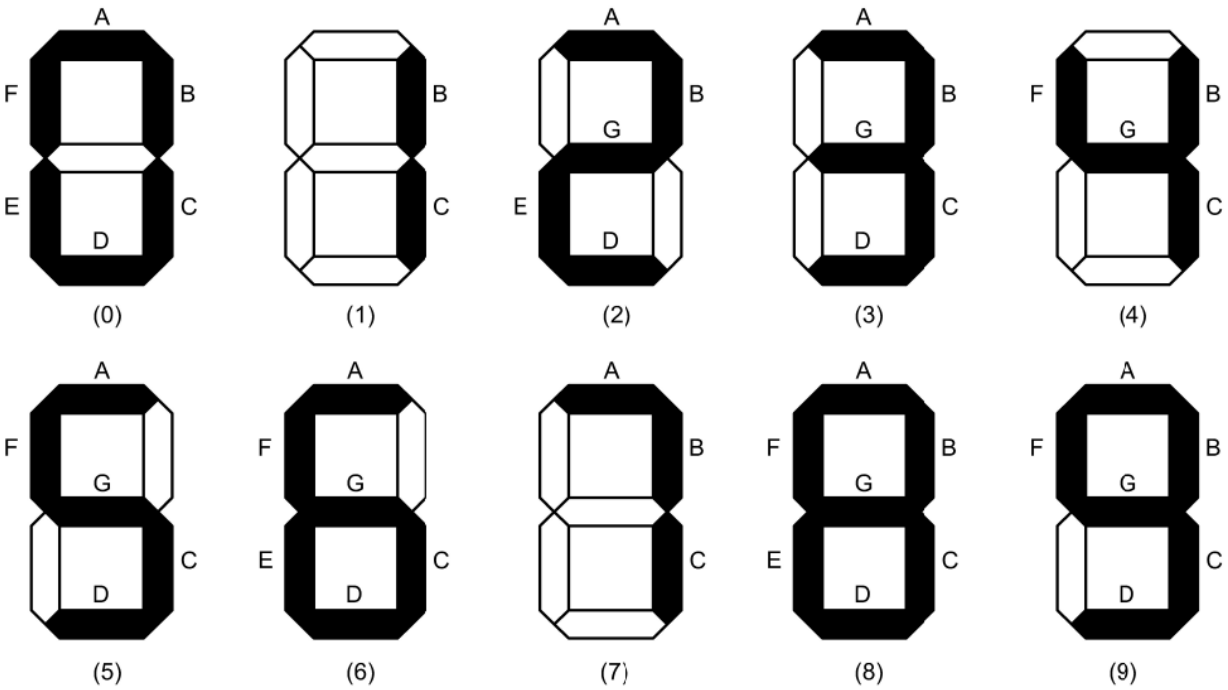
2. PREGLED PODRUČJA TEHNIKA

2.1. Sedam segmentni pokazivač

Zaslon koji se sastoji od sedam segmenata raspoređenih u sedam pozicija naziva se sedam segmentni pokazivač. Na slici 1. prikazano je sedam segmenata raspoređenih u pravokutni način i označeni su oznakom od 1 do 7, vidljivo na slici 2.1. Svaka pozicija naziva se segment jer čini dio znamenke koja se prikazuje. Različitim kombinacijama segmenata mogu se prikazati znamenke od 0 do 9. Za primjer uzet je broj nula, lopatice 1, 2, 3, 5, 6 i 7 su aktivne i okrenute prema naprijed, servo motor u ovoj orijentaciji pokazuje prema gore, odnosno sredinu, pokazane prema korisniku, dok lopatica 4 nije. U prikazivanju brojeva određena lopatica je okrenuta je prema naprijed, ovisno o znaku koji se prikazuje. Različite znamenke koje se mogu prikazivati od 0 do 9 prikazane su slikom 2.2. [2][3].



Slika 2.1. Raspored segmenata po pokazivaču [2]



Slika 2.2. Moguće kombinacije brojeva [2]

2.1.1. Prednosti sedam segmentnog pokazivača

Sedam segmentni mehanički pokazivač je uređaj za prikaz koji se sastoji od sedam pojedinačnih segmenata raspoređenih u obliku znamenke "8". Svaki segment može se pojedinačno kontrolirati za prikaz različitih brojeva ili znakova. Prednost korištenja sedam segmentnog mehaničkog pokazivača je njegova jednostavnost i pouzdanost. To je isplativo rješenje za prikaz numeričkih informacija u različitim aplikacijama. Jedna od prednosti sedam segmentnog mehaničkog pokazivača je njegova jednostavnost korištenja. Sedam segmenata može se kontrolirati jednostavnim električnim signalima, što ga čini kompatibilnim sa širokim rasponom elektroničkih sustava. Ova jednostavnost čini ga popularnim izborom za aplikacije u kojima je potreban osnovni numerički prikaz. U ovom slučaju, odnosno u ovome radu prikazana je trenutna temperatura prostorije te mogućnost odbrojavanja od 99 do 0. Još jedna od mnogih prednosti sedam segmentnog mehaničkog pokazivača je njegova trajnost. Za razliku od drugih tehnologija prikaza, kao što su LCD ili LED zasloni, mehanički indikator se ne oslanja na osjetljive elektroničke komponente. To je robustan uređaj koji može izdržati teška okruženja i mehanička naprezanja. To ga čini prikladnim za primjene u kojima su pouzdanost i dugovječnost važni čimbenici[4][5]. Nadalje, sedam segmentni mehanički pokazivač nudi dobru vidljivost. Segmenti su obično izrađeni od svijetlih materijala, u ovome slučaju segment je isprintan pomoću 3D printera i bijeloga 3D filamenta, ali po standardu segmenti su izrađeni od elemenata kao što su LED diode

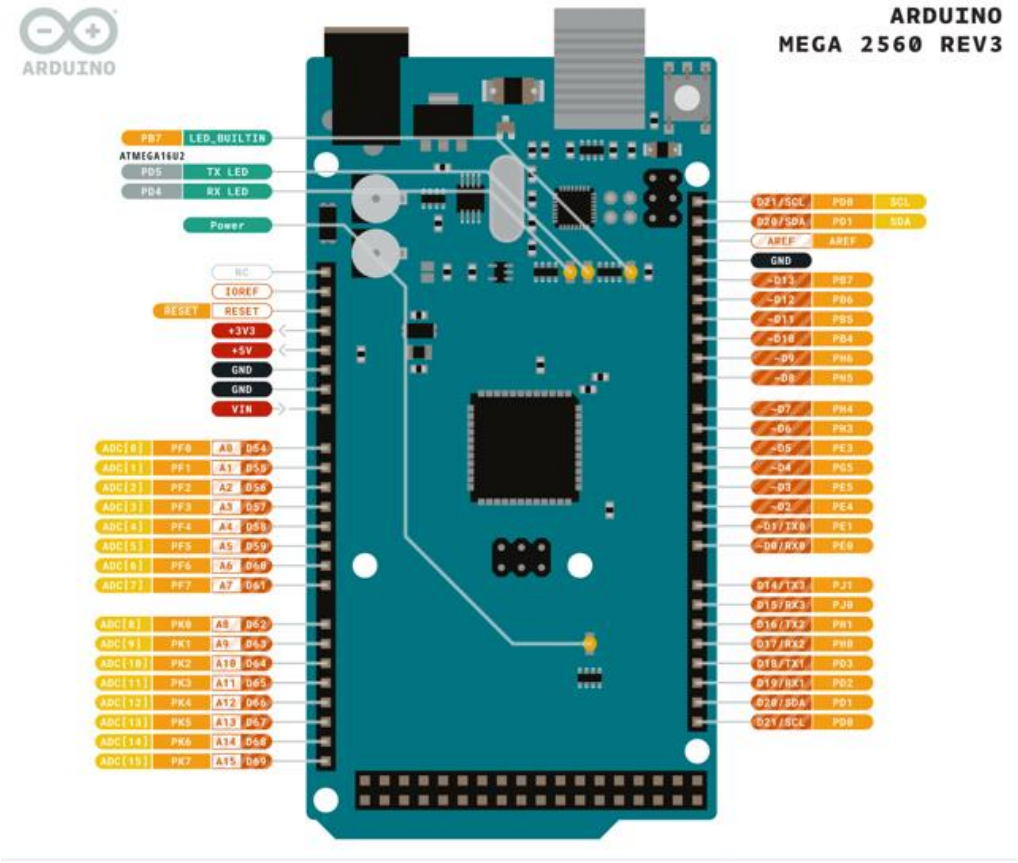
ili žarulje sa žarnom niti, koje pružaju jasne i lako čitljive znakove. Kontrast između segmenata i pozadine osigurava da su prikazane informacije vidljive, čak i u svijetlim ili prigušenim uvjetima osvjetljenja. Osim toga, sedam segmentni mehanički pokazivač je energetski učinkovit. U radu je korišteno naponski pretvarač koji na izlazu daje 5V i 6A, a to je 30W jer Arduino Mega nema dovoljnu snagu kako bi pokretao svih 14 segmenata. Ova energetska učinkovitost postiže se korištenjem komponenti male snage i optimizacijom dizajna zaslona. Jedan servo motor zahtjeva 550 miliampera kako bi mogao raditi. Sve u svemu, sedam segmentni mehanički pokazivač nudi jednostavno, pouzdano, izdržljivo i energetski učinkovito rješenje za prikaz numeričkih informacija. Njegova jednostavnost korištenja, vidljivost i kompatibilnost s različitim elektroničkim sustavima čine ga popularnim izborom u širokom rasponu primjena, uključujući mehaničke satove, industrijske upravljačke ploče i automobilske nadzorne ploče te mnoge druge [6].

2.2. Korišteni dijelovi u izradi rada

2.2.1. Mikroupravljač Arduino Mega

Arduino Mega je razvojna ploča bazirana na ATmega2560 mikroupravljaču i jedna je od popularnijih ploča. Ovaj mikroupravljač je posebno poznat po svojoj količini digitalnih i analognih I/O pinova, što ga čini idealnim za projekte koji zahtijevaju veliki broj senzora, aktuatora i drugih periferija. Arduino Mega koristi AVR mikroupravljač ATmega2560 koji ima 8-bitnu arhitekturu, brzinu od 16 MHz i 256 KB memorije za pohranu programa. Također ima 8 KB SRAM memorije i 4 KB EEPROM memorije za pohranu podataka. Ova kombinacija memorije omogućava izvođenje složenih programa i pohranu podataka na mikroupravljaču. Arduino Mega ima ukupno 54 digitalna I/O pina, od kojih je 15 moguće koristiti kao PWM (*Pulse Width Modulation*) izlaze. Ovi pinovi omogućavaju povezivanje sa senzorima, LED diodama, motorima i drugim elektroničkim komponentama. Osim digitalnih pinova, Arduino Mega također ima 16 analognih ulaza, što omogućava čitanje analognih senzora i napona. Mikroupravljač podržava serijsku komunikaciju putem UART, SPI i I2C protokola. Ovo omogućava komunikaciju s drugim uređajima poput računala, senzora, LCD ekrana i drugih mikroupravljača. Arduino Mega ima 4 UART porta, 1 SPI port i 1 I2C port za takvu komunikaciju. Osim toga, Arduino Mega ima nekoliko dodatnih mogućnosti kao što su 6 PWM kanala za kontrolu brzine motora ili svjetline LED-a, 16 MHz kristalni oscilator za precizno mjerenje vremena, USB konektor za programiranje i napajanje, DC priključak za napajanje i ICSP (*In-Circuit Serial Programming*) priključak za

programiranje mikroupravljača izvan Arduino okruženja. Svi dostupni pinovi vidljivi su na slici 2.3. Arduino Mega je popularan među hobistima, studentima i profesionalcima zbog svoje količine I/O pinova i mogućnosti proširenja. Ovaj mikroupravljač se često koristi u projektima kao što su robotika, automatizacija, internet objekata (IoT), umjetna inteligencija, senzorski sustavi, interaktivne instalacije i mnogi drugi. Zahvaljujući zajednici korisnika Arduino platforme, postoje mnogi dostupni resursi za učenje i podršku za Arduino Mega. Postoje biblioteke i primjeri koda koji olakšavaju programiranje i razvoj projekata. Također postoje brojne nadogradnje koje omogućavaju dodavanje dodatnih funkcionalnosti poput bežične komunikacije, GPS praćenja, upravljanja motorima i mnoge druge. Ukratko, Arduino Mega je snažan mikroupravljač koji pruža velike mogućnosti za razvoj različitih elektroničkih projekata. Njegova fleksibilnost, broj I/O pinova i podrška široke zajednice korisnika čine ga idealnim izborom za mnoge aplikacije u svijetu elektronike i programiranja. Detaljniji opis mikroupravljača Arduino mega može se pronaći na službenoj dokumentaciji [7][8][9].



Slika 2.3. Raspored pinova na Arduino Mega [7]

2.2.2. Servo motori

Za svaki segment korišten je po jedan servo motor koji ima dva stanja 0 ili 1, odnosno zaokreće se za 90 stupnjeva ovisno o aktivnom ili neaktivnom stanju. Servo motor je vrsta motora koja se može okretati s velikom preciznošću. Obično se ovaj tip motora sastoji od upravljačkog kruga koji daje povratne informacije o trenutnom položaju osovine motora, ova povratna informacija omogućuje servo motorima da se okreću s velikom preciznošću. Ako se želi rotirati objekt pod nekim određenim kutom, tada se koristi servo motor. Sastoji se samo od jednostavnog motora koji proizlazi kroz servo mehanizam. Ako se motor napaja istosmjernim napajanjem (DC), onda se naziva istosmjerni servo motor, odnosno DC servo motor, ako se radi o motoru s izmjeničnim napajanjem, tada se radi o AC servo motoru. Sastoji se tri dijela kontroliranog uređaja, izlaznog senzora i povratne informacije. To je sustav zatvorene petlje u kojem koristi pozitivan sustav povratnih informacija za kontrolu kretanja i konačnog položaja osovine. Ovdje se uređajem upravlja povratnim signalom generiranim usporedbom izlaznog signala i referentnog ulaznog signala. Servo motori imaju tri žice koje izlaze iz njih od kojih se dva koriste za spajanje na napajanje i uzemljenje, a dok se treći koristi za slanje signala mikroupravljaču, odnosno PWM. Servo motori u ovom radu rade na +5V [10].

Slika 2.4. Servo motor [10]

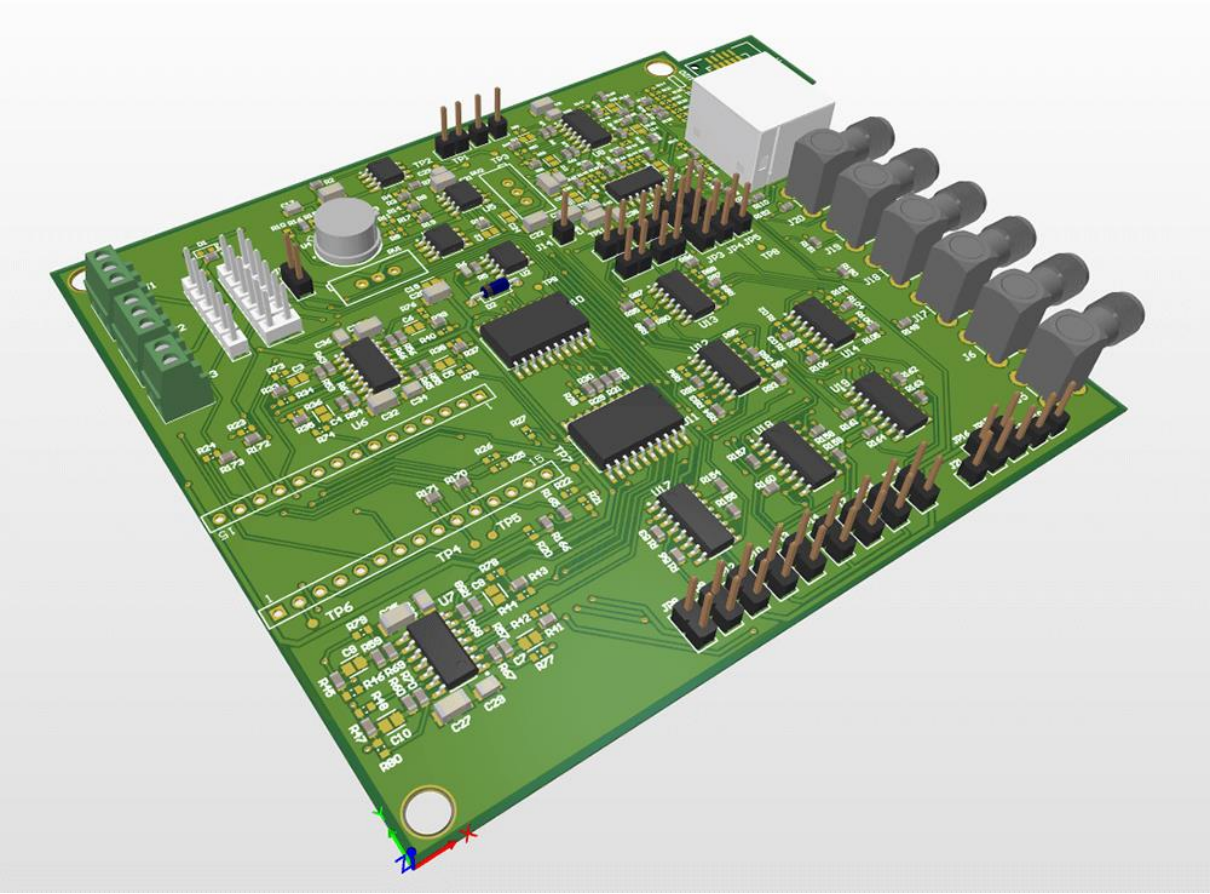
2.2.3. Arduino IDE

Arduino *Integrated Development Environment* (IDE) je besplatni softver za razvoj aplikacija koji se koristi za programiranje mikroupravljača. IDE omogućuje programerima da pišu, sastavljaju (*compile*) i prenose programe na Arduinove ploče. IDE je osmišljen kao jednostavan način za programiranje ploča za početnike, ali je prilagođen i naprednim korisnicima. Kako se Arduino zajednica razvijala, tako se i IDE razvijao kako bi se prilagodio novim potrebama i trendovima. Arduino IDE ima jednostavan i intuitivan sučelje koje omogućuje programerima da lako pišu, sastavljaju, odnosno prevode i prenose programe na Arduinove ploče. IDE podržava programiranje u C ili C++ programskom jeziku, što omogućuje programerima da stvore složene projekte, u ovom radu korišten je C programski jezik. IDE također sadrži ugrađene primjere koda koji mogu pomoći programerima da brže nauče kako se pišu programi. Tekstualni uređivač koda - koristi se za pisanje programa. Prevoditelj, odnosno *compiler* - koristi se za pretvaranje izvornog koda u strojni jezik. *Bootloader* - koristi se za prenošenje koda s računala na Arduinovu ploču. *Serial Monitor* - koristi se za prikazivanje poruka koje se šalju i primaju s Arduinove ploče. IDE također ima nekoliko naprednih značajki koje olakšavaju programiranje Arduinovih ploča, kao što su podrška za biblioteke, alati za debugiranje, odnosno za pronalaženje grešaka i mogućnost povezivanja na internet. Arduino IDE se može preuzeti s web stranice Arduino.cc, gdje je dostupan za Windows, Mac OS i Linux operacijske sustave. Nakon preuzimanja i instalacije IDE-a, korisnici mogu otvoriti tekstualni uređivač koda i početi pisati program. Kada je program napisan, korisnik može kliknuti gumb za *compiling* koji pretvara izvorni kod u strojni jezik. Ako nema grešaka u kodu, korisnik može prenijeti program na Arduinovu ploču koristeći *bootloader*. Arduino IDE također podržava veliki broj biblioteka koje programerima omogućuju da lako dodaju funkcionalnosti u svoje projekte. Biblioteke su napisane u C ili C++ programskom jeziku i mogu se preuzeti s Arduino biblioteke ili s drugih web stranica. Arduino IDE se često koristi za izradu prototipova internet objekata (IoT) aplikacija. Korisnici mogu povezati svoje Arduinove ploče s senzorima i drugom elektronikom kako bi stvorili uređaj koji može mjeriti temperaturu, vlagu, pritisak, svjetlost ili druge vrijednosti. Te se vrijednosti mogu prikazivati pomoću servo motora, odnosno sedam segmentnog mehaničkog pokazivača. Arduino IDE omogućuje programerima stvaranje različitih vrsta projekata. S jednostavnim sučeljem i mnogim značajkama, IDE je dostupan svim korisnicima, bez obzira na razinu znanja o programiranju ili elektronici. Arduino zajednica je aktivna i nudi velik broj primjera koda, biblioteka i resursa koji pomažu korisnicima da lakše nauče kako koristiti IDE i programirati Arduinove ploče [11].

2.2.4. PCB - Printed Circut Board

Tiskane pločice (PCB) su daleko najčešća metoda sastavljanja modernih elektroničkih sklopova. Sastoje se od slojeva izolacijskog materijala i slojeva bakra koji sadrže signalne tragove, napajanje i uzemljenje. Dizajni rasporeda tiskanih pločica su jednako zahtjevan kao dizajn električnog sklopa. Većina modernih sustava sastoji se od višeslojnih ploča bilo to do osam slojeva (ili ponekad čak i više). Tradicionalno, komponente su montirane na gornji sloj u rupama koje se protežu kroz sve slojeve. Oni se nazivaju *through hole* komponente, odnosno komponente s rupama kroz slojeve. U novije vrijeme, sa svim novijim usvajanjem tehnologije komponente se mogu pronaći u gornjem i donjem sloju. Dizajni tiskane pločice su jednako važan kao i dizajn sklopa za ukupne performanse sustava. Efekti na PCB-ima koji štetno utječu na preciznu izvedbu elektroničkih krugova uključuju curenje otpora, padove napona zbog otpora, vijcima i uzemljenju, utjecaj praznog kapaciteta te dielektričku apsorpciju. Osim toga, sklonost PCB-ova da apsorbiraju atmosfersku vlagu znači da se doprinos nekih parazitskih efekata često mijenja iz dana u dan ovisno o vlažnosti zraka. Općenito, efekti PCB-a mogu se podijeliti u dvije glavne kategorije - one koje najviše utječu na statički ili istosmjerni rad sklopa i one koje najviše utječu na dinamički ili izmjenični rad sklopa, posebno pri visokim frekvencijama. Još jedno široko područje dizajna PCB-a je uzemljenje. Uzemljenje je problematično područje za sve analogne i mješovite signale, i može se reći da jednostavna primjena sklopa na PCB-u ne mijenja činjenicu da su potrebne odgovarajuće tehnike. Određeni principi kvalitetnog uzemljenja, poput korištenja uzemljenih slojeva, inherentni su u okruženju PCB-a. Taj faktor je jedna od značajnijih prednosti PCB baziranih analognih sklopova. Neki drugi aspekti uzemljenja koji se moraju upravljati uključuju kontrolu nepoželjnih uzemljenja i povratnih signala koji mogu narušiti performanse. Ti naponi su posljedica vanjskog pojačivača signala, zajedničkih struja ili jednostavno prevelikih padova IR napona u uzemljenim vodičima. Odgovarajuće usmjeravanje i dimenzioniranje vodiča, kao i tehnike obrade diferencijalnih signala i izolacije uzemljenja, omogućuju kontrolu takvih parazitskih napona [12]. Način na koji PCB dizajn radi je postupak koji koristi softver za izgled i računalno potpomognut dizajn (EasyEDA) koji kombinira usmjeravanje i postavljanje električne povezanosti na PCB-u. Kod postavljanja komponenata shematski dijagram mora biti pretvoren u fizički PCB tlocrt. To je zahtjevan proces jer sve komponente moraju biti smještene u malen i uzak prostor. Usmjeravanje je još jedan dio na koji se mora paziti prilikom PCB dizajniranja. To je dio gdje se žice spajaju sa komponenata i stvaraju kompletan i jedinstven PCB izgled. PCB također mora biti i testiran zbog provjere valjanosti te spriječen u ranim fazama [13].

U radu za izradu PCB pločica korišten je program EasyEDA kojeg je moguće pronaći u online verziji.



Slika 2.5. Primjer dizajn PCB-a [13]

2.3. Područje korištenja sedam segmentnog pokazivača

Sedam segmentni mehanički pokazivači su bili popularni u prošlosti, prije pojave modernijih tehnologija, kao što su LED i LCD ekrani. Stoga, postoji mnoga područja koji se bave ovim pokazivačima, uključujući područja elektronike, računarstva i inženjerstva. Postoje radovi koji se bave konstrukcijom i funkcijom sedam segmentnih mehaničkih pokazivača, kao i njihovom primjenom u različitim uređajima, poput digitalnih satova, brojila, mjerača i drugih. Također postoje radovi koji se bave optimizacijom upravljanja sedam segmentnim mehaničkim pokazivačima kako bi se smanjila potrošnja energije i poboljšala preciznost prikaza. Također, postoje i radovi koji se bave usporedbom sedam segmentnih mehaničkih pokazivača s drugim vrstama prikaza, kao što su LED i LCD ekrani, u smislu pouzdanosti, trajnosti, energetske

učinkovitosti i drugih performansi. Postoji mnogo radova napisanih o sedam segmentnim mehaničkim pokazivačima, a njihova primjena i značaj u elektronici i drugim područjima je i dalje prisutna, unatoč razvoju modernijih tehnologija. Jedna od najčešćih primjena sedam segmentnih pokazivača u inženjerstvu je u digitalnim instrumentima, poput mjerača struje, napona, temperature i drugih. Ovi instrumenti koriste sedam segmentne pokazivače za prikazivanje izmjerene vrijednosti na jednostavan i lako čitljiv način. Osim toga, sedam segmentni pokazivači se također koriste u industrijskim kontrolama, gdje se koriste za prikazivanje parametara i postavki različitih procesa. Kada se koriste u inženjerstvu, sedam segmentni pokazivači se obično kontroliraju pomoću mikroupravljača ili drugih programabilnih uređaja. Ovi uređaji se koriste za generiranje signala koji upravljaju svakim od sedam segmenata, što omogućuje prikazivanje različitih simbola. Mikroupravljači se često koriste u aplikacijama gdje se zahtijeva precizna kontrola prikaza, kao što su digitalni instrumenti i kontrolne ploče. U računarstvu, sedam segmentni pokazivači se često koriste za prikazivanje brojeva i drugih znakova u različitim uređajima, poput računalnih satova, kalkulatora, digitalnih termometara i drugih. Također se koriste u različitim računalnim igrama, gdje se koriste za prikazivanje rezultata, broja života i drugih informacija. Jedna od ključnih prednosti sedam segmentnih pokazivača u odnosu na druge vrste prikaza, poput LCD i LED ekrana, je njihova jednostavnost i ekonomičnost. Sedam segmentni pokazivači su jeftiniji od drugih vrsta prikaza, a također zahtijevaju manje napajanje i nisu osjetljivi na vanjske utjecaje, poput vibracija ili elektromagnetskih polja. Međutim, sedam segmentni pokazivači imaju neke nedostatke koji mogu utjecati na njihovu upotrebu u nekim aplikacijama. Na primjer, ovi pokazivači su teško čitljivi ako se koristi više od jednog pokazivača za prikazivanje složenih simbola ili teksta. Također, zbog svoje ograničene razlučivosti, sedam segmentni pokazivači nisu pogodni za prikazivanje složenijih grafičkih elemenata. Da bi se riješili neki od ovih problema, neki proizvođači su razvili višedigitne sedam segmentne pokazivače koji mogu prikazati više od jednog znaka u isto vrijeme.

2.4. Aktualni znanstveni radovi

Autori rada [13] predstavili su razvoj digitalnog sata temeljenog na mikroupravljaču s mehaničkim prikazom pomoću sedam segmentnog prikaza. Rad se bavi razvojem digitalnog sata temeljenog na mikroupravljaču s mehaničkim prikazom pomoću sedam segmenata. Cilj rada je razviti jednostavan i precizan digitalni sat kojeg je moguće koristiti u različitim područjima kao što su industrija, transport, medicina, itd. U radu se detaljno opisuje izrada sata, odabrani mikroupravljač, hardverska i softverska arhitektura te proces izrade. Svi potrebni dijelovi za izradu sata navedeni

su u radu, od mikroupravljača do sedam segmentnog prikaza i ostalih potrebnih dijelova. Mikroupravljač ATMega16 odabran je zbog svoje sposobnosti upravljanja sedam segmentnim prikazom te zbog njegove jednostavne upotrebe. Sedam segmentni prikaz koristi se za prikazivanje sata i datuma. Sat ima mogućnost podešavanja vremena te prikaza sata u 12 ili 24 sata. Autori su također implementirali alarmnu funkciju koja se aktivira u određeno vrijeme koje se može unaprijed postaviti pomoću tipkovnice na mikroupravljaču. Također su dodali mogućnost podešavanja temperature za automatsko podešavanje vremena. U radu je opisan i proces izrade sata, od nabave potrebnih dijelova do njihove montaže. Detaljno je objašnjena softverska arhitektura sata te su navedeni koraci programiranja mikroupravljača. Rezultati testiranja sata su pokazali da je razvijeni sat precizan te da radi bez problema. Autori su zaključili da se njihova metoda izrade digitalnog sata temeljenog na mikroupravljaču s mehaničkim prikazom pomoću sedam segmentnog prikaza može primijeniti u različitim područjima. Ovaj rad je značajan jer pokazuje primjenu sedam segmentnog pokazivača u različitim područjima, u ovom slučaju u izradi digitalnog sata temeljenog na mikroupravljaču. To ukazuje na raznolikost primjena sedam segmentnog prikaza te potencijalne primjene u budućnosti u drugim područjima inženjerstva i računarstva. Softverski dio uključuje programsku petlju koja upravlja radom sedam segmentnog pokazivača. Programski kod se izvodi na mikroupravljaču i uključuje funkcije koje omogućuju prikazivanje vremena, kao i dodatne funkcije kao što su postavljanje alarma i podešavanje svjetline prikaza. S obzirom na to da se radi o jednostavnom uređaju, softverska arhitektura je prilično jednostavna. Programski kod je napisan u C jeziku i optimiziran je kako bi se postigla brza i precizna operacija pokazivača. Uz to, mikroupravljač ima dovoljno memorije i procesorske snage da se sve funkcije izvršavaju bez problema [14].

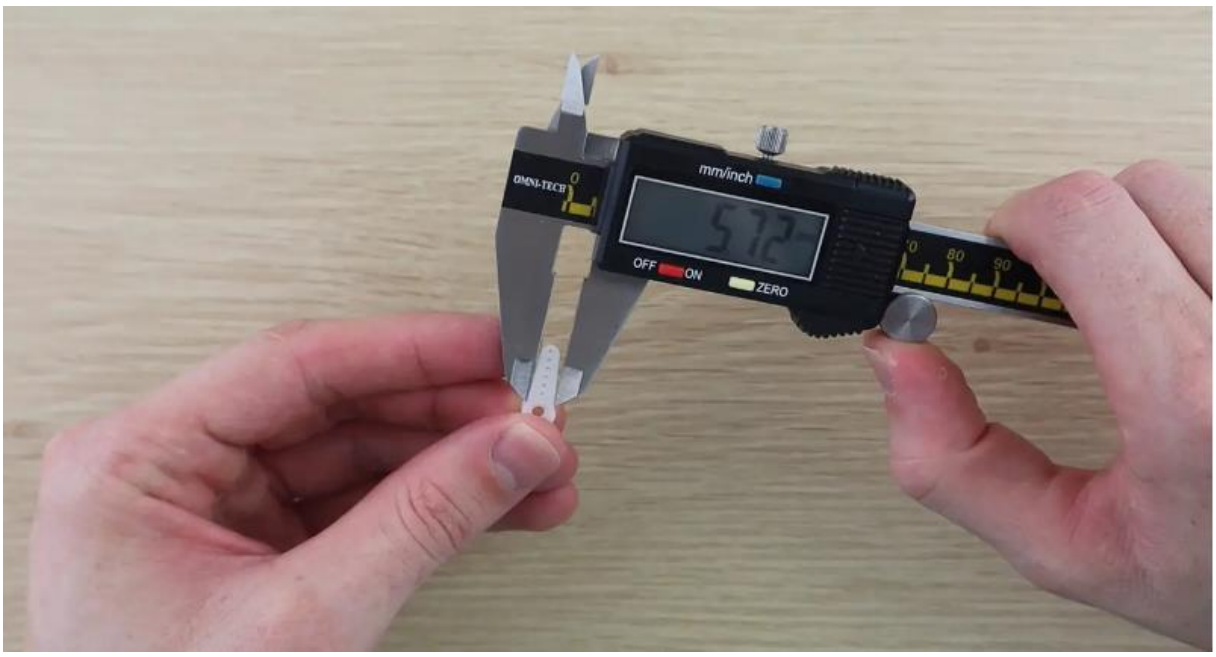
3. REZULTATI IZRADE

Uzimajući u obzir raširenost mehaničkih zaslona, kao oni koji su korišteni u zračnim lukama, željezničkim kolodvorima i mnogim drugim svakodnevnim prilikama. Servo motori primjenjuju se u raznovrsnim robotskim i bioničkim rukama, stoga je domišljena ideja korištenja servoa kako bi pokrenuo sedam segmentni mehanički pokazivač. Zbog potrebe korištenja 14 PWM pinova morao se koristiti Arduino MEGA jer podržava 15 PWM izlaza, dok ostali podržavaju manje, na primjer Uno podražava samo 6 PWM izlaza. Zbog povećeg broja PWM izlaza na Arduino MEGA lako se moglo spojiti dva pokazivača. Te je bilo potrebno duplicirati dijelove koda.

Kako bi se izradio sedam segmentni mehanički pokazivač potrebno je bilo koristiti Arduino Mega, četrnaest servo motora, odgovarajući izvor napajanja (slika 3.9.), traka s pinovima, PCB pločica, 3D printer i filament.

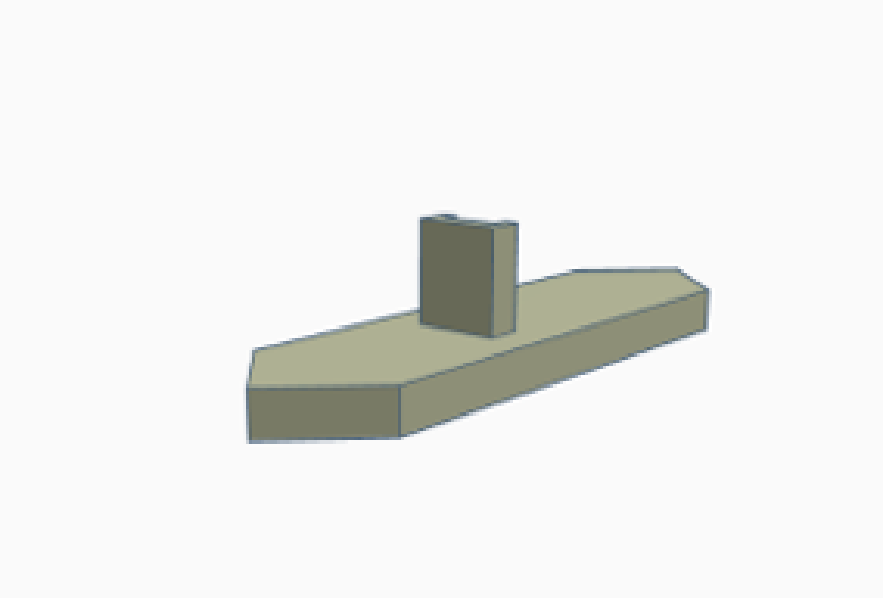
3.1. Komponente zaslona

U početnom dijelu potrebno je isprintati segmente te ih naravno zalijepiti na same servo motore. Segment je identičan za sve servo motore, stoga je potrebno isprintati njih četrnaest. Izmjerene su servo ruke kako bi se mogao dizajnirati segment koji se može izravno zalijepiti ruke servo motora kako bi se izbjegla potreba korištenja bilo kakvih drugih hardvera.

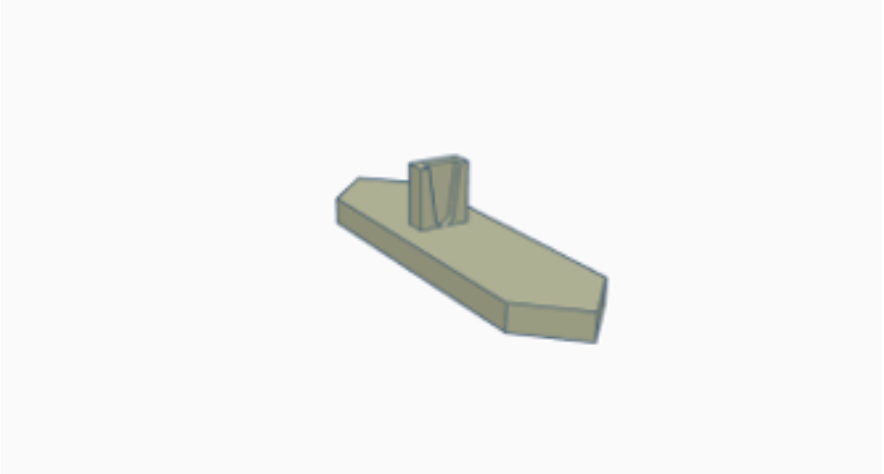


Slika 3.1. Mjerenje ruke servo motora.

Nakon dizajniranja isprintani su četrnaest segmenta koristeći bijeli PLA filement. Koristila se bijela boja zbog njezine lakoće prebrojavanja u drugu boju.



Slika 3.2. segment – vanjski pogled

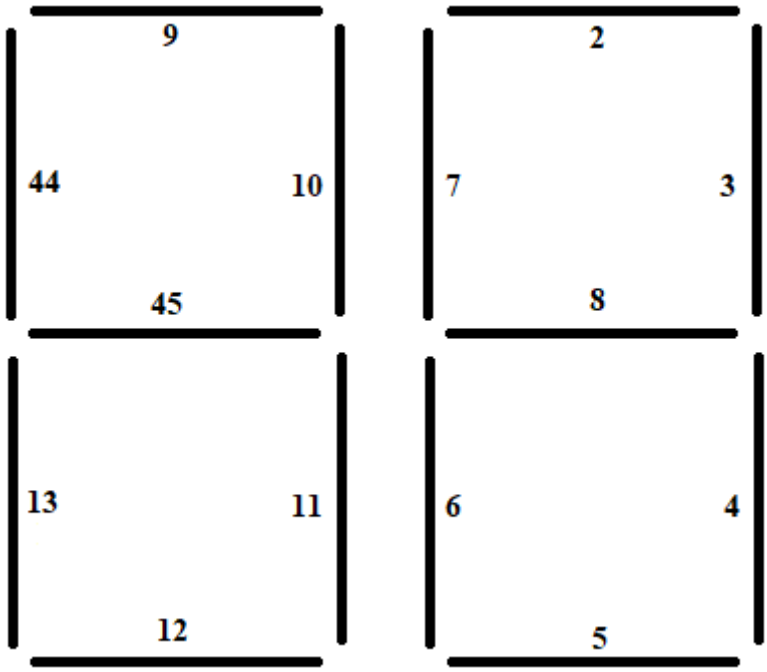


Slika 3.3. Segment – unutarnji pogled

Započeto je tako što je prvo isprintan jedan segment kako bi se provjerilo da li odgovara servo motoru i da se vidi kako funkcionira, odnosno da li je dovoljno lagan jednom kada je zalijepljen na servo motor. Segment je zalijepljen na servo ruku pomoću dvokomponentnog lijepila, odnosno *epoxy-a.*

Nakon odrađenih svih testiranja segmenti su polijepljeni na ruke servo motora. Onda je mapirano koji su segmenti povezani s kojim pinom na Arduino Mega kako bi bili što prikladnije i ljepše povezani.

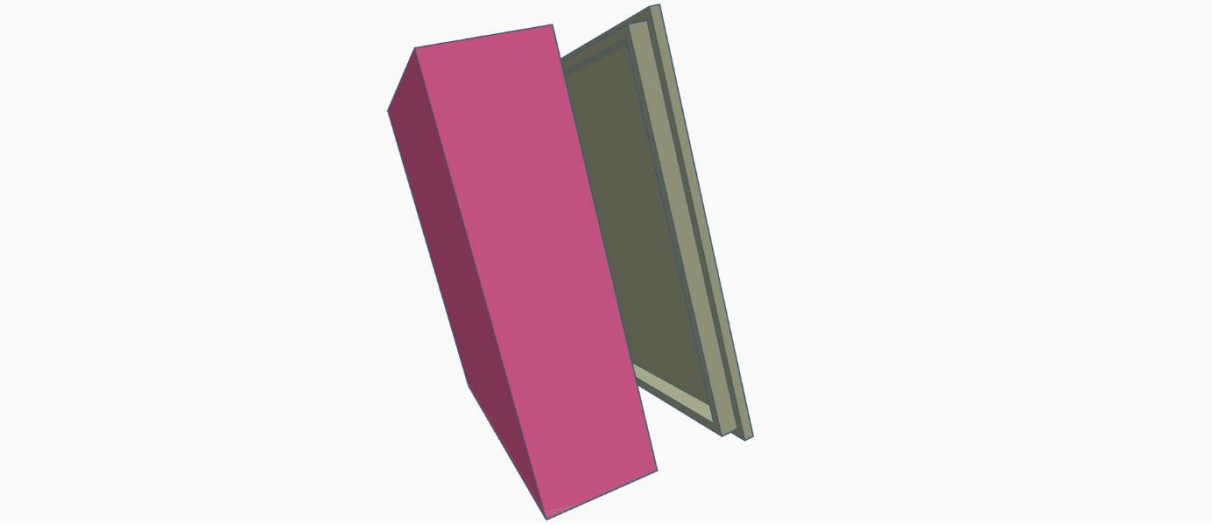
Arduino Mega podržava dvanaest PWM ulaza sekvencijalno od dva do trinaest te su dodatno korišteno pinovi četrdeset četiri i četrdesetpeta kao dodatni PWM pinovi za zadnja dva segmenta.



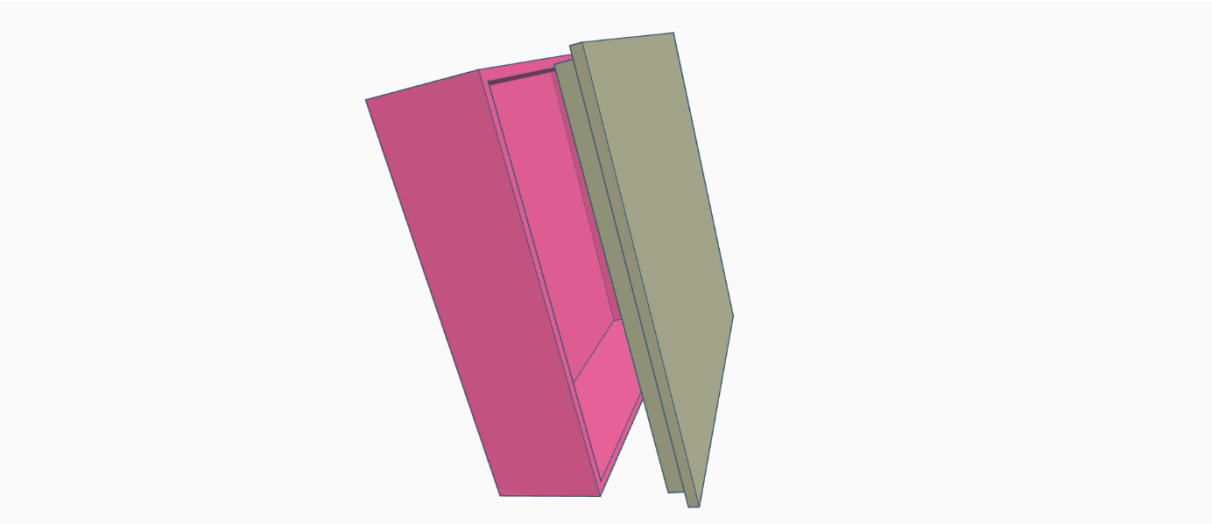
Slika 3.4. Raspored pinova

Zbog velike potražnje snage, odnosno napona korišteno je dodatno napajanje kako bi pokrenulo svih 14 servo motora, slika 3.9. Koristi se naponski pretvarač koji daje na izlazu 5V i 6A i to je ukupno snage 30W za pokretanje svih servo motora. GND i 5V pinovi iz PCB-a povezani su s odgovarajućim pinovima na Arduino Mega, dok su GND pinovi sa svih četrnaest servo motora povezani s odgovarajućim PWM pinovima na istoj pločici. Na taj način, servo motori su dobili potrebnu snagu, a Arduino Mega se koristi za upravljanje pinovima servo motora.

Nakon izrade komponenata zaslona potrebno je izraditi odgovarajuće postolje, odnosno zaslon na kojem su sve komponente. Taj zaslon, odnosno kutija izrađena je od PLA filamenta pomoću 3D printera. Primjer dizajna kutija vidljiv je na slikama 3.5. i 3.6.



Slika 3.5. Izgled dizajna kutije

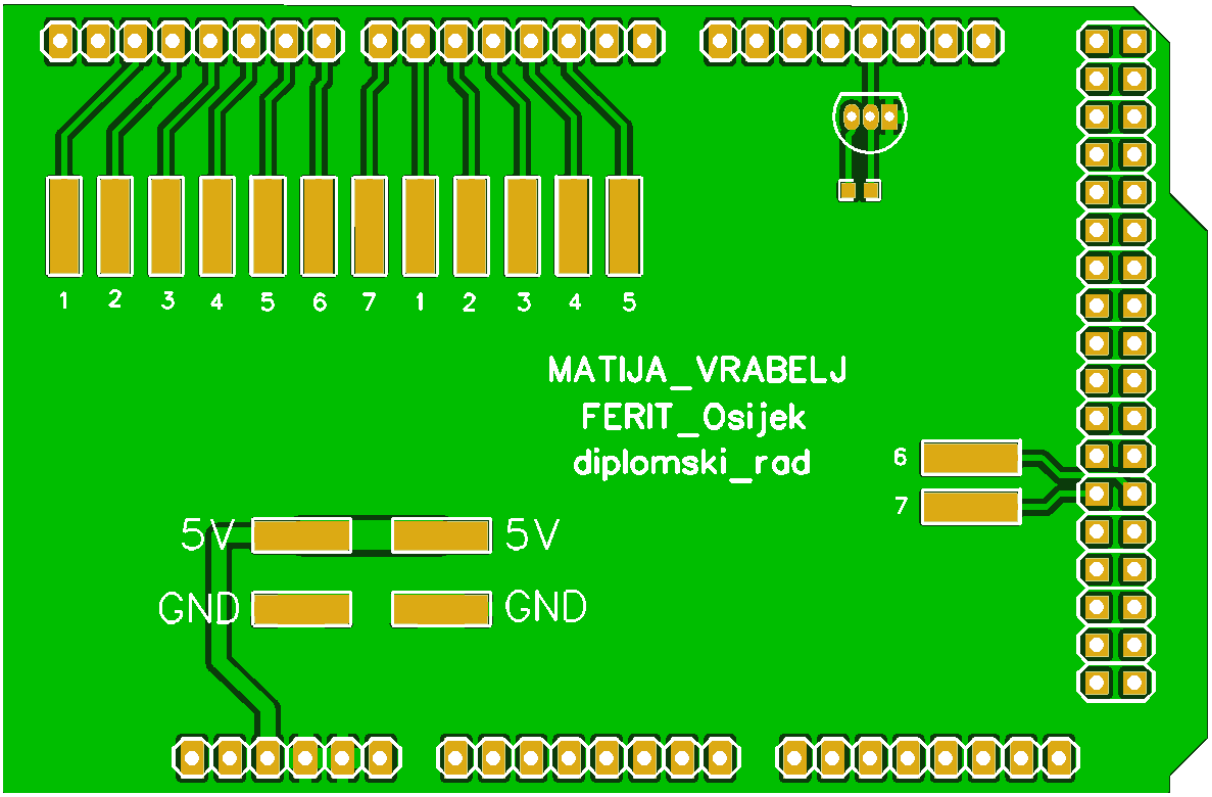


Slika 3.6. Izgled dizajna kutije – drugi kut

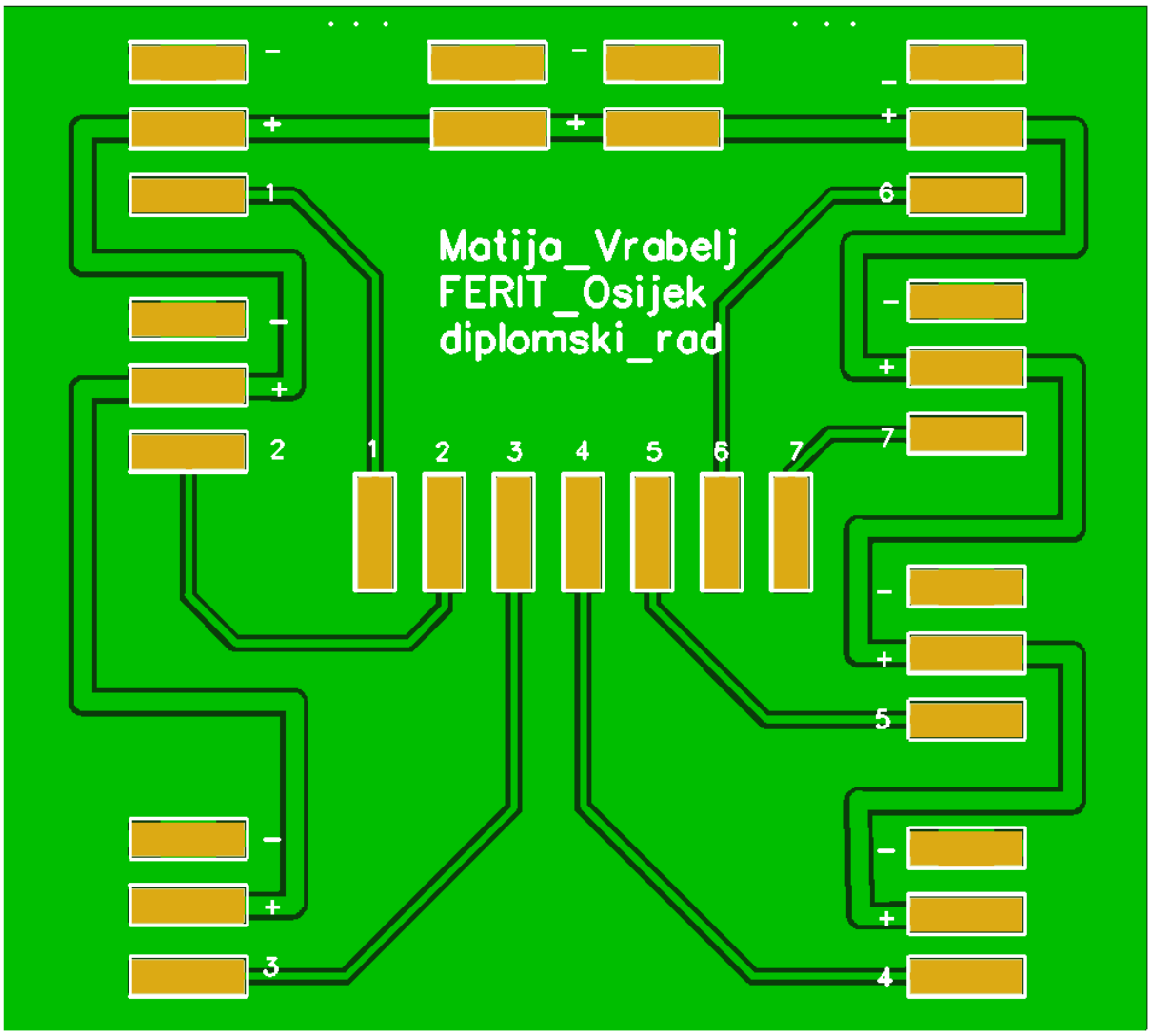
Iz priloženih slika je vidljivo kako se zaslon sastoji od dva dijela, kutije i poklopca iste kako bi činile jednu zatvorenu cjelinu i čist dizajn bez vidljivih komponenti.

3.2. Izrada PCB-a

Zbog njihove jednostavnosti i jeftine izrade korišten je PCB napravljen pomoću programa EasyEDA. Zbog povećeg korištenja struje koje su zahtijevali servo motori Mega nije u mogućnosti udovoljiti zadanim potrebama, stoga je korišten PCB koji je samostalno propuštao 5A uz dodatak vanjskog napajanja čime su se zadovoljile potrebe struje.



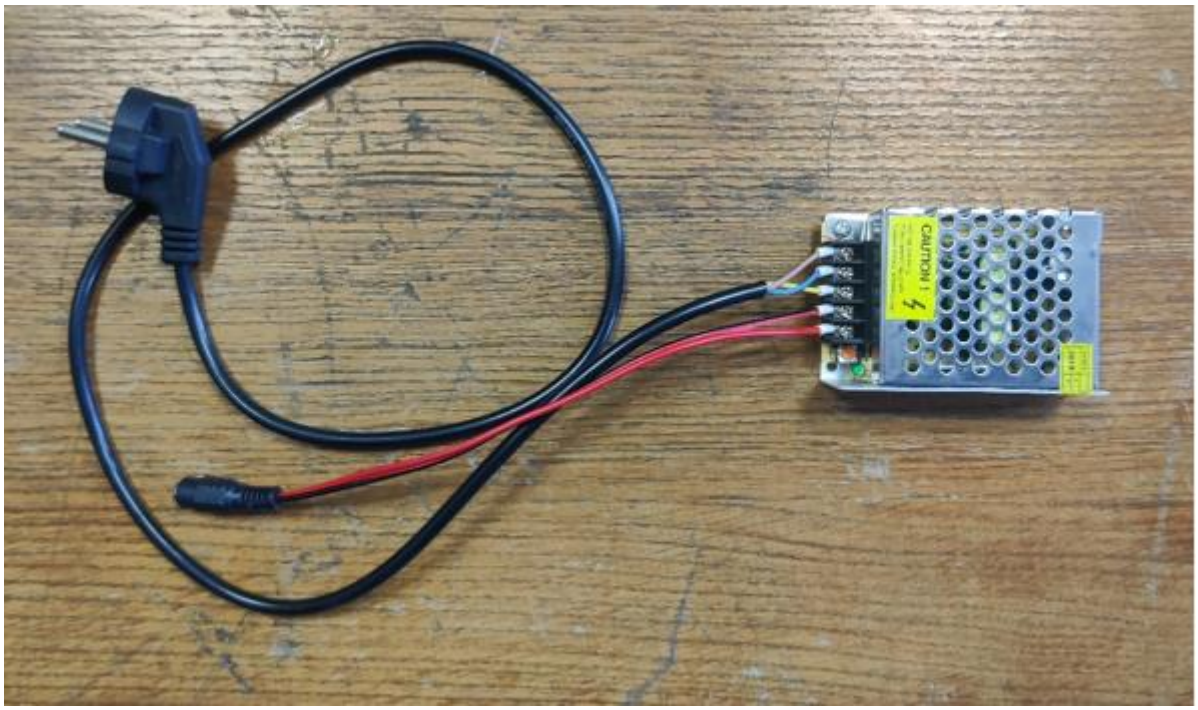
Slika 3.7. PCB – Arduino Mega



Slika 3.8. PCB – sedam segmentni upravljač

Slikom 3.7. prikazan je PCB dizajn Arduino Mega . U gornjem dijelu postavljeni su PWM pinovi, od 2 do 13, svakoga segmenta, odnosno servo SG90 motora, dodatna dva postavljena su na pinove 44 i 45 koji služe kao dodatni PWM pinovi. U donjem dijelu dovedeno je napajanje i uzemljenje. Također u gornjem dijelu vidljiv je senzor temperature DS18B20. Svi spojevi su u gornjem sloju.

Zbog lakšeg pregleda i izrade rada napravljen je PCB pločica na koju se spaja svih 7 servo motora, a dok se na drugih 7 spaja na drugu pločicu. Vidljivo na slici 3.8.



Slika 3.9. Vanjsko napajanje

3.3. Program za odbrojavanje

Prvotno je inicijaliziran niz od 14 objekata servo motora koji se nazivaju "*myServo*", a ovi brojevi odgovaraju numeričkom nizu u dijagramu brojača. Zatim inicijalizirana su tri niza brojeva. Prvi, nazvan "*segment On*", pohranjuje položaje uključenih (*on*) ili podignutih (*up*) servoa za svaki od 14 servo motora, kao svojevrsnu kalibraciju kako bi se osiguralo da su svi savršeno okomiti. Drugi, nazvan "*segment Off*", je isti, ali za položaj kada je kut servo motora 90 stupnjeva (isključen položaj). Neki od tih položaja su 0, a neki 180, ovisno o smjeru u kojem se servo motor treba pomicati. Posljednji je dvodimenzionalni niz znamenki koji pohranjuje položaje segmenata za svaku znamenku od 0 do 9, gdje je 1 uključen (vidljiv) i 0 isključen (90 stupnjeva, nevidljiv).

U kodu za postavljanje (*setup*), 14 servo motora su dodijeljena ispravnim pinovima, a zatim se petljom postavljaju svi na uključen položaj kako bi se prikazao broj 88. Zatim kod čeka 5 sekundi prije početka odbrojavanja.

```
void setup() {
  myservo[0].attach(2);
  myservo[1].attach(3);
  myservo[2].attach(4);
  myservo[3].attach(5);
  myservo[4].attach(6);
  myservo[5].attach(7);
  myservo[6].attach(8);
  myservo[7].attach(9);
  myservo[8].attach(10);
  myservo[9].attach(11);
  myservo[10].attach(12);
  myservo[11].attach(13);
  myservo[12].attach(45);
  myservo[13].attach(44);
  for (int i = 0; i <= 13; i++)
  {
    myservo[i].write(segmentOn[i]);
  }
  delay(5000);
}
```

Slika 3.10. Funkcija dodjeljivanja servo motora ispravnom pinu

Odbrojavanje se provodi pomoću četiri petlje. Vanjska petlja upravlja deseticama, a unutarnja petlja upravlja jedinicama. U svakoj od ovih petlji manja petlja prolazi kroz sedam segmenata i postavlja ih na ispravan položaj, uključen (*on*) ili isključen (*off*), provjeravajući vrijednost u nizu znamenki za odgovarajuću znamenku.

```
for (int h = 6; h >= 0; h--)
{
  if (digits[g][h] == 1)
    myservo[h + 7].write(segmentOn[h + 7]);
  else
    myservo[h + 7].write(segmentOff[h + 7]);
  if (h == 6)
    delay(200);
}
```

Slika 3.11. Mala petlja za upravljanje jedinicama

```
for (int i = 9; i >= 0; i--)
{
  int mustDelay2 = 0;
  if (digits[i][6] != digits[i + 1][6])
  {
    if (digits[i + 1][1] == 1) {
      myservo[1].write(segmentOn[1] - 30);
      mustDelay2 = 1;
    }
    if (digits[i + 1][5] == 1) {
      myservo[5].write(segmentOn[5] + 30);
      mustDelay2 = 1;
    }
  }
  if (i == 9) {
    myservo[1].write(segmentOn[1] - 30);
    myservo[5].write(segmentOn[5] + 30);
  }
  if (mustDelay2 == 1)
    delay(200);
```

Slika 3.12. Velika petlja za upravljanje deseticama

Ostala logika u ovom dijelu služi isključivo za pomaknuti dva segmenta koji su susjedni s središnjim segmentom, kako bi se udaljili za 30 stupnjeva kada se taj segment pomakne, kako se

ne bi sudarali. To uključuje sve varijable "mustDelay", kašnjenja od 200 milisekundi i *if* izjave koje nisu unutar manjih petlji.

3.4. Program za prikazivanje temperature

Predstavljeni kod kombinira upotrebu DS18B20 senzora za mjerenje temperature i 14 servo motora za prikazivanje temperature u sedam segmentnom formatu. U osnovi, kod čita temperaturu s DS18B20 senzora, a zatim koristi servo motore kako bi prikazao svaku znamenku temperature. Kod počinje s uključivanjem potrebnih biblioteka*: OneWire.h, DallasTemperature.h* i *Servo.*h. Zatim je definiran digitalni pin (18) na kojem je senzor spojen te deklariran na objekte *OneWire* i *DallasTemperature* s tim pinom. Također, deklariran je objekt Servo za kontrolu 14 servo motora. U *setup* funkciji, inicijalizirana je serijska komunikacija na brzini 9600 bita po sekundi (bps) putem *Serial.begin(9600).* Zatim je inicijaliziran senzor pomoću *sensors.begin()* te je dodano i sekundno kašnjenje kako bi se senzor stigao inicijalizirati. Servo motori su postavljeni pomoću *myservo[i].attach(pin),* vidljivo na slici 3.10. i postavljeni su u početni položaj segmenta koji predstavlja broj 8 na prikazu (88 prikazano) kako bi se testiralo da li se svi servo motori ispravno povezuju. Petlja počinje s pozivanjem *sensors.requestTemperatures()* kako bi zatražila novo očitanje temperature od senzora. Temperatura se dohvaća pomoću *sensors.getTempCByIndex(0),* a provjerava se je li očitana vrijednost valjana pomoću *DEVICE_DISCONNECTED_C*. Nakon što je dobivena očitana temperaturu u celzijusima, izdvojene su desetice i jedinice temperature pomoću jednostavnih matematičkih operacija. Tako su pripremljeni brojevi za prikaz na servo motorima.

Funkcija *displayTemperature* služi za prikazivanje temperaturu na servo motorima. Ova funkcija je ključni dio koda gdje se postavljaju servo motori kako bi prikazivali odgovarajuće segmente za prikaz znamenki temperature.

```
void loop() {
  sensors.requestTemperatures();
  int temperatureC = sensors.getTempCByIndex(0);
  if (temperatureC != DEVICE_DISCONNECTED_C) {
    Serial.print("Temperatura: ");
    Serial.print(temperatureC);
    Serial.println(" °C");
  } else {
    Serial.println("Senzor nije pronađen!");
  }

  int tensDigit = int(temperatureC) / 10;   // Desetice
  int unitsDigit = int(temperatureC) % 10;  // Jedinice

  displayTemperature(tensDigit, unitsDigit);
  delay(2000);
}
```

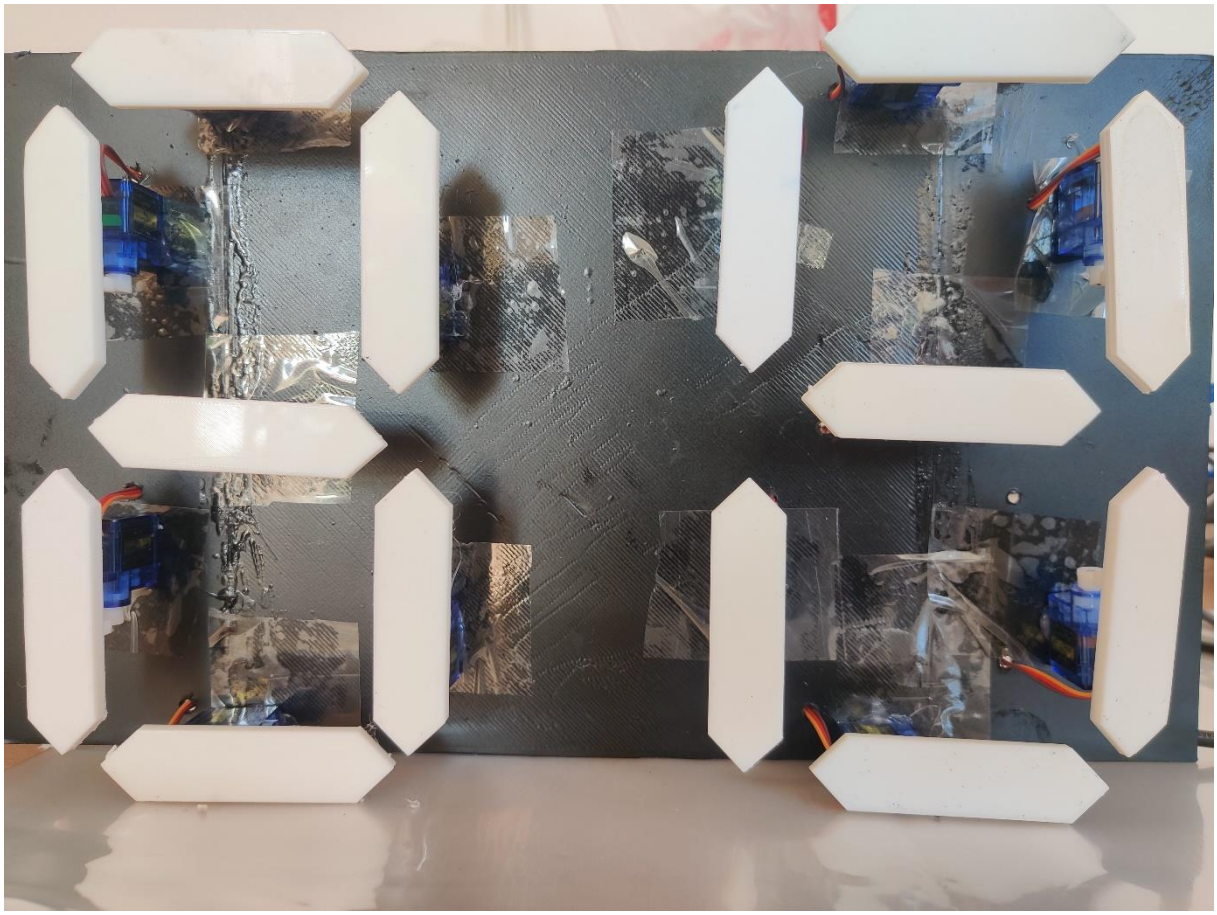
Slika 3.13. Glavna funkcija

Funkcija *displayTemperature* se temelji na originalnom kodu za prikazivanje brojeva, ali je prilagođena prikazu temperature. Slično kao kod prikaza brojeva, korištena je logika za postavljanje servo motora na odgovarajuće pozicije. Prvo se provjerava treba li dodatno kašnjenje kako bi se provjeri segment koji se pomiče radi oslobađanja prostora za novu znamenku (slično kao kada se mijenjaju desetice). Zatim se postavljaju servo motori kako bi prikazivali segmente za desetice i jedinice temperature, uz odgovarajuća kašnjenja između prikaza.

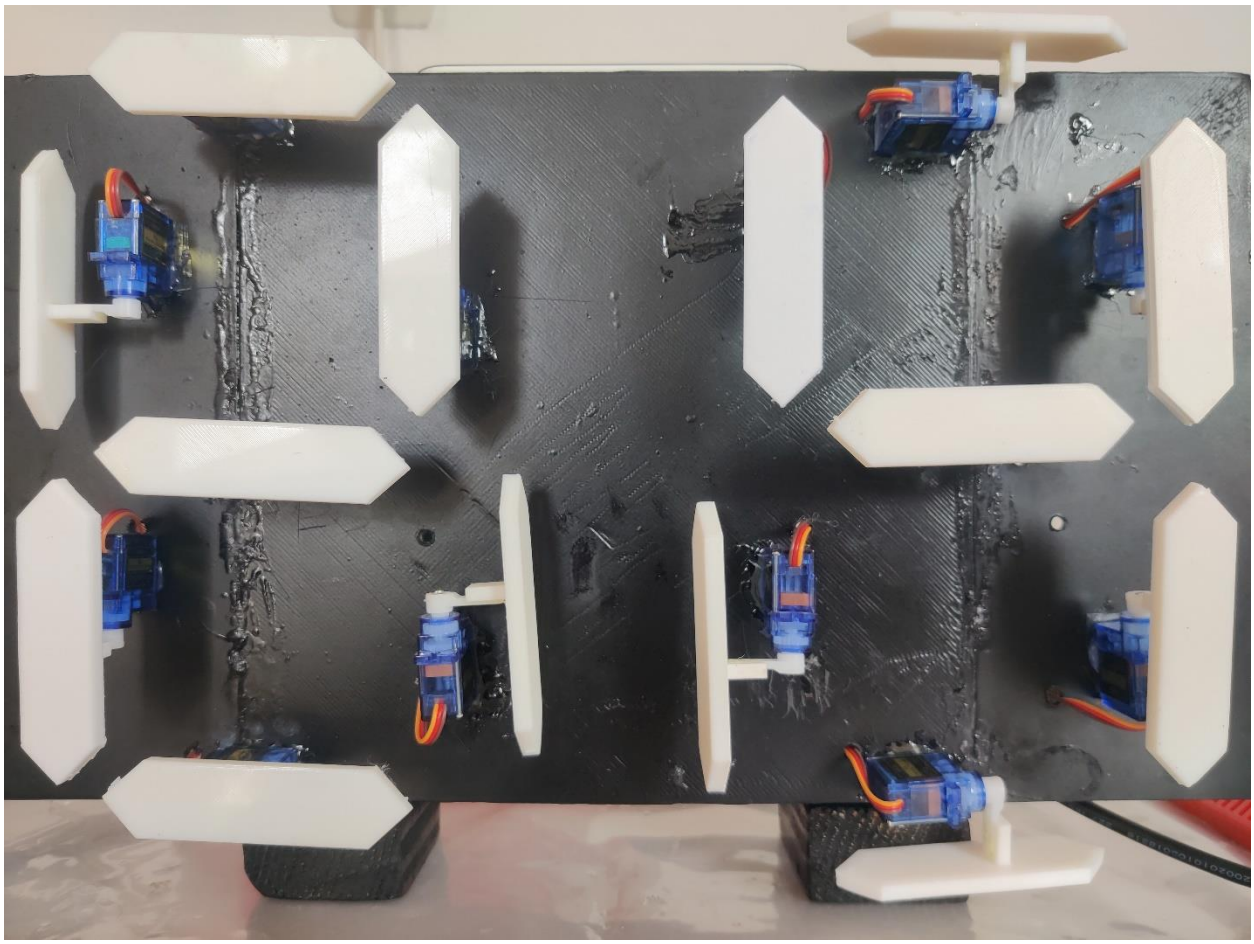
Ovaj kod omogućava čitanje temperature putem DS18B20 senzora i prikazivanje desetica i jedinica temperature koristeći 14 servo motora za sedam segmentni prikaz. Kroz analizu svakog koraka u kodu, jasno je kako se svaka komponenta koristi za postizanje ovog cilja. Također, primijećeno je da je pristup prikazu temperature sličan pristupu prikazu brojeva na segmentnom pokazivaču, s dodatnom logikom za upravljanje segmentima i kašnjenjima. Kombinacija senzora, servo motora i prateće logike omogućava kreativan i funkcionalan prikaz temperature u stvarnom vremenu.

3.5. Pokretanje sedam segmentnog mehaničkog pokazivača

Jednom kada je kod postavljen na Arduino i zaslon je spreman za pokazivanje. Automatski se inicijalizira na početnu vrijednost 88 (slika 3.14.) te pričeka pet sekundi i onda pokazuje određenu temperaturu koja je izmjerena. Također može se primijeniti da odbrojava od 99 do 0 odnosno od 0 do 99. U slučaju ako je korišten kod da odbrojava, jednom kada dođe do nula čeka dvije sekunde, restartira se te kreće iz početka. Naravno, to se sve može promijeniti u samome kodu. Na primjer ako je potrebno promijeniti početni broj od kojega se odbrojava potrebno je postaviti brojač u vanjskoj petlji. Uvijek je potrebno da se kreće od broja devet u znamenki jedinica, inače svaki polazak preskače znamenke veće od postavljenog broja. Na primjer, ne može se postaviti da 67 bude početno, nego već 69, a ako se stavi 67, onda ide unatrag 61, 60, 57, itd.



Slika 3.14. Početna inicijalizacija



Slika 3.15. Očitavanje temperature u vrijednosti od 24 stupnjeva

4. ZAKLJUČAK

U ovome diplomskome radu istražen je i razvijen sustav sedam-segmentnog mehaničkog pokazivača za prikazivanje izmjerene temperature prostorije koristeći 14 servo motora SG90 i senzor temperature . U radu su korištene različite tehnologije, s ciljem stvaranja prilagodljivog i preciznog sustava koji omogućuje korisnicima da lako prate temperaturu prostorije, odnosno odbrojavanja od 99 do 0 i obratno. U pregledu literature, istaknuta je važnost sedam-segmentnih mehaničkih pokazivača u različitim primjenama te upravljanje servo motorima kao pouzdanim sredstvom za precizno pozicioniranje pokazivača. Implementacija sustava temelji se na kombinaciji tehničkih značajki i logike upravljanja, koja omogućuje pretvaranje digitalne temperature u odgovarajući položaj pokazivača. Rezultati testiranja pokazali su visoku preciznost sustava u prikazivanju temperature, što ga čini pouzdanim rješenjem za praćenje okolne klime. Unatoč postignutom uspjehu, prepoznaje se da postoji prostor za daljnje poboljšanje i optimizaciju performansi. Projektiranje i izrada ovog sustava donijeli su različite izazove, uključujući kalibriranje pokazivača, integriranje senzora temperature i sinkronizaciju 14 servo motora. Međutim, ovi izazovi su pružili vrijedne spoznaje o upravljanju višestrukim servo motorima i složenosti koja proizlazi iz takvih aplikacija. Ovaj rad potvrđuje da je sedam-segmentni mehanički pokazivač sa servo motorima SG90 učinkovito rješenje za prikazivanje temperature prostorije te naravno i odbrojavanja. Daljnje poboljšanje učinkovitosti i optimizacija troškova mogu dodatno unaprijediti ovu tehnologiju za široku primjenu u industriji, kućanstvima i drugim područjima.

LITERATURA

[1] D.R. Patrick, S.W. Fardo, V. Chandra, *Electronic Digital System Fundamentals*, Fairmont Press, 2008.

[2] A. Kent Stiffler, *Design with Microprocessors for Mechanical Engineers*, McGraw-Hill, the University of Michigan, 2007.

[3] C. Wook Han, I. B. Kang, J. K. Jeong, *Advanced Display Technology*, Springer Nature Singapore, 2021.

[4] R. K. Jurgen, *Society of Automotive Engineers*, History of Automotive Electronics, 1998.

[5] A. Arockia Bazil Raj, *Based Embedded System Developer's Guide*, CRC Press, 2018.

[6] Sulistyanto, "*Design of 4-digit Count-Up with 7-Segment 5-Inch SM415001L*," journal of robotics and control, 2020.

[7] M. Margolis, B. Jepson, N. R. Weldin, *Arduino Cookbook*, O'Reilly Media, 2020

[8] M. Banzi, D. Cuartielles, *Arduino Mega - Official Documentation*, O'Reilly Media, Inc., 2005.

[9] B. Israel, *Introduction to Arduino Projects*, Independently Published, 2021.

[10] A. Poorve, *What is a Servo Motor? - Understanding the basics of Servo Motor Working*. Dostupno na: https://circuitdigest.com/article/servo-motor-working-and-basics [20.8.2023.]

[11] M. Fezari i A.A. *Dahoud, Integrated Development Environment "IDE" For Arduino*, Al Zaytoona University, Amman, Jordan, 2015.

[12] M. Montrose, EMC and the Printed Circuit Board, IEEE Press, 1999.

[13] A B Lawal, *PCB Design & Layout For DIY Etching*, Independently Published, 2021.

[14] Kokane, D. D., Bhosale, S. P., & Lokhande, S. S., *Microcontroller-Based Digital Clock with Seven Segment Mechanical Display*, 2014.

SAŽETAK

SEDAM SEGMENTNI MEHANIČKI POKAZIVAČ

Ovaj diplomski rad istražuje i razvija sustav sedam-segmentnog mehaničkog pokazivača koji koristi 14 servo motora SG90 za prikazivanje trenutne temperature prostorije. Cilj je stvoriti prilagodljiv i točan sustav koji omogućuje korisnicima lako praćenje temperature. U radu je prikazan pregled literature o sedam-segmentnim mehaničkim pokazivačima i upravljanju servo motorima kao pouzdanim sredstvom za pozicioniranje pokazivača. Detaljno su opisane tehnologija i alati korišteni u projektu te implementacija sustava. Rezultati testiranja potvrđuju visoku preciznost sustava u prikazivanju temperature, uz prepoznavanje potrebe za daljnjim poboljšanjem. Projektiranje i izrada sustava iznijeli su izazove u upravljanju višestrukim servo motorima. Zaključno, ovo istraživanje potvrđuje učinkovitost sedam-segmentnih mehaničkih pokazivača sa servo motorima SG90 u prikazivanju temperature, s naglaskom na potencijal za daljnji razvoj i primjenu u različitim područjima.

Ključne riječi: Arduino Mega, SG90 motor, 3D printer, PCB, EasyEDA

ABSTRACT

SEVEN SEGMENT MECHANICAL DISPLAY

This thesis explores and develops a system of a seven-segment mechanical indicator using 14 SG90 servo motors for accurate display of the current room temperature. The aim is to create an adaptable and precise system that enables users to easily monitor the temperature. The thesis presents a literature review on seven-segment mechanical indicators and the use of servo motors as reliable means for positioning the indicators. Detailed descriptions of the technology and tools used in the project, as well as the system's implementation, are provided. The testing results confirm the high precision of the system in temperature display, with recognition of the need for further improvements. The design and construction of the system presented challenges in managing multiple servo motors. In conclusion, this research validates the effectiveness of seven-segment mechanical indicators with SG90 servo motors in temperature display, with an emphasis on their potential for further development and application in various fields.

Keywords: Arduino Mega, SG90 motor, 3D printer, PCB, EasyEDA

ŽIVOTOPIS

Matija Vrabelj rođen je 27.08.1998. u Koprivnici. Godine 2005. upisuje Osnovnu školu braće Radić te istu završava 2013. godine. Svoje srednjoškolsko obrazovanje započinje 2013. godine kada upisuje Obrtničku školu Koprivnica smjer tehničar za računarstvo te ga završava u redovnom roku 2017. godine. Po završetku srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek gdje upisuje stručni studij elektrotehnike, smjer informatika. Navedeni studij završava u redovnom roku 2020. godine sa vrlo dobrim uspjehom te vođen tim uspjehom upisuje razlikovne obveze koje završava 2021. godine te upisuje diplomski studij elektrotehnike, smjer komunikacije i informatika kojega završava 2023. godine, također u redovnom roku. Tijekom svojih studentskih dana konstantno je upisivao razne tečajeve kako bi usavršio svoje znanje te ga u konačnici i proširio. To su tečajevi online verzije u svrhu programiranja. Također tijekom cijelog svojega studiranja izvršavao je razne studentske poslove kao što je bio rad u tvornici Koktel koji je sklopu Podravke, prodajni predstavnik u Hrvatskom telekomu te kao distributer event opreme u Carlsberg Hrvatska. Uz hrvatski jezik također poznaje i engleski jezik u pričanju i govoru. Jednu od velikih osobina valja istaknuti upornost, nošenje s pritiskom, energičnost, entuzijazam, komunikativnost i spremnost za rad u timu.

PRILOG

Prilog 1. Mjerenje temperature

```
#include <OneWire.h>

#include <DallasTemperature.h>

#include <Servo.h>


// Povezivanje senzora na digitalni pin 18

#define ONE_WIRE_BUS 18

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);


Servo myservo[14];


int segmentOn[14] = { 180, 180, 90, 85, 20, 100, 10, 180, 98, 5, 85, 10, 80, 100 }; //Servo on position values for each servo

int segmentOff[14] = { 90, 88, 180, 180, 100, 180, 105, 80, 180, 99, 180, 90, 180, 10 }; //Servo off position values for each servo

int digits[10][7] = { { 1, 1, 1, 1, 1, 1, 0 }, { 0, 1, 1, 0, 0, 0, 0 }, { 1, 1, 0, 1, 1, 0, 1 }, { 1, 1, 1, 1, 0, 0, 1 }, { 0, 1, 1, 0, 0, 1, 1 }, { 1, 0, 1, 1, 0, 1, 1 }, { 1, 0, 1, 1, 1, 1, 1 }, { 1, 1, 1, 0, 0, 1, 0 }, { 1, 1, 1, 1, 1, 1, 1 }, { 1, 1, 1, 1, 0, 1, 1 } };  //Position values for each digit


void setup() {


  Serial.begin(9600);

  sensors.begin();
```

```
 myservo[0].attach(2);
  myservo[1].attach(3);
  myservo[2].attach(4);
  myservo[3].attach(5);
  myservo[4].attach(6);
  myservo[5].attach(7);
  myservo[6].attach(8);
  myservo[7].attach(9);
  myservo[8].attach(10);
  myservo[9].attach(11);
  myservo[10].attach(12);
  myservo[11].attach(13);
  myservo[12].attach(45);
  myservo[13].attach(44);
  for (int i = 0; i <= 13; i++)  //Set all of the servos to on or up (88 displayed)
  {
    myservo[i].write(segmentOn[i]);
  }
  delay(5000);
}


void loop() {
  sensors.requestTemperatures();                // Zatraži novu temperaturu od senzora
```

```
  int temperatureC = sensors.getTempCByIndex(0);  // Dohvati temperaturu u Celzijusima
  if (temperatureC != DEVICE_DISCONNECTED_C) {
    Serial.print("Temperatura: ");
    Serial.print(temperatureC);
    Serial.println(" °C");
  } else {
    Serial.println("Senzor nije pronađen!");
  }
  // Prikaz temperature na segmentnom displeju
  int tensDigit = int(temperatureC) / 10;   // Desetice
  int unitsDigit = int(temperatureC) % 10;  // Jedinice

  displayTemperature(tensDigit, unitsDigit);
  delay(2000);
}

void displayTemperature(int tens, int units) {
  int mustDelay1 = 0;

  if (digits[tens][6] != digits[tens + 1][6]) {
    if (digits[tens + 1][1] == 1) {
      myservo[8].write(segmentOn[8] - 30);
      mustDelay1 = 1;
    }
```

```
  if (digits[tens + 1][5] == 1) {
    myservo[12].write(segmentOn[12] + 30);
    mustDelay1 = 1;
  }
}

if (tens == 9) {
  myservo[8].write(segmentOn[8] - 30);
  myservo[12].write(segmentOn[12] + 30);
}

if (mustDelay1 == 1)
  delay(200);

for (int h = 6; h >= 0; h--) {
  if (digits[tens][h] == 1)
    myservo[h + 7].write(segmentOn[h + 7]);
  else
    myservo[h + 7].write(segmentOff[h + 7]);

  if (h == 6)
    delay(200);
}
```

```
int mustDelay2 = 0;

if (digits[units][6] != digits[units + 1][6]) {
  if (digits[units + 1][1] == 1) {
    myservo[1].write(segmentOn[1] - 30);
    mustDelay2 = 1;
  }
  if (digits[units + 1][5] == 1) {
    myservo[5].write(segmentOn[5] + 30);
    mustDelay2 = 1;
  }
}

if (units == 9) {
  myservo[1].write(segmentOn[1] - 30);
  myservo[5].write(segmentOn[5] + 30);
}

if (mustDelay2 == 1)
  delay(200);

for (int j = 6; j >= 0; j--) {
  if (digits[units][j] == 1)
    myservo[j].write(segmentOn[j]);
```

```
    else
      myservo[j].write(segmentOff[j]);

    if (j == 6)
      delay(200);
  }

  if (mustDelay2 == 1)
    delay(600);
  else
    delay(800);
}
```

Prilog 2. Odbrojavanje

```
#include <Servo.h>

Servo myservo[14];

int segmentOn[14] = { 90, 88, 81, 80, 90, 78, 98, 90, 88, 100, 75, 0, 0, 180 }; //Servo on position values for each servo

int segmentOff[14] = {180 , 180, 0, 15, 180, 160, 180, 180, 180, 180, 0, 10, 90, 80 }; //Servo off position values for each servo

int digits[10][7] = { { 1, 1, 1, 1, 1, 1, 0 }, { 0, 1, 1, 0, 0, 0, 0 }, { 1, 1, 0, 1, 1, 0, 1 }, { 1, 1, 1, 1, 0, 0, 1 }, { 0, 1, 1, 0, 0, 1, 1 }, { 1, 0, 1, 1, 0, 1, 1 }, { 1, 0, 1, 1, 1, 1, 1 }, { 1, 1, 1, 0, 0, 1, 0 }, { 1, 1, 1, 1, 1, 1, 1 }, { 1, 1, 1, 1, 0, 1, 1 } };  //Position values for each digit

void setup() {
```

```
  myservo[0].attach(2);
  myservo[1].attach(3);
  myservo[2].attach(4);
  myservo[3].attach(5);
  myservo[4].attach(6);
  myservo[5].attach(7);
  myservo[6].attach(8);
  myservo[7].attach(9);
  myservo[8].attach(10);
  myservo[9].attach(11);
  myservo[10].attach(12);
  myservo[11].attach(13);
  myservo[12].attach(45);
  myservo[13].attach(44);
  for (int i = 0; i <= 13; i++)
  {
    myservo[i].write(segmentOn[i]);
  }
  delay(5000);
}

void loop() {
  for (int g = 9; g >= 0; g--)  //Large loop counter to count the tens
  {
```

```
    int mustDelay1 = 0;
    if (digits[g][6] != digits[g + 1][6])  //Logic to move segments next to middle segment out of the way
    {
      if (digits[g + 1][1] == 1) {
        myservo[8].write(segmentOn[8] - 30);
        mustDelay1 = 1;
      }
      if (digits[g + 1][5] == 1) {
        myservo[12].write(segmentOn[12] + 30);
        mustDelay1 = 1;
      }
    }
    if (g == 9) {
      myservo[8].write(segmentOn[8] - 30);
      myservo[12].write(segmentOn[12] + 30);
    }
    if (mustDelay1 == 1)
      delay(200);
    for (int h = 6; h >= 0; h--)                    //Small loop counter to move individual segments  to make up the tens digit
    {
      if (digits[g][h] == 1)
        myservo[h + 7].write(segmentOn[h + 7]);
      else
```

```
      myservo[h + 7].write(segmentOff[h + 7]);
    if (h == 6)
      delay(200);
  }
  for (int i = 9; i >= 0; i--)                    //Large loop counter to count the units
  {
    int mustDelay2 = 0;
    if (digits[i][6] != digits[i + 1][6])                  //Logic to move segments next to
middle segment out of the way
    {
      if (digits[i + 1][1] == 1) {
        myservo[1].write(segmentOn[1] - 30);
        mustDelay2 = 1;
      }
      if (digits[i + 1][5] == 1) {
        myservo[5].write(segmentOn[5] + 30);
        mustDelay2 = 1;
      }
    }
    if (i == 9) {
      myservo[1].write(segmentOn[1] - 30);
      myservo[5].write(segmentOn[5] + 30);
    }
    if (mustDelay2 == 1)
      delay(200);
```

```
    for (int j = 6; j >= 0; j--)  //Small loop counter to move individual segments  to make up the unit digit
    {
      if (digits[i][j] == 1)
        myservo[j].write(segmentOn[j]);
      else
        myservo[j].write(segmentOff[j]);
      if (j == 6)
        delay(200);
    }
    if (mustDelay2 == 1)  //Delay logic to reduce delay if the side segments moved (adding an additional 200ms delay)
      delay(600);
    else
      delay(800);  //Delay between digits. 200ms delay already experienced in the code
   }
  }
 delay(2000);  //Delay after countdown to 0 before resetting
}
```