Maketa vertikalnog transportnog sustava u zgradarstvu

Cindrić, Hrvoje

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

MAKETA VERTIKALNOG TRANSPORTNOG SUSTAVA U ZGRADARSTVU

Diplomski rad

Hrvoje Cindrić

Osijek, 2016.

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek,
2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Hrvoje Cindrić
Studij, smjer:	Sveučilišni studij računarstva, procesno računarstvo
Mat. br. studenta, godina	D-693 R, 2014.
Mentor:	Doc.dr.sc. Tomislav Keser
Sumentor:	
Predsjednik Povjerenstva:	
Član Povjerenstva:	
Naslov diplomskog rada:	Maketa vertikalnog transportnog sustava u zgradarstvu
Primarna znanstvena grana rada:	Procesno računarstvo
Sekundarna znanstvena grana (ili polje) rada:	Računarstvo
Zadatak diplomskog rada:	Projektirati, izraditi i evaluirati sustav za vertikalni transport u modelskoj zgradi s 4 kata i prizemljem. Ugraditi zaštitne mehanizme od preopterećenja kabine te izraditi sustav za udaljeni nadzor.
Prijedlog ocjene pismenog dijela ispita (diplomskog	
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: Postignuti rezultati u odnosu na složenost zadatka: Jasnoća pismenog izražavanja: Razina samostalnosti:

Potpis sumentora:	Potpis mentora:

Dostaviti:

1. Studentska služba

U Osijeku, 2016. godine Potpis predsjednika Odbora:

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

IZJAVA O ORIGINALNOSTI RADA

Osijek,
2016.

Ime i prezime studenta:	Hrvoje Cindrić
Studij :	Sveučilišni studij računarstva, procesno računarstvo
Mat. br. studenta, godina upisa:	D-693 R, 2014.

Ovom izjavom izjavljujem da je rad pod nazivom**:**

Maketa vertikalnog transportnog sustava u zgradarstvu

izrađen pod vodstvom mentora: **Doc.dr.sc. Tomislava Kesera**

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Prije 200 godina nitko ne bi mogao ni zamisliti da će milijuni ljudi živjeti i raditi u zgradama koje su toliko visoke da se čovjek pješice ne može lagano popeti na posljednje katove. Danas ne možemo zamisliti velike nebodere u gradu bez dizala. Dizalo ili lift je izmišljeno prije više od skoro dva stoljeća. U 19. stoljeću u New Yorku su imali hidraulična dizala. Kabina, odnosno platforma, bila je montirana na vrhu dugačkog klipa koji se nalazio u cilindru. Da bi se dizalo uspinjalo, odozdo je u taj cilindar pumpana voda, a da bi se spuštalo, otvarana je slavina za ispuštanje vode iz cilindra. Ta je voda odvođena natrag u rezervoar i mogla se iznova koristiti što je bilo dosta praktično. Dizala ove vrste danas uglavnom više nisu u upotrebi. Ta su dizala bila spora, a pošto šipka što podiže kabinu mora ulaziti čitava okomito u zemlju, takva se dizala ne mogu koristiti u visokim zgradama jer zahtijevaju duboke ukope. Gradnja visokih zgrada postala je moguća upravo zahvaljujući električnom dizalu. Takvo se dizalo podiže pomoću kablova koji se namotavaju oko bubnja na vrhu zgrade. Kod najnovijih dizala bubanj je zamijenjen jednostavnom koloturom koju izravno pokreće motor. Suvremena dizala imaju mnoge naprave čiji je cilj sprječavanje nesretnih slučajeva. Jedna od tih naprava je zračni jastuk na dnu otvora. Kad se dizalo spušta, platforma sve više zatvara presjek okomitog otvora da bi što manje zraka izašlo vani. Posljedica toga je stvaranje zračnog jastuka, odnosno usporavanje pada kabine.

1.1. Zadatak i struktura rada

Zadatak ovoga diplomskog rada je projektirati, izraditi i evaluirati sustav za vertikalni transport u zgradarstvu. Sustav projektirati na način da opslužuje transportne zahtjeve u modelskoj zgradi s 4 kata i prizemljem. Ugraditi zaštitne mehanizme od preopterećenja kabine te dodati ostale najkorištenije mehanizme i metode u prijevozu. Izraditi sustav za udaljeni nadzor i upravljanje ovim transportnim sustavom.

Sustav u toku izrade prolazi kroz pet faza, u prvoj fazi izrađuje se mehanički ustroj sustava, odnosno konstrukcija modelske zgrade koja mora opsluživati transportne zahtjeve za 4 kata i prizemlje. Ovom dijelu još pripada izrada kabine koja je glavni element za vertikalni prijevoz između katova. U drugoj fazi izrade izrađuje se električki ustroj sustava, tj. ugradnja svih električnih elemenata kao što su senzori, aktuatori, mikroupravljač, lcd zaslon, zaštitni mehanizam od preopterećenja kabine, tipkala za pozivanje, svjetlosni i zvučni signalizatori te se provodi razvlačenje bakrenih vodića kako bi se uspostavila komunikacija izmedju svih tih komponenti. Treća i četvrta faza je programiranje, tj. izrada softverskog rješenja. Izrađuju se dva programa, prvim programom programira se mikroupravljač koji će poslije i upravljati ovim vertikalnim transportnim sustavom. Drugi program predstavlja računalni sustav za udaljeni nadzor i upravljanje, dva programa se povezuju serijskom komunikacijom kako bi mogli sinkronizirano raditi. U petoj fazi sustav se pušta u pogon te se testira njegov rad i ispravljaju pogreške te određuje pouzdanost elektroničkog sklopa. O svakoj fazi izrade detaljnije se govori u narednim poglavljima.

2. TRANSPORTNI SUSTAVI U ZGRADARSTVU

2.1. O transportnim sustavima

Transportna ili prijevozna usluga je kretanje ljudi i dobara s jednog mjesta na drugo. Načini transporta jesu zračni, željeznički, cestovni, voda, kablovi, cjevovodi i prostor. Transportna polja mogu se podijeliti u tri grane: transportna infrastruktura, vozila i transportne operacije. Ona se ne može uskladištiti, sačuvati ili ponuditi na tržištu jer nema materijalni oblik kao što je slučaj sa samom robnom proizvodnjom.

Temeljni elementi razvoja transporta su sredstva za rad, predmet rada i radna snaga, odnosno sam rad.

Kroz povijest postojale su razne faze u transportnom procesu. Te se faze odnose na način rada i razvitak i napredovanje tehnologije za rad koja se koristi u procesima. Faze se mogu podijeliti u četiri skupine:

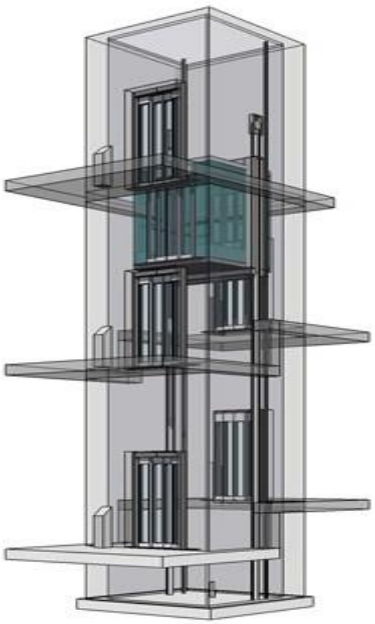
1. Manualizacija- najniži stupanj razvoja transportne tehnologije; u toj fazi koristi se samo fizička energija ljudi ili životinja za proizvodnju prometne usluge.
2. Mehanizacija- veže se uz otkriće parnog stroja i korištenje mehaničke energije. U toj fazi čovjek obavlja pripremne i završne operacije, a prijevoz se obavlja mehanički primjenom strojeva.
3. Automatizacija- prvi sustavi u kojima čovjek nadgleda proces koji je osmislio.
4. Automatika (robotika)- današnji stupanj razvoja transportnog inženjerstva- strojevi programiraju sve od ideje do realizacije [1].

2.2. Vertikalni transportni sustavi u zgradarstvu

Vertikalni transportni sustav u zgradarstvu ili dizalo je uređaj za prijevoz ljudi ili tereta među katovima zgrade ili radnih platformi. Uobičajno su pokretani elektromotorom koji ili pokreće užad za vuču i protutežni sustav, ili pumpa hidrauličnu tekućinu za podizanje cilindričnih klipova. „Prvo dizalo nastalo je 236 godine prije Krista, a konstruirao ga je Arhimed. Instalirano je u samostanu u Egiptu. Hidraulična dizala su jeftinija, ali instaliranje cilindara na veće visine postaje nepraktično. Često su izgrađena dva dizala tako da njihove kabine se uvijek kreću sinkronizirano u suprotnim smjerovima, te su jedna drugoj protuteža. Kod dizala važne su karakteristike nosivosti, brzine vožnje i dimenzije kabine. Dizala su našla

primjenu u mnogim prostorima. Najčešća primjena im je prijevoz putnika između katova u zgradama, prijevoz hrane i posuđa u restoranima sa nivoa pripreme do nivoa posluživanja, u bolnicama. Malo drugačije realizacije dizala koriste se u silosima za podizanje materijala. Takva dizala realizirana su pomoću lanaca na kojem su instalirane kante. Rana dizala nisu imala automatiku, niti automatsko pozicioniranje položaja kabine. Njima su upravljali operateri koji su upravljali elektro motorom. Novija dizala imaju instaliran sustav zaštite od kriminala. Zaštitaru ili recepcioneru je omogućeno zaustavljanje dizala na željenom katu te otvaranje vrata po njegovoj želji [2].

Dana 23. ožujka 1857. godine instalirano je prvo sigurnosno dizalo američkog industrijalca Elishe Otisa. Taj je izumitelj i poduzetnik postao slavan po pantentu koji je osigurao dizala od pada u slučaju pucanja kabela. Naime, Otisova dizala imala su sustav koji ih je automatski zaustavljao kad bi se počela naglo spuštati. Svoj je izum Otis predstavio u Kristalnoj palači u Londonu vrlo dramatičnom prezentacijom (navodno je osobno stajao u dizalu koje bi se rušilo pred gledateljima). Prvo Otisovo komercijalno dizalo instalirano je u zgradi trgovca E. V. Haughwouta u New Yorku, na adresi 488 Broadway, dovršenoj spomenute 1857. godine. To je dizalo bilo pokretano parnim strojem instaliranim u podrumu, a zgrada je imala ukupno pet etaža iznad razine zemlje. Premda se nije radilo o visokoj zgradi, Otisovo je dizalo spomenutom trgovcu služio za privlačenje mušterija, jer se radilo o velikom tehnološkom novitetu [3].



Slika 2.1. Modelski prikaz voznog okna vertikalnog transportnog sustava [4].

3. MAKETA VERTIKALNOG TRANSPORTNOG SUSTAVA U ZGRADARSTVU

3.1. Značajke i zahtjevi nad maketom

Osnovni cilj projekta ugradnje dizala je odrediti zahtjeve koja dizala u zgradi moraju ispuniti i značajke i svojstva dizala te veličine kojima dizalo djeluje na zgradu, na način koji mora osigurati ispunjavanje bitnih zahtjeva za građevinu, zahtjeve pristupačnosti te funkciju koju u građevini dizalo ima [5].

Konstrukcijski dio	Mora zadovoljavati uvjet da kabina transportnog sustava može opsluživati 4 kata i prizemlje u modelskoj zgradi.
Mikroupravljač	Arduino Mega 2560
Glavni senzori	Optički senzori TCRT5000
Aktuatori	DC motor 12V (Globe motor) Mini servo motor (Tower pro SG90)
Zaštitini mehanizam od preopterećenja voznog okna	Elektronska vaga 1Kg (*Load Cell* 5V)
Svjetlosni, zvučni signalizatori i lcd zaslon	Led diode 5mm crvene *Piezo Electronic Buzzer* 5V Lcd zaslon 16*2
Sustav za udaljeni nadzor i kontrolu transposrtnog susutava	Aplikacija u programskom jeziku C# (CSharp)
Ostalo	Tipkala za pozivanje kabine i kontrolu vratima kabine, bakreni vodići i ostali popratni materijal.

Tablica 3.1. Značajke i zahtjevi nad maketom

3.2. Sklopovsko rješenje sustava

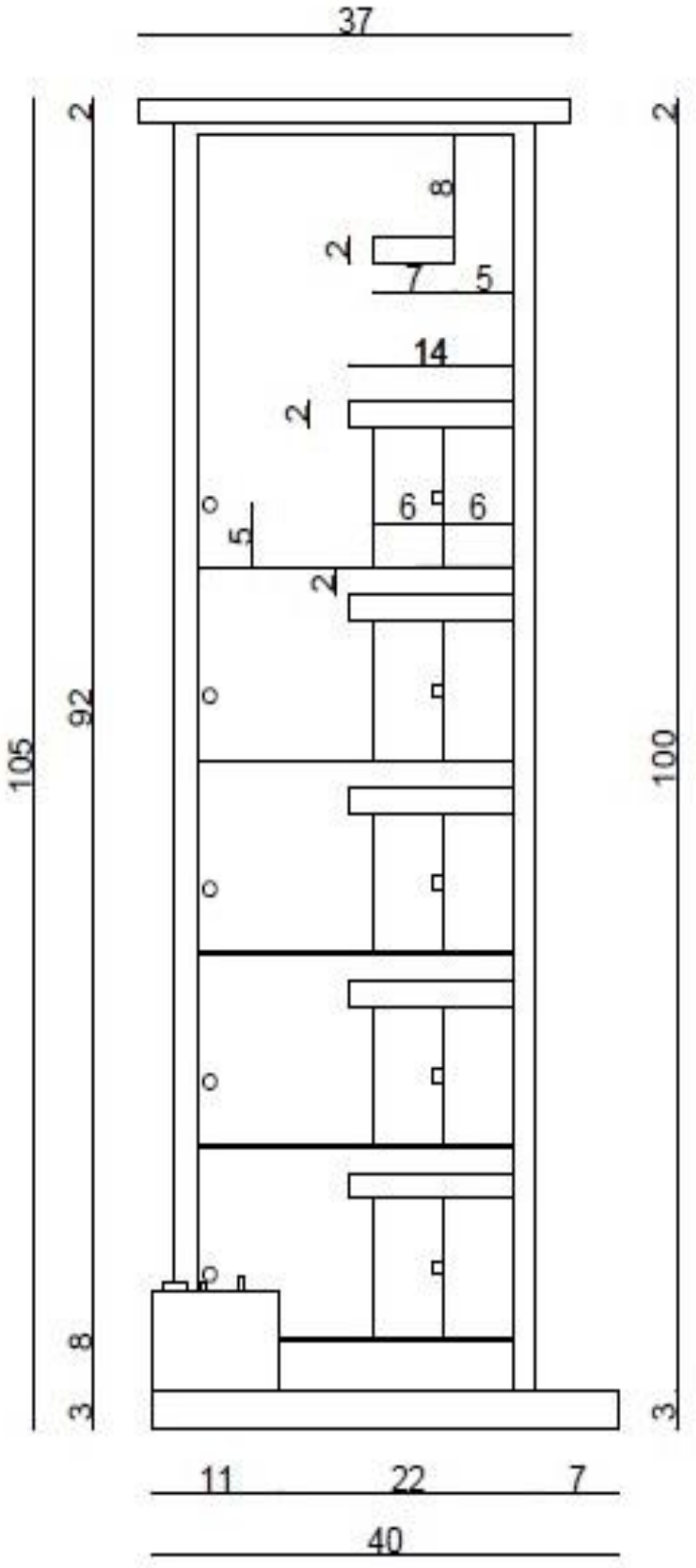
3.2.1. Mehanički ustroj makete

Cijela mehanička konstrukcija transposrtnog sustava kao i kabina dizala je ručni rad. Projektirana i izrađena u radionici korištenjem raznih materijala, drvenih, metalnih, plastičnih i pleksiglasa.

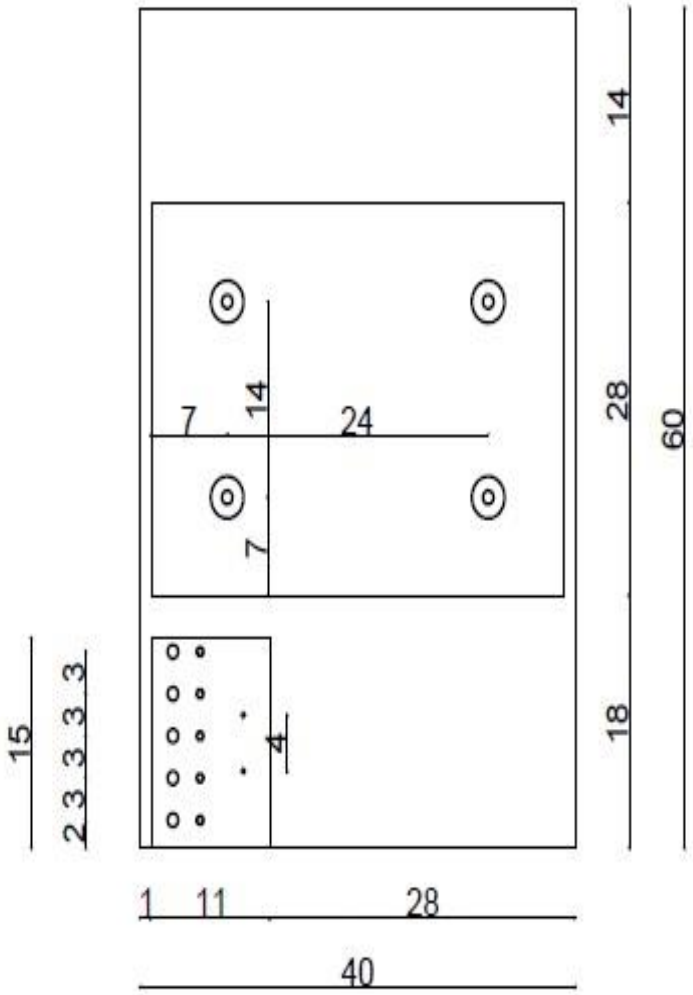
Postolje makete je napravljeno od komada kuhinjske radne ploče dimenzija 40cm x 60cm na koju su s njezine donje strane pričvršćene četiri noge. Glavni kostur konstrukcije čine četiri željezna kutna profila visine 95cm koji su pričvršćeni na postolje. Dok su na visini 70cm međusobno horizontalno povezani istim takvim kutnim profilima, vidljivo na slici 3.1. Na tom „drugom katu“ konstrukcije postolje je drvena ploča dimenzija 28cm x 40 cm s rupom na sredini kroz koju prolazi užad za vuču kabine. Na tom „drugom katu“ konstrukcije je smješten elektromotor kao i koloture na koje se namotava užad kada se kabina dizala giba vertikalno, također tu je smješten i mikroupravljač te adapter za napajanje cijelog sustava. Na glavnu konstrukciju je također pričvršćene i dvije plastične kanalice kroz koje prolaze bakreni vodići od tipkala, senzora i ostalih elektroničkih komponenti, te dvije metalne vodilice na kojima klizi kabina dizala kad se vertikalno giba. Obloga glavne konstrukcije s prednje i lijeve strane je napravljena od plastične ploče. Na prednjoj strani su izrezana vrata, ukupno njih pet, za četiri kata i prizemlje, ispod svakog proreza horizontalno je pričvršćen plastični „balkon“ kako bi se točno definiralo koji je koji kat. Također svaki prorez zatvaraju vrata napravljena od pleksiglasa i koja se ručno otvaraju, na prednjoj strani su jos instalirani svjetlosni signalizatori (led diode) te tipkala za pozivanje. Obloga na desnoj i zadnjoj strani napravljena je od pleksiglasa, prozirna je te je vidljivo vozno okno.

Kabina vertikalnog transportnog sustava je kučište običnog napajanja za računalo, svi dijelovi su izvađeni te je izrezan okvir za vrata, ugrađen je servo motor koji otvara i zatvara vrata, na dnu kabine je ugrađena vaga za određivanje težine kabine, dok je s gornje strane postavljene koloture preko kojih elektromotor i užad za vuču vertikalno gibaju kabinu od kata do kata. Na postolju je još montirana razvodna kutija u koju su smješteni svjetlosni signalizatori (led diode), tipkala za pozivanje katova i tipkala za upravljanje vratima kabine a koji predstavljaju kontrolu nad vertikalnim transportnim sustavom koja bi se trebala nalaziti u kabini.

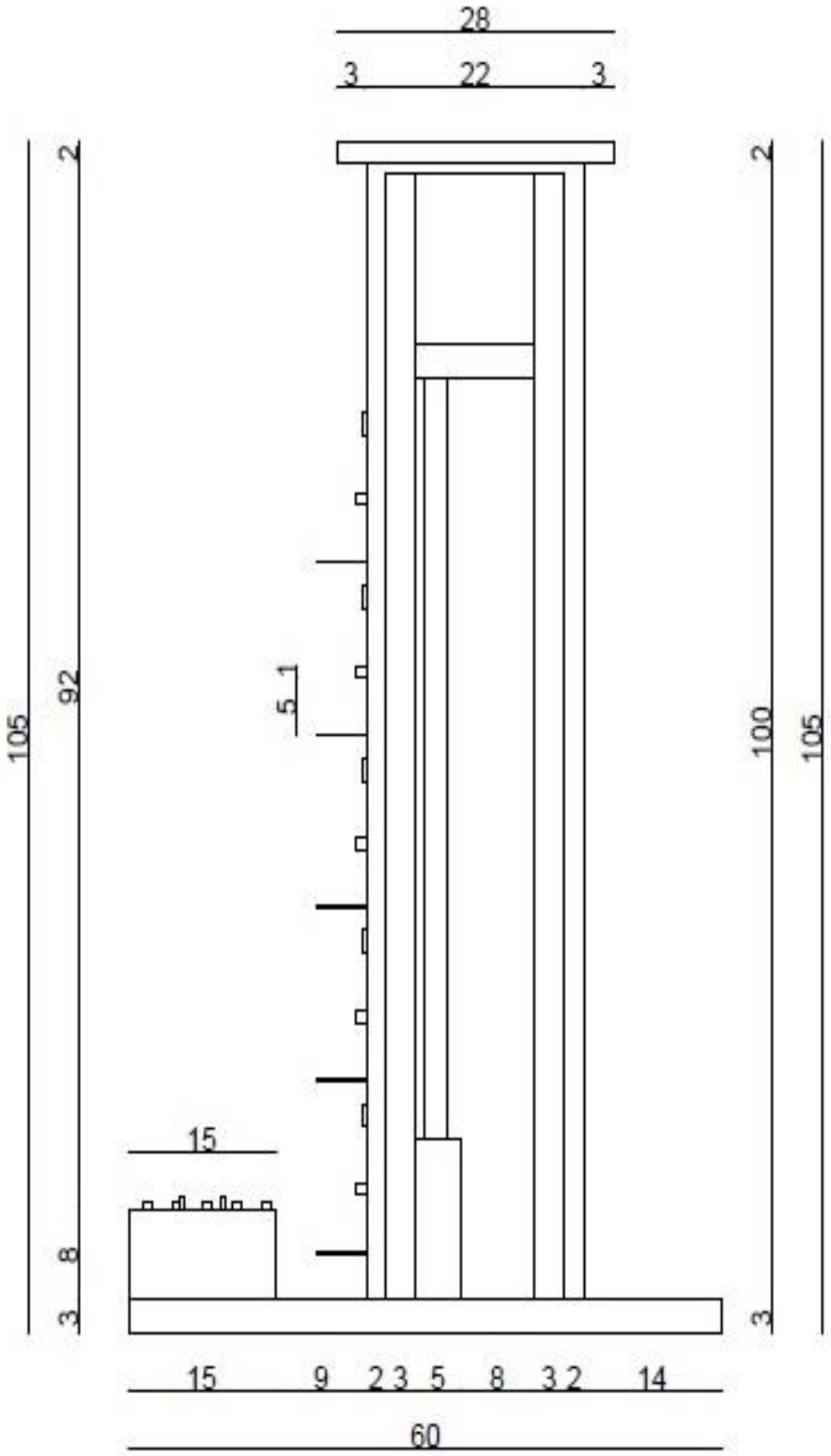
Slika 3.1. Maketa vertikalnog transportnog sustava za vrijeme izrade i nakon dovršetka.



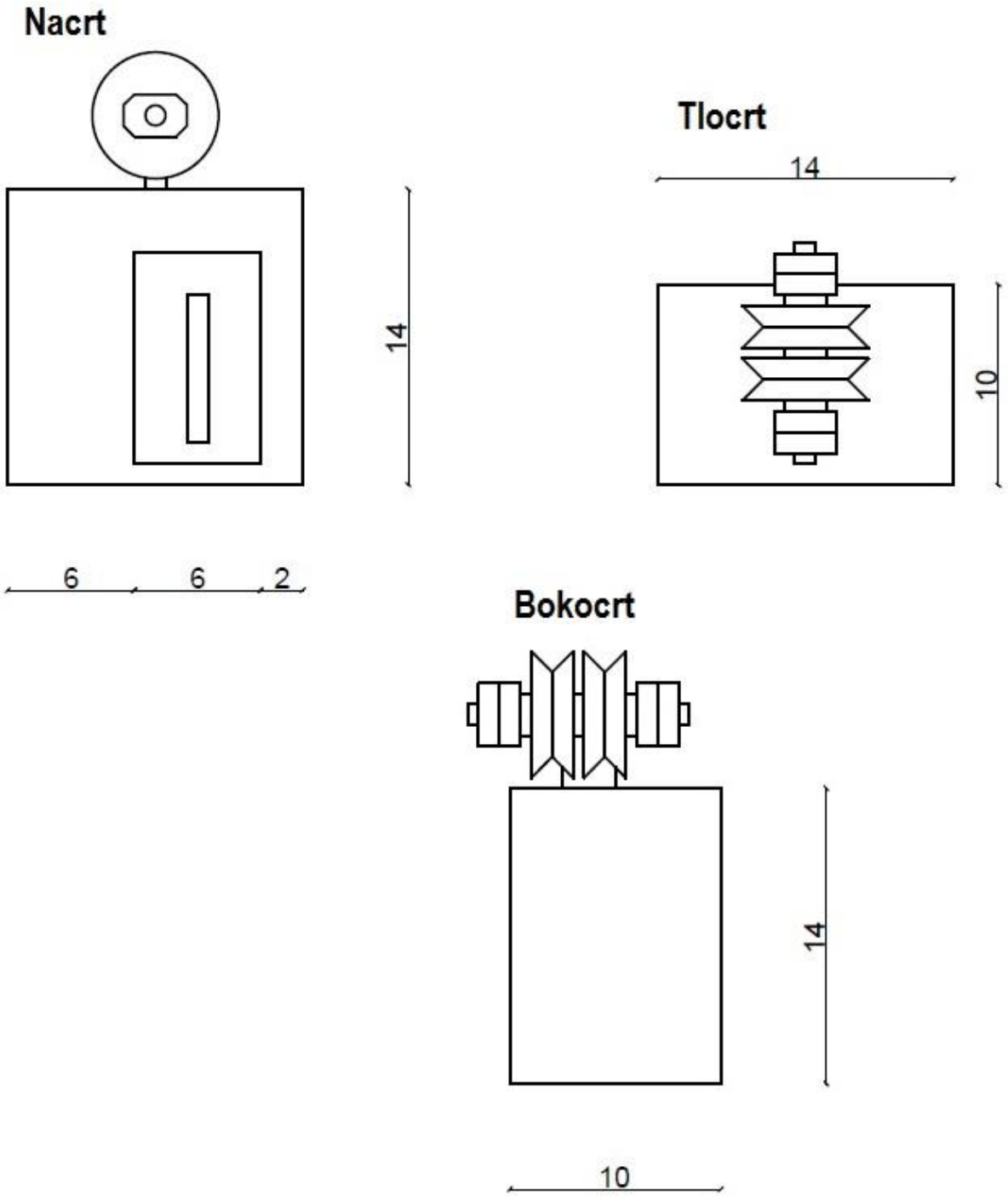
Slika 3.2. Nacrt modelske zgrade.



Slika 3.3. Tlocrt modelske zgrade.



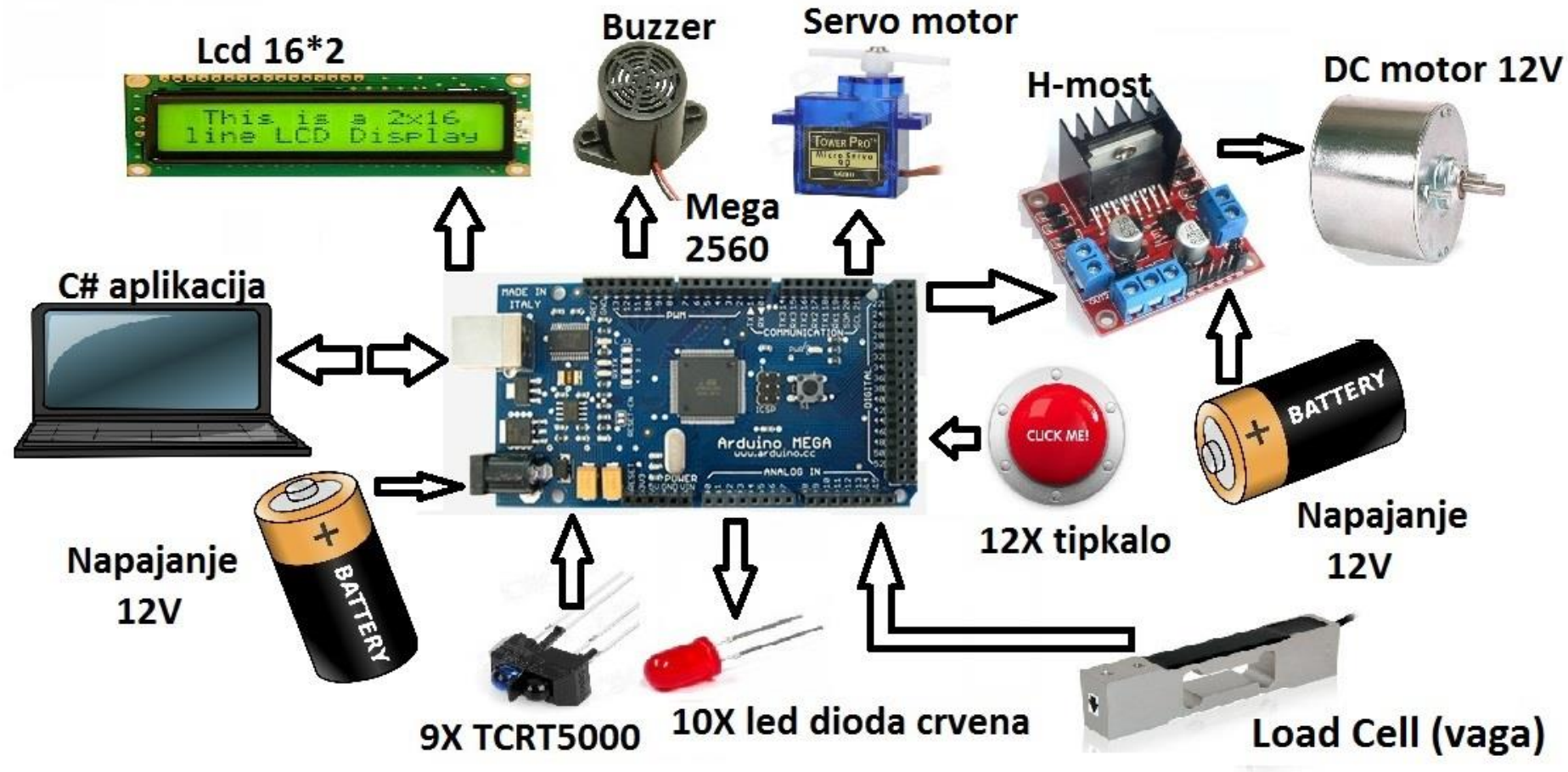
Slika 3.4. Bokocrt modelske zgrade.



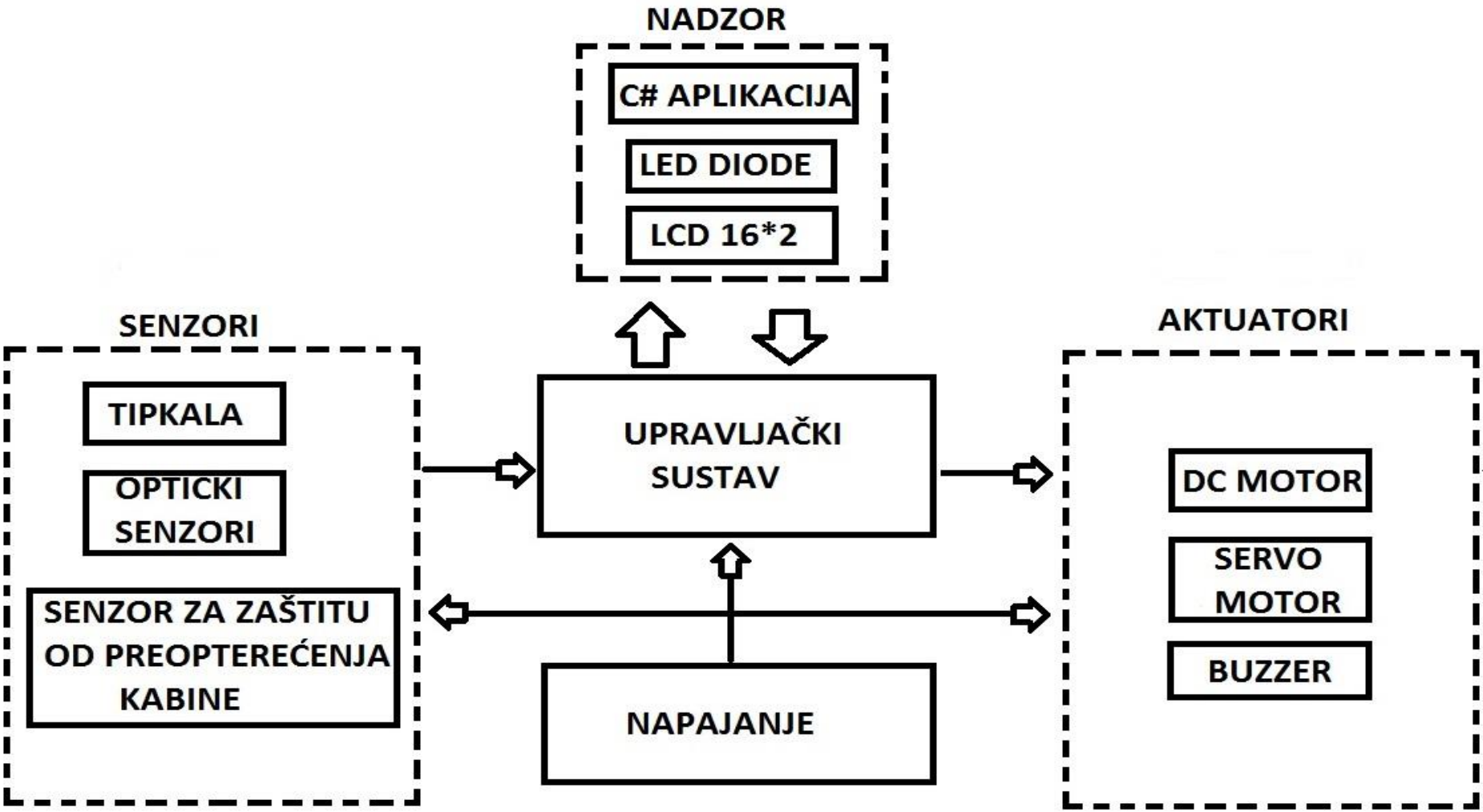
Slika 3.5. Nacrt, tlocrt i bokocrt kabine vertikalnog transportnog sustava.

3.2.2. Električni ustroj makete

Sve najvažnije elektroničke i elektromehaničke komponente transportnog sustava ugrađene su u gornjem dijelu makete. Tu se nalazi mikroupravljač, aktuator za pokretanje kabine i ispravljač za napajanje. Mikroupravljač i „srce“ ovog transportnog sustava je Arduino Mega 2560, aktuator je DC motor 12V, napajanje cijelog sustava dobiveno je iz AC/DC ispravljača koji daje 12V istosmjernog napona. Lcd zaslon 16x2 ugrađen je na vanjskom okviru i na njemu uvijek piše pozicija kabine te dali su vrata otvorena ili zatvorena. Za otvaranje i zatvaranje vrata zadužen je mali servo motor. Zaštita od preopterećenja kabine izvedena je korištenjem vage. Vaga je sastavljena od *Load Cell* plus operacijsko pojačalo INA125. Tipkalima se poziva kabina vertikalnog transportnog sustava na kat. Senzori koji zaustavljaju kabinu su optički senzori TCRT5000. Svjetlosna signalizacija su led diode 5mm crvene boje. Osim što se vrata sama zatvaraju i otvaraju prilikom dolaska na kat postoje dva tipkala s kojima je uvijek moguće otvoriti vrata osim kada je kabina u pokretu. Svi električki vodići s kojima je povezan cijeli električki ustroj makete nalaze se u plastičnim kanalicama radi boljeg estetskog izgleda.



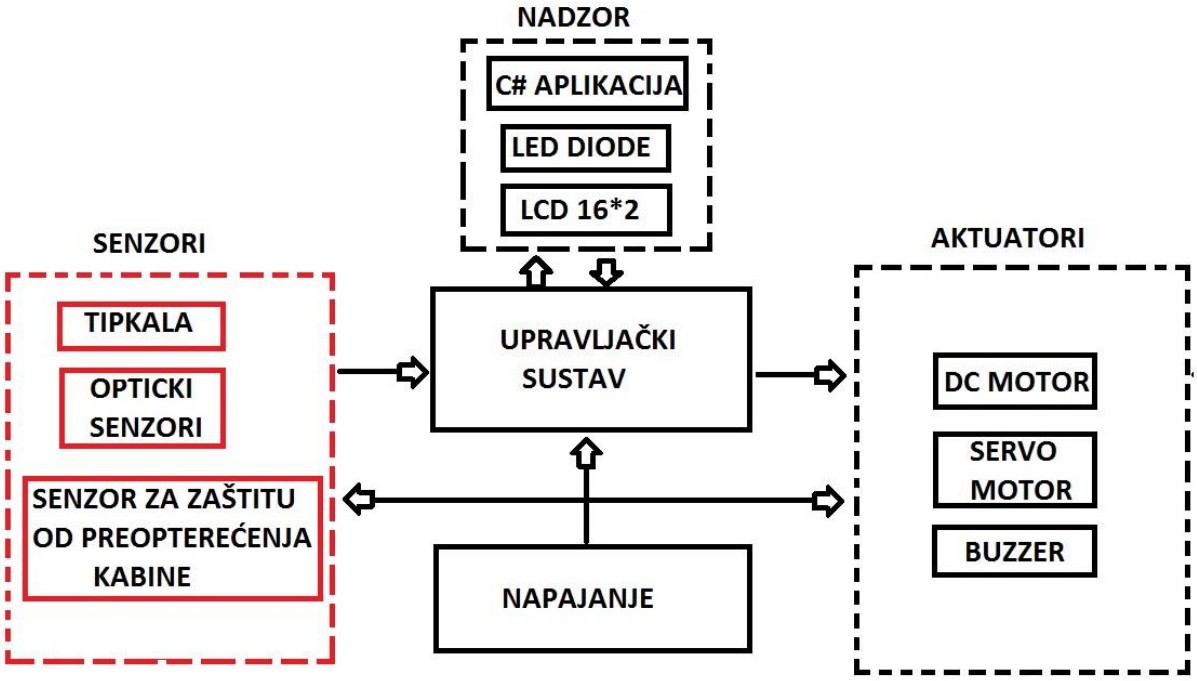
Slika 3.6. Idejno rješenje elektroničkog sustava.



Slika 3.7. Funkcionalni blok dijagram vertikalnog transportnog sustava.

3.3. Upravljački sustav

3.3.1. Senzori



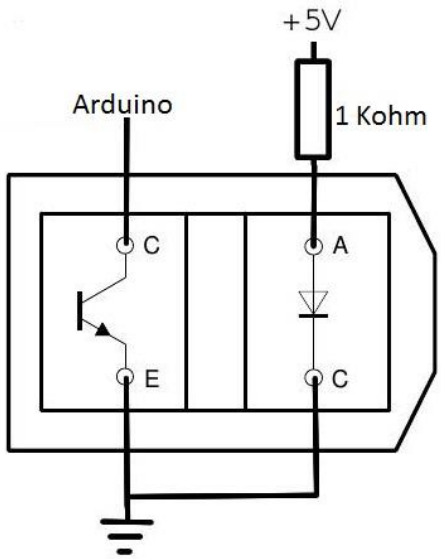
Senzor ili pretvornik je uređaj koji mjeri fizikalnu veličinu (npr. temperaturu, vlažnost zraka, tlak, broj okretaja motora) i pretvara je u signal pogodan za daljnju obradu (najčešće u električni signal). Senzori se dijele prema: načinu rada, složenosti, vrsti izlaznog signala i prema načinu prikaza signala. Za pojam senzorike možemo reći da se konstantno pojavljuje u klasičnoj primjeni u zadnjih pet godina. Automatizacijom proizvodnje čija se raširenost svakog dana sve više proširuje došli smo do spoznaje da je usporedni razvoj senzorike neophodan. Kao glavni primjer upotrebe senzorike možemo navesti upotrebu senzora u tvorničkim trakama u velikim, ali i malim postrojenjima. Senzori su najčešće postavljeni kako bi nadzirali, odnosno registrirali protok pogonske trake, kao i mnogobrojne čimbenike proizvodnje.

Optički senzori, njihov princip rada se zasniva na promjeni parametara optičkog signala sa promjenom fizičke veličine. Samim tim ovi senzori nemaju galvanske ili magnetske veze, već samo optičke. Kod optičkih senzora je postignuto: galvansko odvajanje, zaštita od šumova, mogućnost mjerenja fizičkih veličina, standardizacija izlaznog signala, visoka kvaliteta statičkih i dinamičkih karakteristika. Ovi senzori se mogu upotrijebiti u svim područjima djelovanja jakog magnetskog polja, visoke temperature, električnih šumova i

kemijske korozije, pa su mnogo fleksibilniji i pouzdaniji od klasičnih senzora. Loše osobine su: složenost izrade, obrada signala, zahtjevaju optičku vidljivost između predajnika i prijemnika, osjetljivost na mehaničke vibracije [6].

U ovom projektu je korišten optički senzor **TCRT5000**, točnije njih 9 komada. Dimenzije su mu 10.2 x 5.8 x 7 mm. Vrsta detektora: fototranzistor. Radna udaljenost mu je 0.2mm do 15mm. Tipična izlazna struja kod testiranja Ic = 1mA. Valna duljina emitera je 950nm. Kod ovog projekta se koriste za registriranje položaja kabine na pojedinom katu te za kontrolu brzine same kabine. Na arduino se spajaju preko njegovih digitalnih pinova (shema spajanja prikazana na slici 3.9.).

Slika 3.8. Optički senzor TCRT5000 [7].

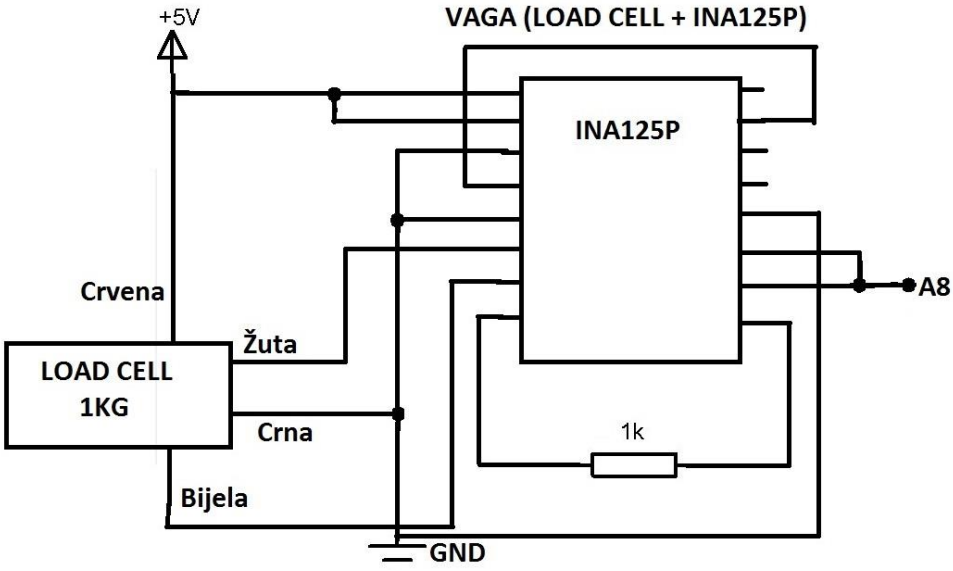


Slika 3.9. Shema spajanja senzora TCRT5000 na Arduino

Sustav za zaštitu od preopterećenja kabine transpotnog sustava

Load Cell je sonda koja se koristi za stvaranje električnog signala čija magnituda je izravno proporcionalna sili koja se mjeri. Na mikroupravljač se spaja preko operacijskog pojačala INA125. Shema spajanja prikazana je na slici 3.11. A8 je analogni pin na Arduinu.

Slika 3.10. *Load Cell* i operacijsko pojačalo INA125 [20][21].

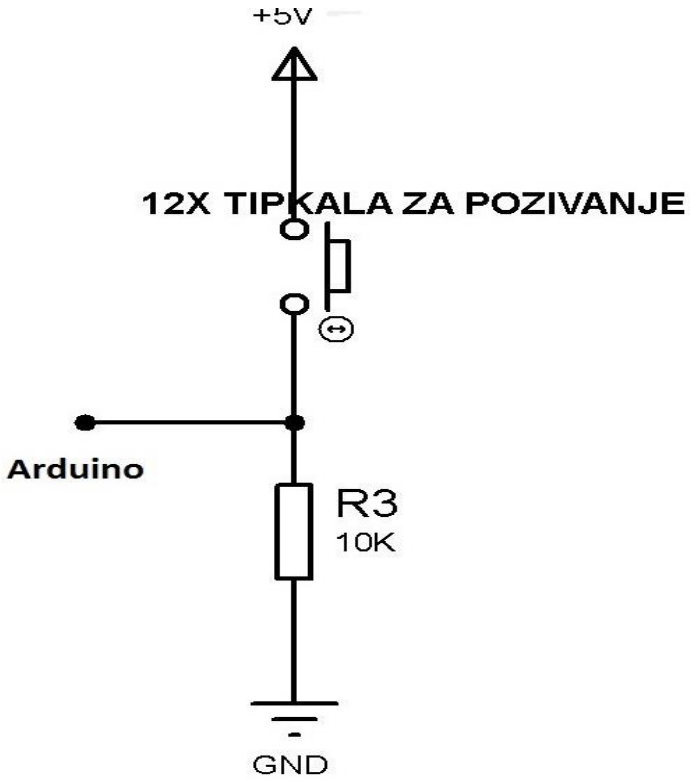


Slika 3.11. Shema spajanja sustava za zaštitu od preopterećenja kabine

Tipkala za pozivanje

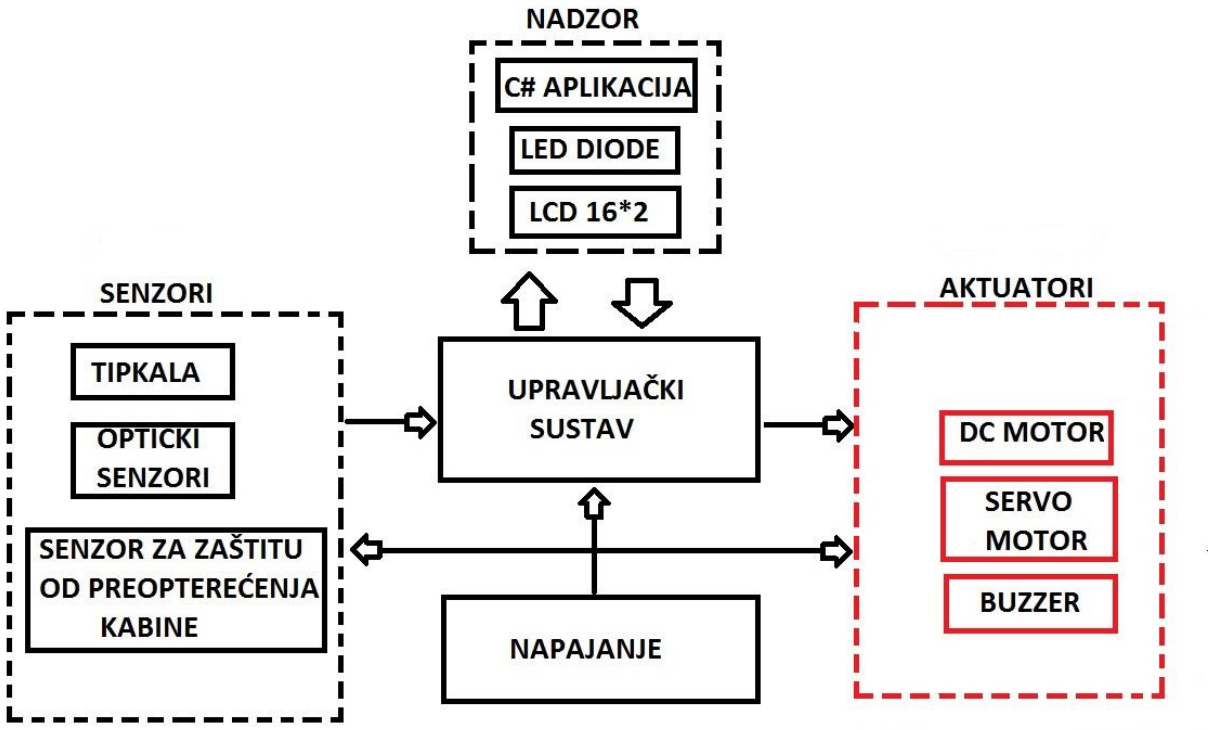
Tipkala za pozivanje se koriste kada se želi kabinu vertikalnog transportnog susutava pozvati na određeni kat. Postoje tipkala koja se nalaze unutar kabine i tipkala koja se nalaze izvan kabine tj. na modelu zgrade. Shema spajanja na Arduino prikazana je na slici 3.13. Za spajanje se koriste digitalni pinovi na Arduinu.

Slika 3.12. Tipkalo [22].



Slika 3.13. Shema spajanja tipkala na Arduino

3.3.2. Aktuatori



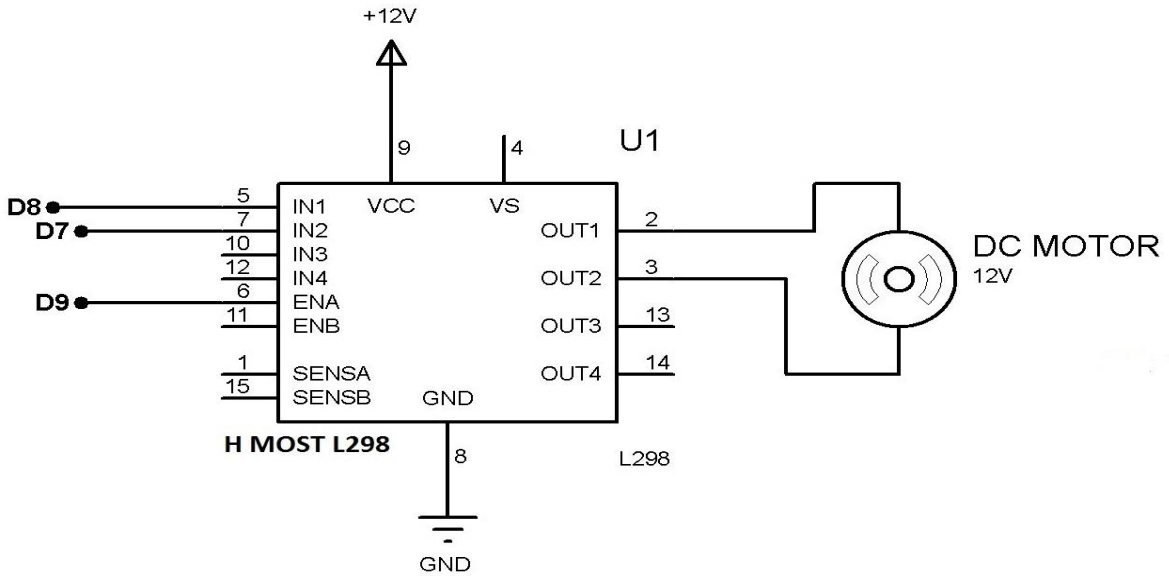
Aktuatori su uređaji koji pretvaraju električke ili fluidne ulaze u mehaničke izlaze, kao što su pozicija, sila ili moment. Razina izlazne energije je mnogo veća od razine energije ulaznog signala, tako da se zahtijeva korištenje električnog naboja, pneumatskog pritiska, hidrauličkog pritiska, idt. Klasifikacija i evaluacija najvažnijih aktuatorskih koncepata može se koncentriati u tri glavne grupe: elektromehanički aktuatori, aktuatori koji koriste snagu fluida i alternativni aktuatorski konepti (inteligentni, mikroaktuatori). Pod elektromehaničke aktuatore pripadaju DC motor, AC motor, koračni motor, servo motor, linearni motor... Aktuatori koji koriste snagu fluida mogu biti hidraulički (ventili, pumpe, motori) i pneumatski (regulacijski ventili, zasuni, motori). Mikroaktuatori mogu biti piezoelektrični, magnetostriktivni, elektrokemijski, termalni i memorijsko mentalni [8].

U ovom projektu je korišteno tri aktuatora, dva su elektromehanička dok je jedan mikroaktuator. Prvi elektromehanički je DC motor 12V koji se koristi za pokretanje kabine u voznom oknu transportnog sustava i koji se spaja na arduino preko H-mosta L298N, pomoću H-mosta je moguće upravljati brojem okretaja motora a time onda i brzinom kretanja kabine. Drugi elektromehanički aktuator je DC servo motor, on je montiran na kabini te služi za otvaranje i zatvaranje vrata kabine. Treći aktuator je mikroaktuator i to piezoelektrični buzzer koji se koristi za zvučnu signalizaciju kada je kabina transportnog sustava pozicionirana na

katu i još se koristi kao zvučno upozorenje ako dođe do preopterećenja same kabine transportnog sustava.

DC motor (slika 3.8.) je elektromehanički uređaj koji istosmjernu struju pretvara u rotacijsko gibanje. Klasični istosmjerni motor se sastoji od rotirajuće armature koja je oblikovana u obliku elektromagneta s dva pola i od statora kojega čine dva permanentna magneta. Krajevi namota armature spojeni su na rotacijski prekidač, komutator, koji prilikom svakog okretaja rotora dvaput mijenja smjer toka struje kroz armaturni namot stvarajući tako moment koji zakreće rotor. Shema spajanja DC motora na Arduino prikazana je na slici 3.15. D7, D8, D9 su digitalni pinovi na Arduinu.

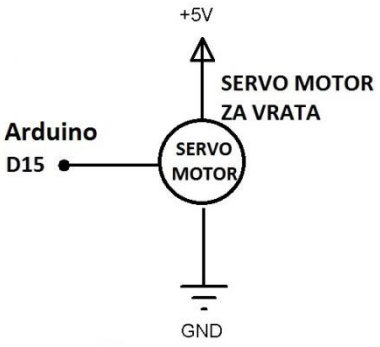
Slika 3.14. DC motor [9].



Slika 3.15. Shema spajanja DC motora na Arduino

Servo motor je tip električnog motora koji ima mogučnost precizne kontrole kutnog položaja, brzine i ubrzanja. Sastoji se od prikladnog motora na čiji je rotor spojen senzor (enkoder) za povratnu informaciju o položaju. Servo motori se dijele na istosmjerni servo motor (DC servo motor) i izmjenični servo motori (AC servo motor). Servo motori za razliku od stepper motora imaju konstantni okretni moment, a zbog sustava zatvorene petlje nema bojazni od gubitka koraka.

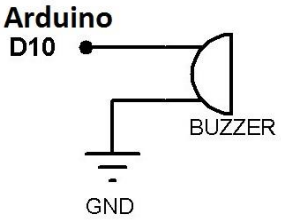
Slika 3.16. DC servo motor (isti ovakav korišten u projektu) [10].



Slika 3.17. Shema spajanja servo motora na Arduino

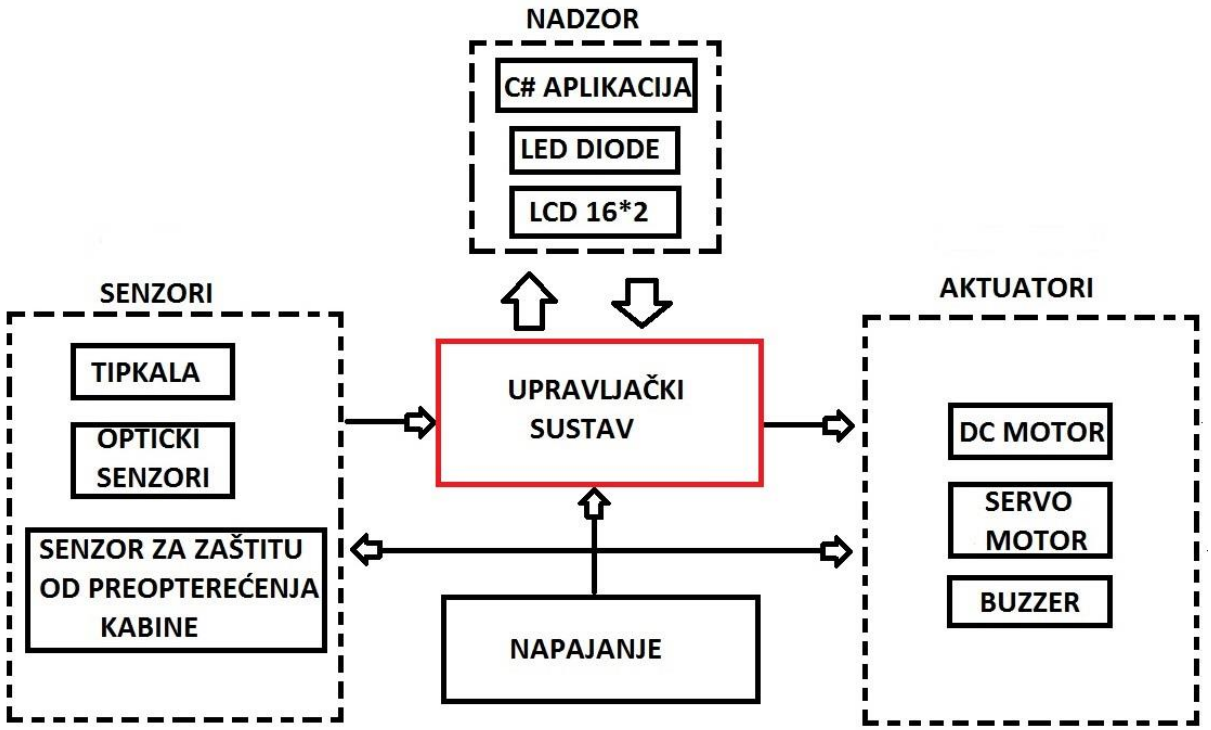
***Buzzer* ili zujalica** je audio signalni uređaj koji može biti mehanički, elektromehanički ili piezoelektični. Namjena mu je proizvoditi zvukove kod nekih upozorenja, alarma, ili potvrdu korisničkog unosa recimo kod pritiska na računalni miš. Shema spajanja na Arduino prikazana je na slici 3.19. D10 je digitalni pin na Arduinu.

Slika 3.18. *Buzzer* (zujalica) [11].



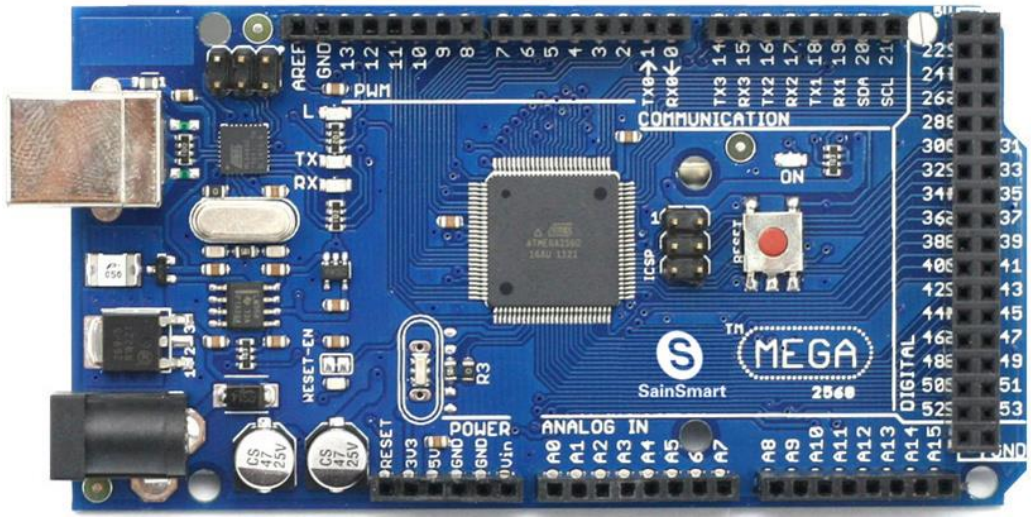
Slika 3.19. Shema spajanja *buzzera* na Arduino

3.3.3. Mikroupravljač



Mikroupravljač je mikroračunalo zaduženo za upravljanje i kontrolu određenih procesa. Mikroupravljači imaju vrlo rasprostranjenu primjenu u različitm uređajima i sustavima. Arhitektura mikroupravljača izravno je zavisna o njegovoj primjeni. Pored procesora, memorija, ulaza i izlaza, tu pripadaju i posebne izvedbe U/I sklopova, pojačala, pretvarača i sl. Posebno područje su integrirani mikroupravljači (svi sklopovi standardizirani i izvedeni u jednom čipu). Mikroupravljač je integrirani sklop koji sadrži sve navedene elemente te predstavlja osnovu pri dizajniranju cjelovitog sustava za nadzor i upravljanje.

Arduino Mega 2560 je široko primjenjiv mikroupravljač brojnih mogućnosti s posbenim naglaskom na jednostavnost njegove primjene i veliku programsku podršku za izvedbu različitih projektnih rješenja. Arduino je univerzalni mikroupravljač zasnovan na Atmel tehnologiji te je idealan za razvoj upravljačke elektronike i robotike. Sustav je otvorenog koda temeljen na jednostavnoj razvojnoj pločici s ulazno/ izlaznim konektorima te besplatnom programskom podrškom s jednostavnim korisničkim sučeljem. Programiranje uređaja se izvodi iz integriranog razvojnog okruženja koje postoji za Windows, Mac i Linux operativni sustav u programskom jeziku sličnom C-u. Za proširenje Arduina dostupni su brojni *shildovi* koji imaju dodatne mogućnosti ili čak svoje vlastite mikroupravljače.



Slika 3.20. Arduino Mega 2560 [12].

Mikrokontroler	Atmega2560
Radni napon	5 V
Ulazni napon (preporučeni)	7 V – 12 V
Ulazni napon (maksimalni)	20 V
Digitalni U/I pinovi	54 (15 pwm)
Analogni ulazni pinovi	16
DC struja po U/I pinu	20 mA
DC struja za 3.3V pin	50 mA
Flash memorija	256 KB (8 KB bootloader)
SRAM	8 KB
EEPROM	4 KB
Radni takt	16 MHz
Dužina	101.52 mm
Širina	53.3 mm
Težina	37 g

Tablica 3.1. Tehničke specifikacije Arduino Mege 2560 [13]

Analogni ulazi

Arduino sadrži 16 kanalni 10-bitni AD pretvornik. Korištenjem funkcije *analogRead(),* ulazni napon podijeli se između 0 i 5V u integer vrijednosti između 0 i 1023. Funkcija *analogRead()* prima broj pina s kojeg se želi čitati te vraća integer vrijednosti između 0 i 1023. Dobiva se korak kvantizacije od: 5 V/ 1024 = 0.0049 V (4.9 mV). Za čitanje s analognog ulaza potrebno je 100 µs, stoga je najviša brzina čitanja 10000 puta u sekundi.

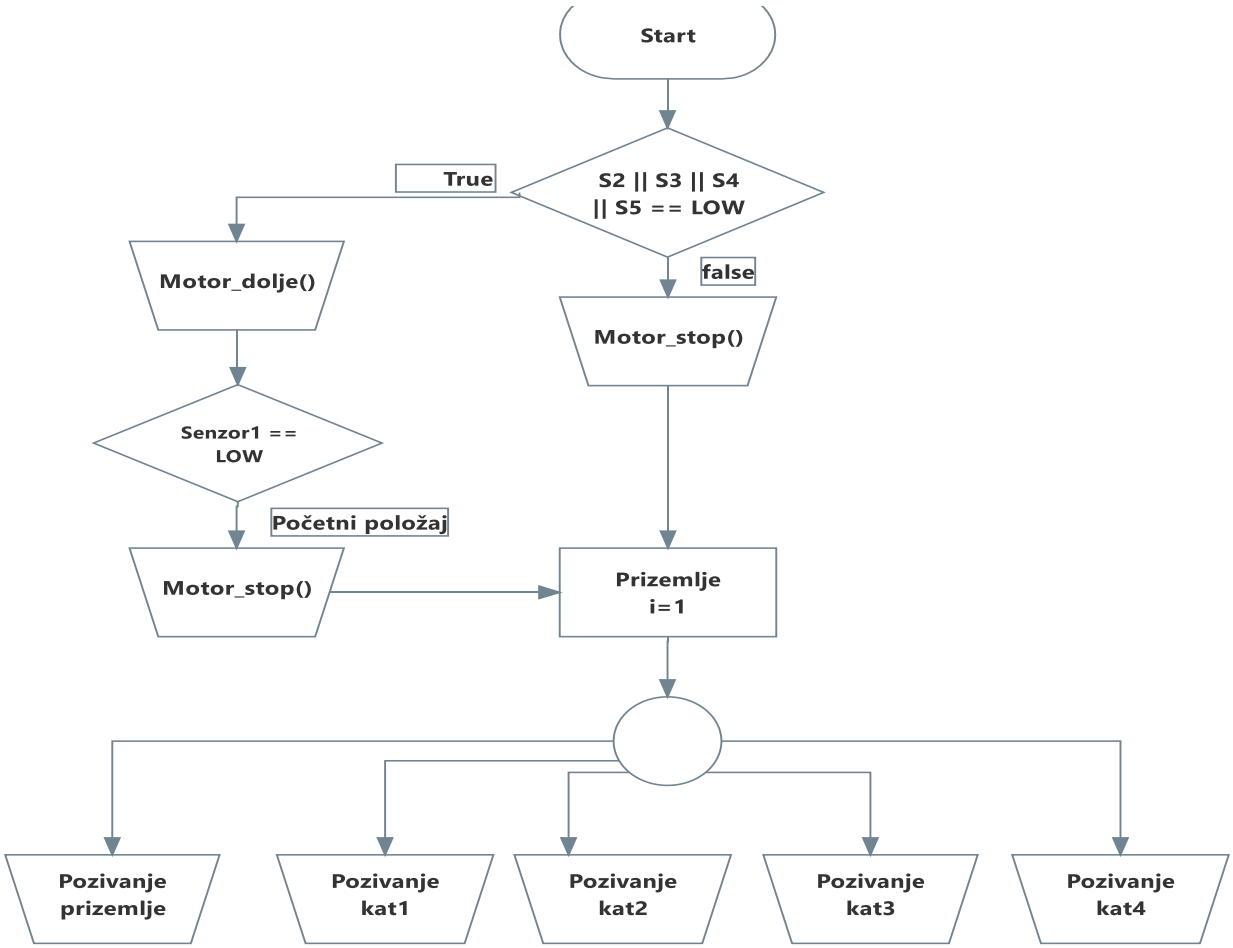
Digitalni ulazi/ izlazi

Funkcija *digitalRead()* čita vrijednosti sa određenog pina i vraća *HIGH* ili *LOW*, dok funkcija *digitalWrite()* upisuje vrijednost *HIGH* ili *LOW* na odabrani pin. Funkcija *analogWrite()* daje analognu vrijednost (PWM signal) na PWM digitalne pinove. Koristiti se za promjenu brzine motora ili za promjenu jačine osvjetljenja LED diode.

3.4. Programsko rješenje

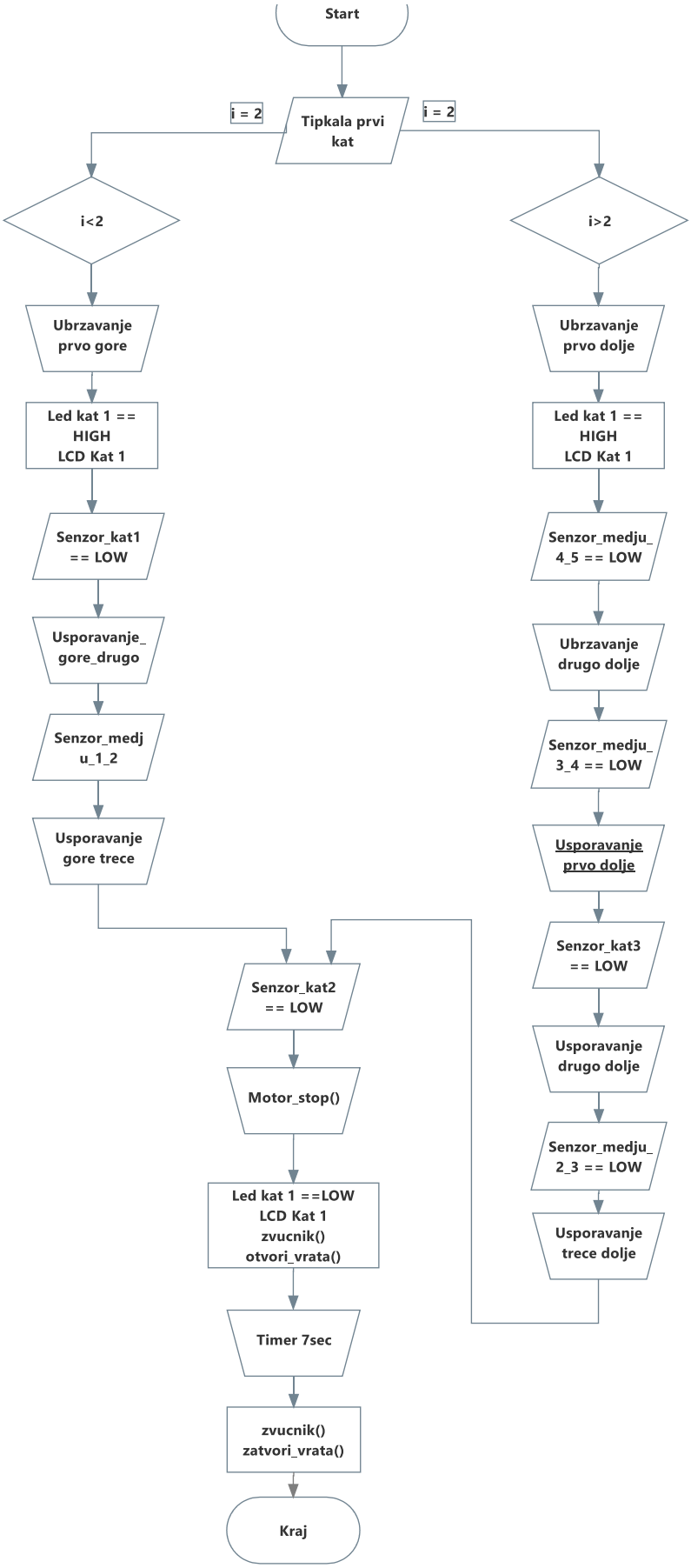
3.4.1. Algoritam upravljanja

U matematici, računarstvu, lingvistici i srodnim disciplinama, **algoritam ili postupnik** je konačan slijed dobro definiranih naredbi za ostvarenje zadatka, koji će za dano početno stanje terminirati u definiranom konačnom stanju. Algoritmi imaju slijedeća svojstva: diskretnost (u odvojenim koracima izvode se diskretne operacije algoritma koje vode ka konačnom cilju), konačnost (označava sposobnost algoritma da nakon konačnog broja koraka daje izlazne podatke odnosno rezultate), determiniranost (za iste ulazne podatke algoritam uvijek daje iste rezultate), masovnost (algoritam je primjenjiv na veći broj ulaznih vrijednosti). Moderno računarstvo je nezamislivo bez primjene algoritama, njihove matematičke analize te postupcima ubrzavanja njihova izvođenja (optimiranje, optimiziranje). Sva su ta područja povezana i međusobno se nadopunjuju [14].



Slika 3.21. Glavni algoritam upravljanja vertikalnim transportnim sustavom

Na slici 3.21. je prikazan glavni algoritam upravljanja vertikalnim transportnim sustavom, vidljivo je da ako se kabina dizala nalazi na nekom od katova a da nije u prizemlju da se onda ona automatski spušta prema dolje do prizemlja i tako sustav postavlja u početni položaj. Kad je sustav u početnom položaju može se pozivati željeni kat. U glavnom algoritmu upravljanja nije prikazan cijeli algoritam za pozivanje kabine na neki od katova već je na slici 3.16. prikazano pozivanje kabine na 1. kat vertikalnog sustava. Kao što je to prikazano za 1. kat takvi su slični algoritmi i za pozivanje na ostale katove. S2, S3, S4 i S5 su optički senzori koji zaustavljaju kabinu sustava kad dolazi na određeni kat. Isprogramirani su tako da reagiraju kad je *LOW* stanje na njima. Funkcija *motor_stop()* trenutno zaustavlja vartikalno gibanje kabine sustava. Funkcija *motor_dolje()* pokreće kabinu vertikalnog transpotnog sustava prema dolje, tj. s viših katova na niže dok funkcija *motor_gore()* radi suprotno.



Slika 3.22. Algoritam pozivanja kabine vertikalnog sustava na 1. kat

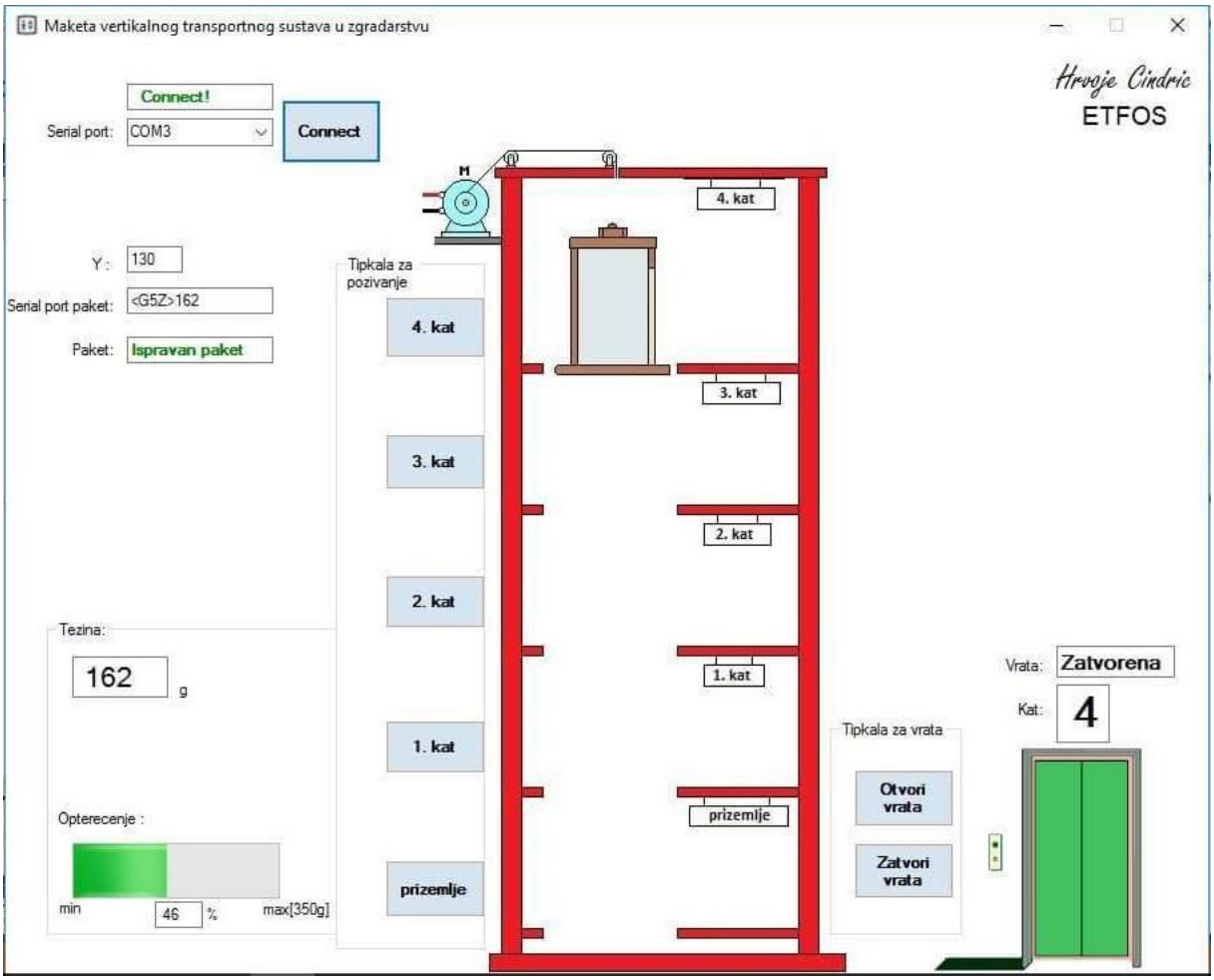
Dio koda za pozivanje kabine vertikalnog sustava na 1.kat s višeg kata (npr. 4.kata)

```
if(stanje_provjera_vrata == HIGH && kat2_vani_status == HIGH || kat2_unutra_status ==
HIGH && stanje_provjera_vrata == HIGH || c=='2' && stanje_provjera_vrata == HIGH)
    {                                          // Pritisak na tipkala za pozivanje kabine
     if(i>2){
     while(i>2) {
     i--;
    ubrzavanje_prvo_dolje();
    digitalWrite(led_kat2_vani, HIGH); //Uključivanje svjetlosne signalizacije na katu.
    digitalWrite(led_kat2_unutra, HIGH);
    lcd.clear();                               // Prikaz  kata na lcd zaslonu
    lcd.setCursor(2, 0); lcd.print(char(1));  lcd.setCursor(4, 0);  lcd.print("1 kat");
    kretanje = 'D';  kat = '2'; vrata = 'Z';
    }
    }
  // Senzori među katovima za ubrzavanje i usporavanje kabine
    if(i==2 && senzor_medju_4_5_status == LOW){
     ubrzavanje_drugo_dolje();
    }
    if(i==2 && senzor_medju_3_4_status == LOW){
     usporavanje_dolje_prvo();
    }
    if(i==2 && senzor_kat3_status == LOW){
     usporavanje_dolje_drugo();
    }
     if(i==2 && senzor_medju_2_3_status == LOW){
     usporavanje_dolje_trece();
    }
  // Zaustavljanje kabine na 1. katu
    if( senzor_kat2_status != zadnje_stanje_kat2){
     if (i== 2 && senzor_kat2_status == LOW){
     i=2;
     motor_stop();
    digitalWrite(led_kat2_vani, LOW);
```

```
  digitalWrite(led_kat2_unutra, LOW);
  lcd.clear();   lcd.setCursor(2, 0);  lcd.print(char(3));
lcd.setCursor(4, 0);  lcd.print("1 kat");
zvucnik();                              // Zvucna signalizacija
otvoren_lift();                         // Otvaranje vrata kabine
poc_vrijeme_dva = millis();
  }
  }
zadnje_stanje_kat2 = senzor_kat2_status;
```

Pritiskom tipkala za pozivanje, bilo ono unutar kabine ili izvan kabine počinje procedura pozivanja kabine na kat. Procedura se sastoji (npr. za pozivanje kabine s 4. kata na 1. kat) kada se pritisne tipkalo za 1. kat uključe se led diode koje signaliziraju na koji je kat pozvana kabina, na lcdu je ispisan kat na koji kabina kabina treba doci i strelica prema dolje. Motor se uključi i počinje spuštati kabinu polako prema dolje (funkcija *ubrzavanje_prvo_dolje()*), prilikom dolaska do senzora koji se nalazi između 3. i 4. kata motor ubrzava i kabina se vertikalno giba svojom maksimalnom brzinom (funkcija *ubrzavanje_drugo_dolje()*). Ubrzavanje i usporavanje, tj. upravljanje brzinom DC motora je riješeno putem PWM + H most. Prilikom prolaska kabine pored senzora koji se nalazi između 2. i 3. kata počinje prvo kočenje kabine i brzina se smanjuje (funkcija *usporavanje_dolje_prvo()*), na idućem senzoru, tj. senzoru 2. kata počinje drugo uporavanje gdje se brzina još više smanjuje (funkcija *usporavanje_dolje_drugo()*), na idućem senzoru koji se nalazi između 1. i 2. kata počinje treće usporavanje i brzina se još više smanjuje i onda polako kabina dolazi do senzora na 1. katu gdje se zaustavlja. Kad se kabina zaustavi, upali se zujalica (buzzer) u trajanju 3 sekunde i nakon toga se vrata na kabini otvore i timer počinje odbrojavati, kada on dođe do 7 sekundi ponovno se pali zujalica na 3 sekunda i vrata se zatvaraju. Vrata je moguće otvoriti i zatvoriti u svakom trenutku kada je kabina na nekom od katova pritiskom tipkala za otvaranje odnosno zatvaranje vrata. Ovdje je opisana procedura pozivanja kabine s 4. kata na 1. kat, za sve ostale kombinacije je sličan postupak. Također je rad vertikalnog sustava moguće promatrati putem aplikacije za udaljeni nadzor, opisanoj u idućem poglavlju.

Sustav za udaljeni nadzor i kontrolu transportnog sustava (slika 3.23) ili SCADA sustav (eng. Supervisory Control And Data Acquisition) predstavlja računalni sustav za nadzor, mjerenje, i upravljanje industrijskim sustavima. Svaki industrijski proces kojeg ima smisla automatizirati je odličan kandidat za primjenu SCADA sustava. Ovi sustavi u različitim obilicima postoje još od 60-tih godina, a od 90-tih godina 20. stoljeća doživljavaju ekspanziju sa pohavom sve bržih i efikasnijih računalnih i mikrokontrolerskih uređaja. Mogu se koristiti za jednostavn nadzor (npr. Temperature, vlažnosti zraka, tlaka), kao i za kompleksan nadzor i upravljanje (npr. Proizvodni procesi u tvornicama ili regulacija željezničkog i cestovnog prometa) [15].



Slika 3.23. Sustav za udaljeni nadzor i kontrolu vertikalnog transportnog sustava.

Za početak treba reći da ova aplikacija za udaljeni nadzor i kontrolu je sustav za rad u stvarnom vremenu (**Real-Time System**). Sustavi za rad u stvarnom vremenu imaju strogo definirana vremenska ograničenja u kojima moraju djelovati. Dijele se na hard real time

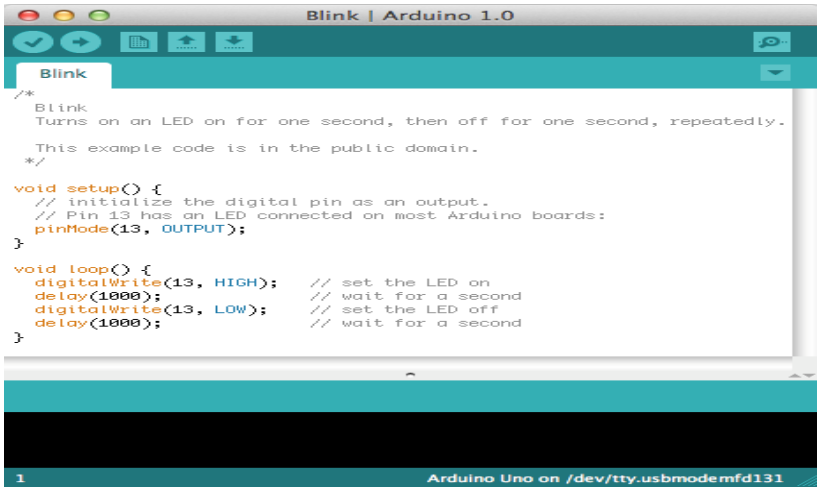
sustave i na soft real time sustave. Kod hard real time sustava se garanitra da će zadani zadatak biti izvršen u točno definiranim vremenskim ograničenjima (primjer je zrakoplov). Kod soft real time sustava poslovi se obavljaju prema redoslijedu koji je određen prema prioritetima. Ako neki zadatak ima veći prioritet od drugih onda će se sustav „pozabaviti“ njegovim rješavanjem. Ovi sustavi nisu pogodni u robotici i za industrijsku kontrolu.

Na sustavu za udaljeni nadzor transportnog sustava prikazanog na slici 3.23. moguće je u svakom trenutku pratiti kretanje kabine sustava, kretanje kabine je zapravo animacija kretanja slike po y osi. Za svaki kat se zna koliko točno iznosi y, te ako kabina ide npr. prema 4. katu, kad je y = 130 kabina, tj. slika kabine se zaustavlja, isto vrijedi i za ostale katove. Osim položaja kabine može se vidjeti i dali su vrata otvorena ili zatvorena. To je riješeno na način da imaju dvije slike i onda ovisno o stanju vrata one se izmjenjuju. U kabini transportnog sustva ugrađena je elektronika koja služi kao zaštitni mehanizam od preopterečenja kabine, riječ je o klasičnoj vagi koja podatke također šalje u ovaj sustav za nadzor te je kontroloru u svakom trenutku vidljiva težina kabine i dali je ona u dozvoljenim granicama. U slučaju da ona nadmaši maksimalnu granicu pali se alarm na samom transportnom sustavu a i u aplikaciji je vidljivo da je opterečenje kabine preveliko. Dok god je prevelika težina, na kabini se ne mogu zatvoriti vrata te ona nemože napustiti kat na kojem se nalazi. Osim samog nadzora stanja vertikalnog transportnog sustava pomoću aplikacije je moguće i upravljati sustavom pomuću tipkala (buttona) koji se nalaze u samoj aplikaciji. Moguće je klikom računalnog miša pozivati kabinu na svaki od katova, također je moguće i otvarati i zatvarati vrata. Aplikacija radi na način da s mikroupravljača preko serijske veze prima pakete, u tom paketu se nalazi pozicija objekta tj. dali se kabina giba prema dolje ili prema gore, zatim se nalazi broj kata prema kojem kabina ide, treća stvar je stanje vrata, dali su zatvorena ili otvorena. Na kraju paketa je vrijednost koja predstavlja težinu kabine. Paket stiže u komadu a onda ga se isčitava (parsira) dio po dio i po tome aplikacija radi. O komunikaciji između mikroupravljača i aplikacije za udaljeni nadzor i kontrolu više u poglavlju **integracija sustava.**

3.4.2. Razvojni sustavi

Razvojni sustavi koji se koriste u ovom projektu su Arduino i programski jezik CSharp (C#).

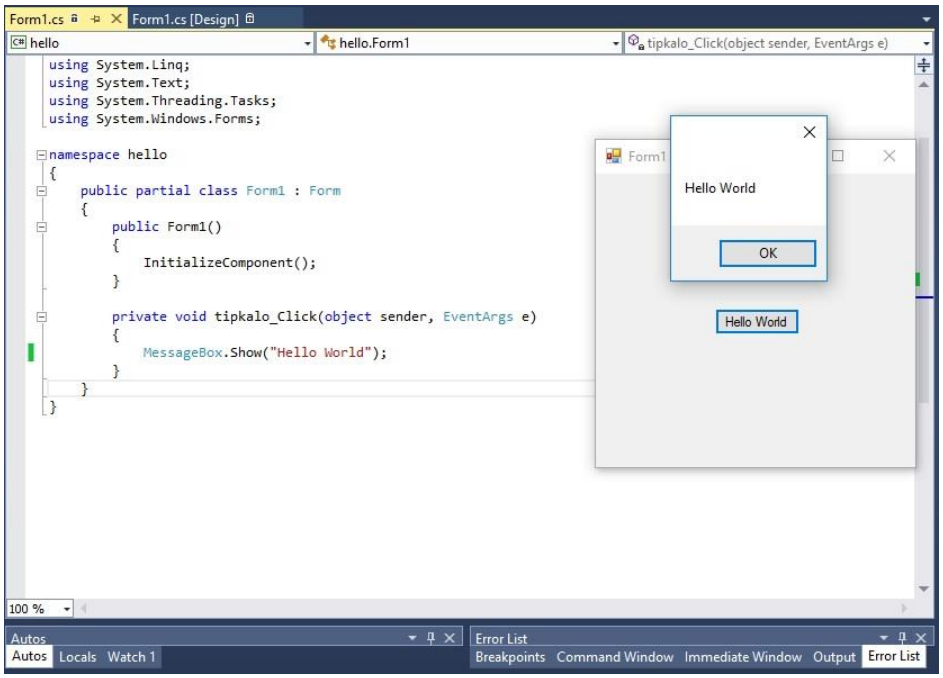
Arduino je open-source platforma za kreiranje elektroničkih prototipova bazirana na sklopovlju i programskom paketu koji je fleksibilan i jednostavan za korištenje. Arduino je namijenjen automatičarima, elektroničarima, umjetnicima, dizajnerima, hobistima i svima koji su zainteresirani za kreiranje objekata ili okruženja. Arduino platforma je skup elektroničkih i softverskih komponenti koje se mogu jednostavno povezivati u složenije cjeline s ciljem izrade zabavnih i poučnih elektroničkih sklopova. Srce arduina su mikrokontroleri. Mikrokontroler je malo računalo sadržano na jednom integriranom sklopu. Arduino okruženje najčešće koristi 8 bitne mikrokontrolere koje proizvodi tvrtka ATMEL. Najrasprostranjeniji model je ATMEGA328P koji se korisiti na osnovnoj Arduino prototipnoj pločici. Kako je Arduino open-source platforma dozvoljeno je njezino dijeljenje i preuređivanje u svrhu kreiranja novih platformi koje su međusobno kompatibilne tako da su razvojem nastale još mnoge inačice razvojnih okruženja baziranih na Arduino platformi. U osnovi, sve Arduino kompatibilne pločice sastoje se od mikrokontrolera, integriranog sklopa za komunikaciju s računalom, te perifernih elektroničkih dijelova za osiguravanja mogučnosti rada mikrokontrolera kao što su stabilizatori napona, kvarcni oscilator za generiranje frkvencija takta i slično. Za pisanje programa za Arduino mikrokontrolere koristi se Arduino programsko okruženje koje je moguće besplatno preuzeti na Internetu. Kako bi se mogli napisani programi prebaciti u mikrokontroler na razvojnoj pločici koristi se USB veza s računalom [16].



Slika 3.24. Arduino IDE programsko okruženje [17].

CSharp (C#) je nastao u tvrtki Microsoft i razvijen je od strane tima stručnjaka predvođenih sa Anders Hejlsberg-om i Scott Wiltamuth-om. Na tržištu se pojavio 2000. godine zajedno sa .NET platformom. C# je nastao s ciljem da bude jednostavan, siguran, moderan, objektno orijentiran jezik visokih performansi za .NET platformu. C# je nastao na temelju objektnih jezika Java, C++ i Visual Basic. Vrlo je sličan Javi i C++ jeziku (sintaksa i semantika je dobrim dijelom preuzeta iz Jave, koja je kao i C# potpuno objektno orijentirani jezik). Ali C# za razliku od Jave nije neovisan o platformi, tj. operativnom sustavu, već je kreiran za izradu stolnih (desktop) i Internet aplikacija u .Microsoft .NET okruženju. C# sadrži sve dobre odlike poptuno objektno programskog jezika (koje većinom preuzima iz C++ i Jave), a u sklopu .NET platforme omogućava kreiranje vizualnih aplikacija čak i onim korisnicima koji nemaju programerskog iskustva. Također daje dobar uvid u način na koji nastaju objektne i vizualne aplikacije, i vrlo je lako naučiti korisnike kako projektirati takve aplikacije i upravljati njihovim korištenjem. C# sadrži samo oko 80 ključnih riječi i na desetke ugrađenih tipova podataka. C# ima velike mogućnosti u definiranju klasa (tipova objekata), novih metoda i svojstava, te korištenju enkapsulacije, nasljeđivanje i polimorfizma kao što je to omogućeno u C++ i Javi. Također podržava rad s pokazivačima i „garbage collection“. C# koristi postupak zvan garbage collection za oslobađanje memorije koju zauzimaju objekti koji više nisu dostupni programu. Programer je oslobođen brige o tome koji mu objekti više ne trebaju, jer to umjesto njega radi sustav. Ako je objekt postojao i bio korišten neko vrijeme, može postojati nekoliko poziva na njega. Objekt postaje *garbage* tek nakon što nestanu svi pozivi na njega.

Alat za programiranje C# aplikacija je **Microsoft Visual Studio**. Microsoft Visual Studio je integrirano razvojno okruženje (ili IDE) kojeg pravi Microsoft. Koristi se za razvoj računalnih programa za Windows, web stranice, aplikacije i usluge. Visual Studio podržava različite programske jezike. Ugrađeni programski jezici su C, C++ i C++/CLI (preko Visual C++), VB.Net (preko Visual Basic .NET)-a, C# (preko Visual C#) i F# (počevši od programa Visual Studio 2010). Podrška za ostale programske jezike poput M, Python i Ruby-ja kao i ostalih je dostupan instalacijom jezičkih servisa koji se mogu zasebno instalirati. Također podržava XML/XSLT, HTML/XHTML, JavaScript i CSS [18].



Slika 3.25. Primjer C# programa („Hello World“) u Microsoft Visual Studio razvojnom okruženju.

3.5. Integracija sustava

Integracija zapravo predstavlja spajanje, ujedinjavanje ili povezivanje nekih dijelova (elemenata) u cjelinu koja harmonično djeluje u postojanju zajedničkih ciljeva, odnosno koja sadrži sklad između pojedinačnih i zajedničkih ciljeva

Upravo u ovom projektu transportnog sustava integracija je spajanje hardverskih i softverskih dijelova sustava. Potrebno je spojiti program iz mikroupravljača sa sustavom za udaljeni nadzor. Komunikacija se odvija putem *serijske USB veze*. Usb (univerzalna serijska sabirnica) je tehnološko rješenje za komunikaciju računala s vanjskim uređajima pri čemu se podaci razmjenjuju serijski relativno velikom brzinom. USB je zamijenio razna dotadašnja serijska i paralelna sučenja na računalima. Cilj tehnologije USB jest rasterećivanje glavne sabirnice računala od posebnih kartica za proširenje, kao i olakšavanje umetanja i odvajanja vanjskih uređaja bez potrebe za ponovno pokretanja računala.

Dio koda u programskom jeziku C# koji omogućava komunikaciju mikroupravljača s računalom putem serijske USB veze.

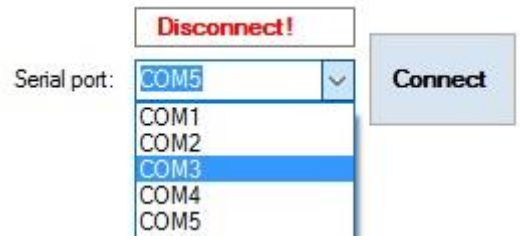
```
using System.IO.Ports;
namespace dizalo
{
   public partial class dizalo : Form
   {
private class Item
      {
         // definiranje varijabli da se mogu popunjavati Comboboxevi
         public string Name;
         public int Value;
         public Item(string name, int value)
         {
            Name = name; Value = value;

         }
         public override string ToString()
         {
            return Name;
         }

      }
public dizalo()
      {
         combo_serial.Items.Add(new Item("COM1", 1));
         combo_serial.Items.Add(new Item("COM2", 2));
         combo_serial.Items.Add(new Item("COM3", 3));
         combo_serial.Items.Add(new Item("COM4", 4));
         combo_serial.Items.Add(new Item("COM5", 5));
         }
         // Tipkalo za odabir serijskog porta na koji se treba spojiti.
private void odaberi_port_Click(object sender, EventArgs e)
      {
```

```
    serialPort1.PortName = combo_serial.SelectedItem.ToString();
    serialPort1.BaudRate = 9600;
    serialPort1.DtrEnable = true;
// Provjera dali je uspješna komunikacija između mikroupravljača i programa
try
{
if (!serialPort1.IsOpen)
{
    serialPort1.DataReceived += serialPort1_DataReceived;
    serialPort1.Open();
    connect_textBox.Text = "  Connect!";
    connect_textBox.ForeColor = Color.Green;
}
}
catch (Exception ex)
{
    connect_textBox.Text = "  Disconnect!";
    connect_textBox.ForeColor = Color.Red;
}

        }
}
}
```



Slika 3.26. Meni za odabir serijskog porta u programskom jeziku C#

4. ISPITIVANJE I REZULTATI

4.1. Ispitne metode i evaluacijski parametri

Funckionalno testiranje

Funkcionalno testiranje ima za cilj procjenu da li je promatrano ponašanje testiranog softvera u skladu sa svojim specifikacijama. Uobičajeno, programeri testiraju da li su zadovoljeni zahtjevi. Ako određeni zahtjev može biti izvršen i dobijen je zadovoljavajući rezultat, test slučaj je u redu. Ako su test slučajevi odabrani tako da je svakom zahtjevu dodjeljen bar jedan test slučaj, program se smatra testiranim i trebalo bi da radi kada svi test slučajevi pokažu zadovoljavajuće rezultate. Međutim, i kod ovog načina testiranja postoje slabosti. Tko provjerava zahtjeve? Često se ne pravi usporedba između konačnih specifikacija i specifikacija dizajna. Ovaj problem potiče od ljudi koji postavljaju zahtjeve. Često ne mogu da vizualiziraju ponašanje softvera koji će na kraju biti napravljen. Na kraju krajeva, finalni proizvod može sadržavati grešku koja se već nalazila u zahtjevima. A ljudi koji su postavljali zahtjeve imali su drugačiju ideju kako bi to izgledalo i funkcioniralo.

Testiranje pouzdanosti elektroničkog sklopa

Ovdje je provedeno ispitivanje pouzdanosti elektroničkog sklopa vertikalnog transportnog sustava. Ispitavanje, to jest određivanje pouzdanosti elektronike izvedeno je u programskom paketu RELEX. RELEX je potpuno integrirani alat za predviđanje pouzdanosti i analizu električke, elektroničke, mehaničke, i elektromehaničke opreme. RELEX također pruža mogućnosti kao što su blok dijagram pouzdanosti, optimizacija i simulacija sustava, Weibullova analiza, FMEA, FMECA, analiza pomoću stabla događaja, Markovljeva analiza, predviđanje održavljivosti, analiza troškova životnog ciklusa.

Predviđanje pouzdanosti je jedno od najčešćih oblika analize pouzdanosti. Predviđanje pouzdanosti je analiza dijelova i komponenti u svrhu predviđanja brzine kojom se uređaj kvari. Ona se obično temelji na utvrđenom modelu. Modeli pružaju proceduru za izračun inteziteta kvarova komponenti. **Intezitet kvarova** se računa na temelju prikupljenih informacija u vezi komponente (kao što su podaci o opterećenju, informacije o kvaliteti, temperaturi i podaci okoliša) prema standardnim jednadžbama. Dobiveni intenzitet kvara može se koristiti u analizi pouzdanosti. Intenzitet kvara (engl. failure rate) definira se kao brzina pojavljivanja kvarova. Vrijednost se obično izražava kao broj kvarova na milijun sati

(FPMH). Intenzitet kvara je zapravo predviđeni broj kvarova koji će se pojaviti u određenom vremenu. Prilikom izračuna koriste se uspostavljeni modeli predviđanja pouzdanosti. Izračuni se obično temelje na podacima o komponenti kao što su temperatura, okoliš i opterećenje. Intenzitet kvara za sastavljene uređaje dobije se zbrajanjem intenziteta kvarova pojedinih komponenti u uređaju. **MTBF** označava srednje vrijeme između kvarova (engl. mean time between failures). MTBF jest recipročna vrijednost za konstantni intenzitet kvara sustava. Jednostavnije rečeno, MTBF se može opisati kao broj sati koji treba proći do kvara na komponenti, sastavljenom uređaju ili sustavu. Vrlo važno je shvatiti da MTBF nije točna vrijednost, već očekivana vrijednost do vremena kvara. Predviđanja pouzdanosti temelji se na modelima pouzdanosti. Za elektroničke komponente najraširenији modeli su MIL-HDBK-217 i Telcordia. **MIL-HDBK-217** je model ministarstva obrane SAD za predviđanje pouzdanosti. Namijenjen je matematičkom određivanju pouzdanosti za bilo koji tip elektroničkog uređaja. Koristi se u vojnoj i komercijalnoj industriji [19].

4.2. Rezultati ispitivanja

Rezultati testiranja funkcionalnosti

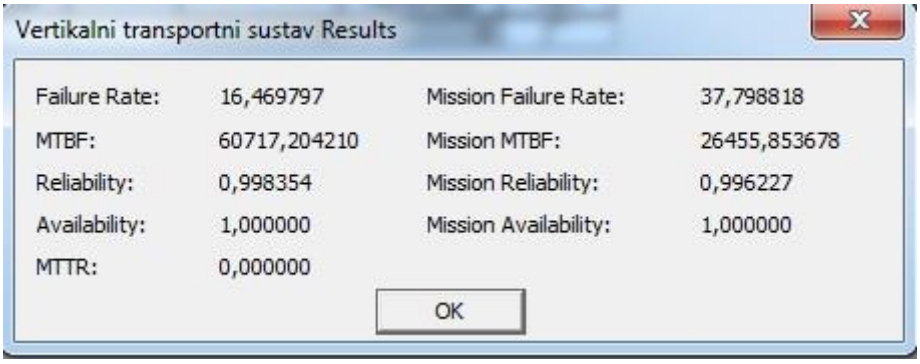
Redni broj	Testni slučaj	Uspješno izvršenje zahtjeva	Napomena
1.	Pozivanje kabine u prizemlje	DA	Testni slučaj u potpunosti ispravno izvršen
2.	Pozivanje kabine na 1. kat	DA	Testni slučaj u potpunosti ispravno izvršen
3.	Pozivanje kabine na 2. kat	DA	Testni slučaj u potpunosti ispravno izvršen
4.	Pozivanje kabine na 3. kat	DA	Testni slučaj u potpunosti ispravno izvršen
5.	Pozivanje kabine na 4. kat	DA	Testni slučaj u potpunosti ispravno izvršen
6.	Provjera ispravnosti svjetlosne signalizacije	DA	Testni slučaj u potpunosti ispravno izvršen
7.	Provjera ispravnosti zvučne signalizacije	DA	Testni slučaj u potpunosti ispravno izvršen
8.	Provjera ispravnosti kontrole nad vratima kabine	DA	Testni slučaj u potpunosti ispravno izvršen
9.	Provjera ispravnosti sustava za upozorenje od preopterećenja kabine	DA	Testni slučaj u potpunosti ispravno izvršen
10.	Provjera ispravnosti povezivanja na sustav za udaljeni nadzor i kontrolu	DA	Testni slučaj u potpunosti ispravno izvršen
11.	Provjera ispravnosti rada SCADA sustava	DA	Testni slučaj u potpunosti ispravno izvršen
12.	Provjera ispravnosti LCD zaslona	DA	Testni slučaj u potpunosti ispravno izvršen
13.	Pozivanje kabine sustava na više katova u isto vrijeme	NE	Testni slučaj nije u potpunosti ispravno izvršen, kabina dizala odlazi na kat koji je zadnji pozvan

Tablica 4.1. Rezultati testiranja funkcionalnosti vertikalnog transportnog sustava.

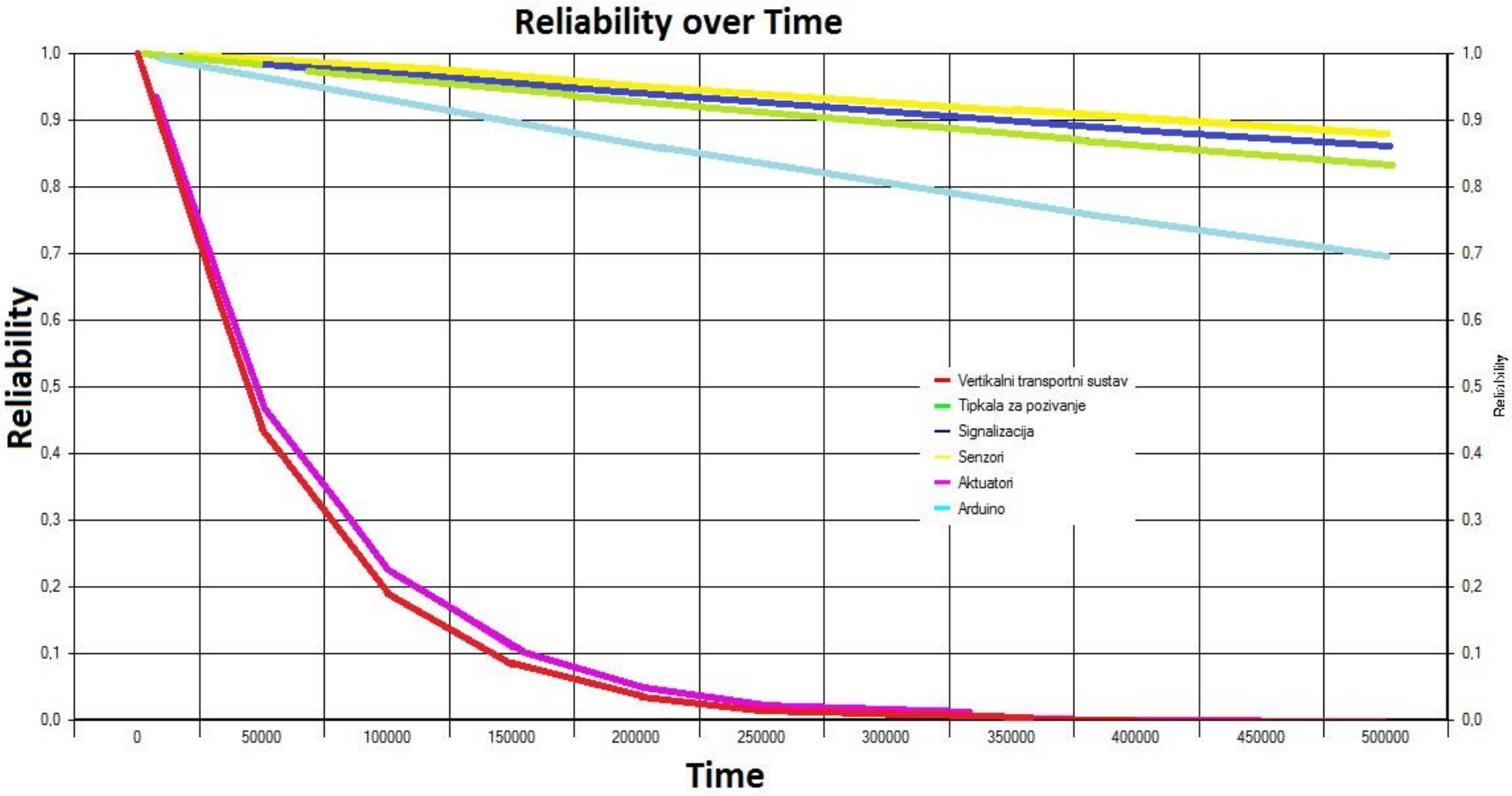
Iz tablice 4.1. lako je zaključiti da je većina test slučaja u potpunosti ispravno izvršena, iz tih dobivenih rezultata može se zaključiti da maketa vertikalnog transportnog sustava ispunjava svrhu za koju je napravljena. Jedini test slučaj koji nije u potpunosti izvršen je pozivanje više katova u isto vrijeme ali to se nije ni zahtjevalo u ovom projektu pa to ostaje kao jedna od opcija za moguće buduće nadogradnje.

Rezultati testiranja pouzdanosti sustava

Rezultati ispitivanja se dobivaju nakon što se cijeli elektronički sklop transportnog sustava unese u programski paket RELEX. To se radi na način da RELEX ima bazu podataka s elektroničkim komponentama i uzimaju se one komponente iz baze od kojih se sastoji elektronika sustava. Ta elektonika može se podjeliti u nekoliko grupa kao što je napravljeno i u ovom slučaju. Tako osim glavnog sustava imamo podsustave Arduino, tipkala za pozivanje, signalizacija, senzori i aktuatori. Za svaki taj podsustav su dodane elektroničke ili elektromehaničke komponente od kojih se oni sastoje, pa recimo za aktuatore imamo DC motor, servo motor i H-most. Za prikaz rezultata je generirano default report izvješće na kojem se vidi koliko je MTBF (srednje vrijeme između kvarova) te intezitet kvarova. Grafički je prikazana ovisnost pouzdanosti o vremenu.



Slika 4.1. Rezultati ispitivanja elektroničkog sklopa transportnog sustava.



Slika 4.2. Graf ovisnosti pouzdanosti sustava o vremenu.

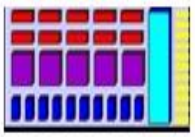
Relexov *DefaulReport* (slika 4.3) omogućava da dobijemo tablični prikaz elektroničkog sklopa i njegovih dijelova koji su izabrani u bazi podataka Relexa i za koje želimo ispitati pouzdanost. Kao što se može vidjeti u tablicama dobijemo niz specifikacija o komponentama sklopa. Također imamo i uvid u MTBF tj. srednje vrijeme između kvarova ili vrijeme koliko bi pojedina komponenta trebala ispravno raditi prije kvara.

Iz dobivenih podataka da se zaključiti da podsustav **aktuatori** koji se sastoji od jednog DC motora, jednog servo motora i H-mosta ima daleko najveći intezitet kvarova i najmanje vrijeme između kvarova (MTBF) te time ruši pouzdanost cijeloukupnog transportnog sustava. Najveću pouzdanost ima podsustav **senzori** koji se sastoji od elektroničkih komponenti otpornika i optičkih senzora.

Relex Software Report Page# 1

Standard Reliability Prediction Report

Part Number	Sustav	Description	Top-level assembly
Reference Des		File Name	dizalo.RPJ
Date	lipanj 6, 2016	Time	8:47
Environment	GB, GC - Ground Benign, Controlled	Failure Rate	16.469797
Temperature	30.00	MTBF	60,717



Assembly Name	Part Number	Ref Des	Qty	Failure Rate	MTBF
Vertikalni transportni sustav	Sustav		1.00	16.469797	60,717
Tipkala za pozivanje	E		1.00	0.364809	2,741,161
Signalizacija	D		1.00	0.304168	3,287,653
Senzori	C		1.00	0.262928	3,803,329
Aktuatori	B		1.00	14.810178	67,521
Arduino	A		1.00	0.727714	1,374,167

Slika 4.3. *Default report* RELEX-ov izvještaj za transportni susutav

5. ZAKLJUČAK

Zadatak ovoga rada je projektirati, izraditi i evaluirati sustav za vertikalni transport u zgradarstvu. Sustav projektirati na način da opslužuje transportne zahtjeve u modelskoj zgradi s 4 kata i prizemljem. Ugraditi zaštitne mehanizme od preopterećenja kabine te dodati ostale najkorištenije mehanizme i metode u prijevozu. Izraditi sustav za udaljeni nadzor i upravljanje ovim transportnim sustavom. Transportna ili prijevozna usluga je kretanje ljudi i dobara s jednog mjesta na drugo. Načini transporta jesu zračni, željeznički, cestovni, voda, kablovi, cjevovodi i prostor. Transportna polja mogu se podijeliti u tri grane: transportna infrastruktura, vozila i transportne operacije. Ona se ne može uskladištiti, sačuvati ili ponuditi na tržištu jer nema materijalni oblik kao što je slučaj sa samom robnom proizvodnjom. Vertikalni transportni sustav u zgradarstvu ili dizalo je uređaj za prijevoz ljudi ili tereta među katovima zgrade ili radnih platformi. Dizala su našla primjenu u mnogim prostorima. Najčešća primjena im je prijevoz putnika između katova u zgradama, prijevoz hrane i posuđa u restoranima sa nivoa pripreme do nivoa posluživanja, u bolnicama. Cijela mehanička konstrukcija transposrtnog sustava kao i kabina dizala je ručni rad. Projektirana i izrađena u radionici korištenjem raznih materijala, drvenih, metalnih, plastičnih i pleksiglasa. Sve najvažnije elektroničke i elektromehaničke komponente transportnog sustava ugrađene su u gornjem dijelu makete. Tu se nalazi mikroupravljač, aktuator za pokretanje kabine i ispravljač za napajanje. Mikroupravljač i „srce“ ovog transportnog sustava je Arduino Mega 2560, aktuator je DC motor 12V, napajanje cijelog sustava dobiveno je iz AC/DC ispravljača koji daje 12V istosmjernog napona. U ovom projektu je korišten optički senzor TCRT5000, točnije njih 9 komada. Kontrola nad vratima izvodi se pomoću malog servo motora, preopterečenje kabine pomoću vage ugrađene u kabinu a pozivanje kabine na kat pomoću tipkala za pozivanje. Sustav za udaljeni nadzor i kontrolu transportnog sustava ili „SCADA sustav (eng. Supervisory Control And Data Acquisition) predstavlja računalni sustav za nadzor, mjerenje, i upravljanje industrijskim sustavima. Svaki industrijski proces kojeg ima smisla automatizirati je odličan kandidat za primjenu SCADA sustava. SCADA sustav za ovaj sustav transporta izveden je u programskom jeziku C#. C# je nastao s ciljem da bude jednostavan, siguran, moderan, objektno orijentiran jezik visokih performansi za .NET platformu. C# je nastao na temelju objektnih jezika Java, C++ i Visual Basic. Vrlo je sličan Javi i C++ jeziku (sintaksa i semantika je dobrim dijelom preuzeta iz Jave, koja je kao i C# potpuno objektno orijentirani jezik). Integracija ovoga sustava je spajanje hardverskih i softverskih dijelova sustava. Potrebno je spojiti program iz mikroupravljača sa sustavom za

udaljeni nadzor. Komunikacija se odvija putem serijske USB veze. Ispitavanje funkcionalnosti sustava je provedeno kroz testne slučajeve i najveći broj njih je uspješno završio. Određivanje pouzdanosti elektronike izvedeno je u programskom paketu RELEX.

6. LITERATURA

[1] Županović, I.; *Tehnologija cestovnog prometa*, Sveučilište u Zagrebu, Fakultet prometnih znanosti, Zagreb 1994., str. 24.

[2] Elevator, *https://en.wikipedia.org/wiki/Elevator/*, (Pristupljeno 21.4.2016.)

[3] Dizalo, *http://povijest.hr/tag/dizalo/*, (Pristupljeno 21.4. 2016.)

[4] The Elevator System, *http://www.revit-content.com/content/elevator/*, (Pristupljeno 21.4.2016.)

[5] Ugradnja dizala, *http://www.hkis.hr/*, (Pristupljeno 30.4.2016.)

[6] Optički senzori, *http://mehatronik.com/2014/03/opticki-senzori/*, (Pristupljeno 14.5.2016.)

[7] Optical sensors, *http://futureelectronics.com/technologies/optical-sensors/*, (Pristupljeno 14.5.2016.)

[8] Aktuatori, *http://people.etf.unsa.ba/~jvelagic/laras/dok/Lekcija1.pdf*, (Pristupljeno 14.5.2016.)

[9] DC motor, *http://www.batteryspace.com/dcmotorheavyduty90vdcmotor.aspx*, (Pristupljeno 14.5.2016.)

[10] Servo motor, *http://elab.pk/en/servo-motors/65-servo-motor-9g-mini.html*, (Pristupljeno 14.5.2016.).

[11] Arduino Piezo speakers, *http://shallowsky.com/arduino/class/buzzer.html*, (Pristupljeno 14.5.2016.)

[12] Arduino Mega, *http://www.electroschematics.com/7963/arduino/pinout/*, (Pristupljeno 14.5.2016.)

[13] Arduino Mega Specification, *https://www.arduino.cc//en//Main/ArduinoBoard Mega2560*, (Pristupljeno 1.6.2016.)

[14] Kiš Miroslav, *Englesko-hrvatski i hrvatsko-engleski informatički rječnik*, Zagreb, Naklada Ljevak, 2000., str. 36.

[15] SCADA, *http://spvp.zesoi.fer.hr/seminari/2001/scada/* , (Pristupljeno 6.6.2016).

[16] What is Arduino, *http://e-elektro.blogspot.hr/2014/06/sto-je-arduino.html*, (Pristupljeno 6.6.2016.)

[17] Arduino programsko okruženje, *http://www.arduino.cc/en/Guide/MacOSX*, (Pristupljeno 6.6.2016.)

[18] CSharp, *http://www.etfos.unios.hr/~lukic/oop/Auditorne_vjezC5%BEbe_5.pdf* , (Pristupljeno 6.6.2016.)

[19] *Predložak za 1. labaratorijsku vježbu iz kolegija pouzdanost i dijagnostika računalnih sustava*, Sveučilište u Osijeku, Elektrotehnički fakultet u Osijeku, Osijek, 2016, str. 4)

[20] Load Cell, *http://qqtrading.com.my/flex-force-sensor*, (Pristupljeno 6.6.2016.)

[21] INA125, *http://www.ti.com/product/INA125*, (Pristupljeno 6.6.2016.)

[22] Tipkalo, *https://scratch.mit.edu/studios/1778374/activity/*, (Pristupljeno 6.6.2016.)

[23] Led dioda, *http://lampatronics.com/product-category/ledled-stripirfuse/led/* (Pristupljeno 6.6.2016.)

[24] LCD zaslon, *https://electrosome.com/interfacing-lcd-atmega32-microcontroller-atmel-studio/*, (Pristupljeno 6.6.2016.)

SAŽETAK

Da bi se ovaj projekt mogao ostvariti potrebno je provesti pet glavnih koraka. Prvi korak je praktična realizacija makete, tj. potrebno je konstruirati modelski oblik zgrade tako da vertikalni transportni sustav unutar nje same može opsluživati četiri kata i prizemlje. Drugi korak je električki ustroj, instalacija svih glavnih elektroničkih elemenata kao što su mikroupravljač, senzori, aktuatori, tipkala, sustav za zaštitu od preopterečenja kabine. U trećem koraku se provodi implementacija programskog koda mikroupravljača kojim on upravlja transportnim sustavom. Četvrti korak je implementacija programskog koda u programskom jezik C# za udaljeni nadzor i kontrolu transportnog sustava. Peti korak je puštanje transportnog sustava u pogon, ispitivanje mogućnosti sustava i ispravljanje pogrešaka. Inače vertikalni transportni sustavi su u današnje moderno vrijeme neizostavan dio svake zgrade jer omogućuju brzi transport između katova zgrade kojih može biti i preko stotinjak.

KLJUČNE RIJEČI

Vertikalni transport, mikroupravljač, aktuatori, senzori, algoritam, SCADA, C#, RELEX.

ABSTRACT

In order to realize this project could need to spend five main steps. The first step is the practical realization of the model, ie. It is necessary to construct a model form of the building so that the vertical transport system within itself can serve four floors and ground floor. The second step is electrically structure, installation of all major electronic elements such as microcontroller, sensors, actuators, push buttons, system overload protection cabins. In the third step is carried out implementation of the code microcontroller which he managed transport system. The fourth step is the implementation of the program code in the programming language C # for remote monitoring and control of the transmission system. The fifth step is the release of the transport system in operation, examining the possibilities of the system and debugging. Otherwise vertical transport systems are in today's modern times is an indispensable part of every building because they allow rapid transport between floors of the building which can be over a hundred.

KEYWORDS

Vertical transportation, microcontroller, actuators, sensors, algorithm, SCADA, C#, RELEX.

ŽIVOTOPIS

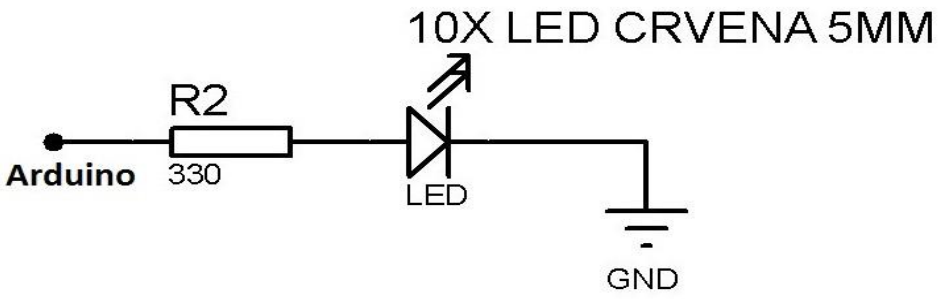
Hrvoje Cindrić, rođen je 12. veljače 1992. godine u Starim Mikanovcima u kojima je odrastao i pohađao Osnovnu školu Stjepana Cvrkovića. Nakon završetka osnovne škole upisuje Tehničku školu Ruđera Boškovića u Vinkovcima, smjer elektrotehničar. Za vrijeme srednje škole sudjeluje na županijskom natjecanju mladih tehničara i osvaja 1. mjesto s kojim odlazi na 52. državno natjecanje mladih tehničara u Dubrovnik. Nakon završetka srednje škole upisuje stručni studij automatike na Elektrotehničkom fakultetu u Osijeku, koji završava 2013. godine. Trenutno je student 2. godine diplomskog studija procesnog računarstva na Elektrotehničkom fakultetu u Osijeku. Posjeduje vozačku B kategorije a od stranih jezika koristi se engleskim.

7. PRILOZI

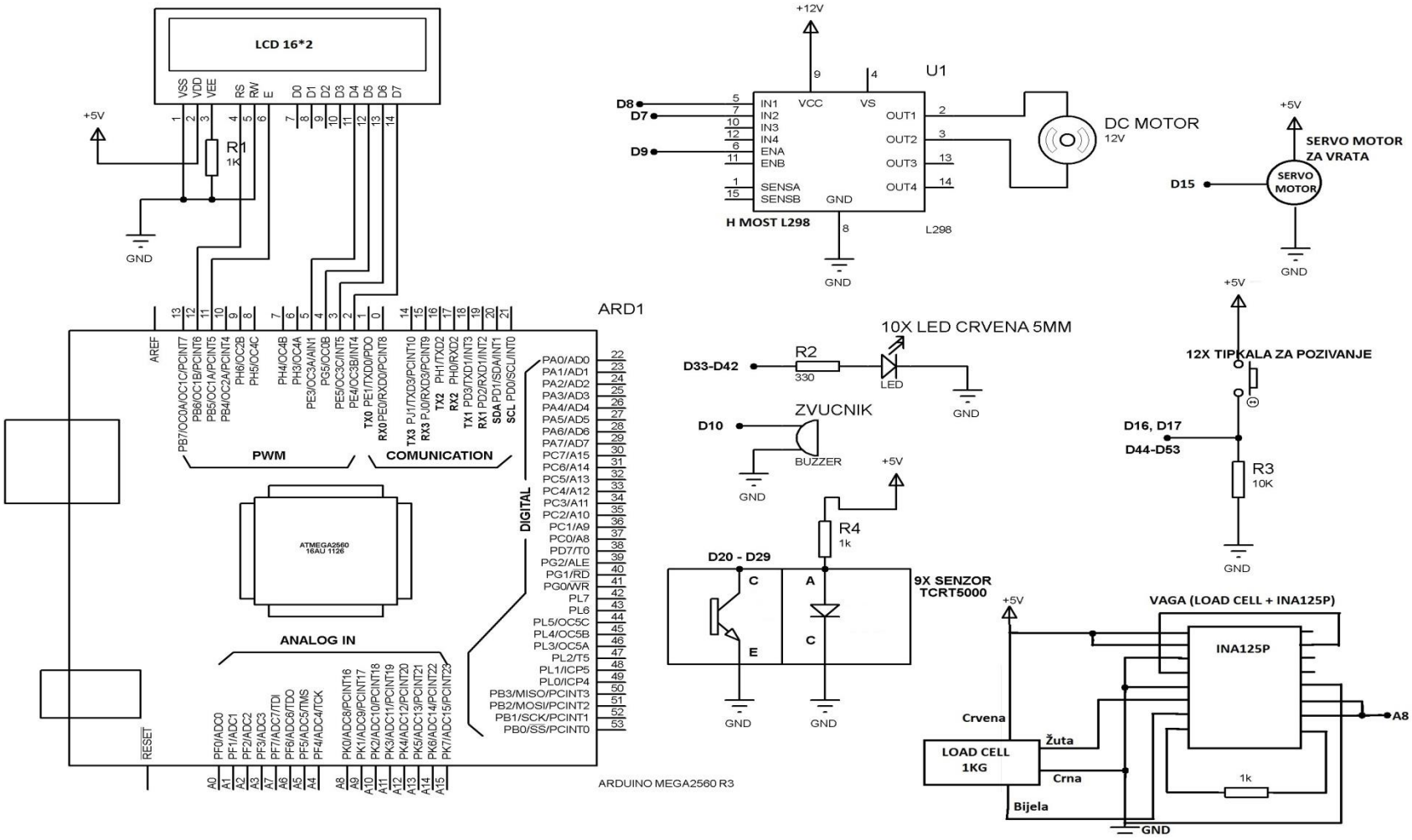
Svjetlosna signalizacija

Za svjetlosnu signalizaciju koriste se crvene led diode 5mm. Prilikom pritiska tipkala na nekom katu pali se led dioda koja svjetli sve dok kabina transportnog sustava ne pristigne na taj kat. Shema spajanja Led dioda na Arduino prikazana je na slici 7.2. Za spajanje se koriste digitalni pinovi na Arduinu.

Slika 7.1. Led dioda[23].



Slika 7.2. Shema spajanja led diode na Arduino.

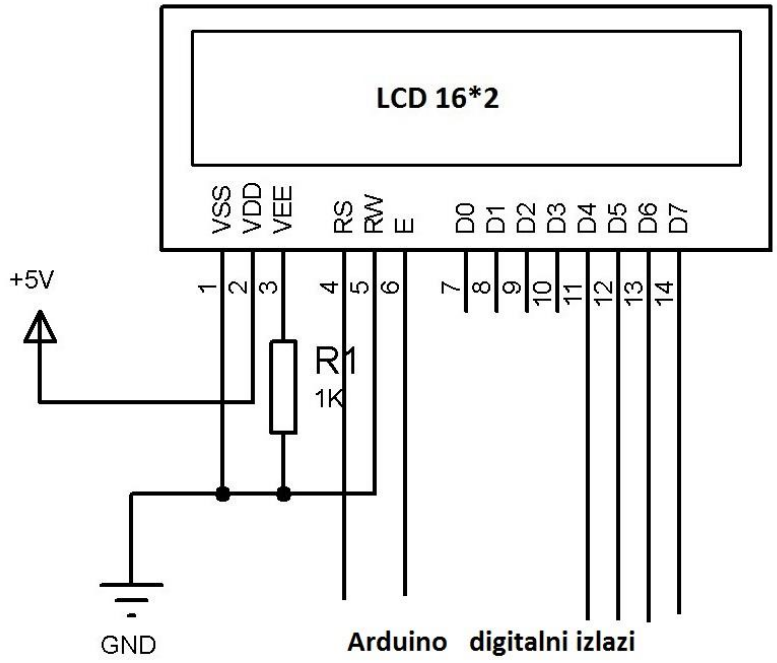


Slika 7.3. Električna shema vertikalnog transportnog sustava

LCD zaslon 16x2

Lcd ekran s HD44780 kontrolerom omogućuje prikazivanje teksta, simbola, vrijednosti ili bilo čega drugoga što dolazi s mikroupravljača. Sposoban pokazati 16 znakova u dva reda naći će svoju primjenu u mnoštvu projekata koji trebaju poslati jasno vidljivu i čitku izlaznu informaciju. Shema spajanja na Arduino je prikazana na slici 7.5.

Slika 7. 4. LCD 16x2 [24].



Slika 7.5. Shema spajanja lcd 16x2 na mikroupravljač

Programski kod korišten u ovom projektu (Arduino)

```
#include <LiquidCrystal.h>
#include <Servo.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int i=1;
int loadCell = A8;
int vaga = 0;
Servo mojservo;
int tezina;
 char kretanje = 'D';
 char kat = '1';
 char vrata = 'Z';

// vrijeme timera za vagu i slanje podataka preko serial porta
unsigned long previousMillis = 0;
long OnTime = 500;

// timer za trajanje otvorenih vrata
uint32_t konacno_vrijeme;
uint32_t poc_vrijeme;

//timer kad lift dodje na kat da se vrata otovre i zatvore nakon 10 sekundi
uint32_t konacno_vrijeme_dva;
uint32_t poc_vrijeme_dva;

//zvucnik
int buzzer = 10;

int zadnje_stanje_kat5 = 0;
int zadnje_stanje_kat4 = 0;
int zadnje_stanje_kat3 = 0;
int zadnje_stanje_kat2 = 0;
int zadnje_stanje_kat1 = 0;

//definiranje karaktera na displeju
byte Chardown[8] = { // arrow down
  B00100,
  B00100,
  B00100,
  B00100,
  B10101,
  B01110,
  B00100,
  B00000};
byte Charup[8] = {  // arrow up
  B00100,
  B01110,
  B10101,
  B00100,
  B00100,
  B00100,
  B00100,
  B00000};
  byte Charright[8] = {   // arrow right
  B00000,
  B00100,
  B00010,
  B11111,
  B00010,
  B00100,
  B00000,
  B00000};
//vrata
int provjera_vrata = 18;
int stanje_provjera_vrata = 0;
int tip_otvori_vrata = 16;
int tip_zatvori_vrata = 17;
int stanje_otvori_vrata = 0;
int stanje_zatvori_vrata = 0;

//tipkala unutar lifta
int kat1_unutra = 53;
int kat2_unutra = 52;
int kat3_unutra = 50;
int kat4_unutra = 51;
int kat5_unutra = 49;
//tipakala izvan lifta
int kat1_vani = 48;
int kat2_vani = 47;
int kat3_vani = 46;
int kat4_vani = 45;
int kat5_vani = 44;
//senzori(infra red)
const int senzor_kat1 = 20;
const int senzor_kat2 = 21;
const int senzor_kat3 = 22;
const int senzor_kat4 = 13;
const int senzor_kat5 = 24;

//senzori medjukat
const int senzor_medju_1_2 = 29;
const int senzor_medju_2_3 = 28;
const int senzor_medju_3_4 = 27;
const int senzor_medju_4_5 = 26;


//ledice
int led_kat1_unutra = 42;
int led_kat5_unutra = 41;
int led_kat3_unutra = 40;
int led_kat2_unutra = 38;
int led_kat4_unutra = 39;
int led_kat1_vani = 37;
int led_kat2_vani = 36;
int led_kat3_vani = 35;
int led_kat4_vani = 34;
int led_kat5_vani = 33;
// status tipkala
int kat1_unutra_status = 0;
int kat2_unutra_status = 0;
int kat3_unutra_status = 0;
int kat4_unutra_status = 0;
int kat5_unutra_status = 0;
int kat1_vani_status = 0;
int kat2_vani_status = 0;
int kat3_vani_status = 0;
int kat4_vani_status = 0;
int kat5_vani_status = 0;
//status senzora
int senzor_kat1_status = 0;
int senzor_kat2_status = 0;
int senzor_kat3_status = 0;
int senzor_kat4_status = 0;
int senzor_kat5_status = 0;

//status medjusenzora
int senzor_medju_1_2_status = 0;
int senzor_medju_2_3_status = 0;
int senzor_medju_3_4_status = 0;
int senzor_medju_4_5_status = 0;

// VRATA DIZALA
void otvoren_lift(){
  mojservo.attach(15);
  delay(250);
  mojservo.write(80);
  delay(250);
  mojservo.detach();
}

 void zatvoren_lift(){
   mojservo.attach(15);
   delay(250);
```

```
  mojservo.write(165);
  delay(250);
  mojservo.detach();
 }


//motor
int motPin1 = 8;
int motPin2 = 7;
int speedPin = 9;

// stanja motora
 void motor_dolje()
 {
  analogWrite( speedPin, 255);
  digitalWrite(motPin1, LOW);
  digitalWrite(motPin2, HIGH);
  }
  void motor_gore()
 {
   analogWrite( speedPin, 255);
  digitalWrite(motPin1, HIGH);
  digitalWrite(motPin2, LOW);
 }
   void motor_stop()
   {
    analogWrite( speedPin, 75);
   digitalWrite(motPin1, LOW);
   digitalWrite(motPin2, LOW);
  }

  // USPORAVANJE MOTOROM
  void usporavanje_dolje_prvo(){
   digitalWrite(motPin1, LOW);
  digitalWrite(motPin2, HIGH);
  for(int usd = 255; usd>100; --usd){
   analogWrite(speedPin, usd);
  }
  }
  void usporavanje_dolje_drugo(){
   digitalWrite(motPin1, LOW);
  digitalWrite(motPin2, HIGH);
  for(int usd = 99; usd>70; --usd){
   analogWrite(speedPin, usd);
  }
  }
   void usporavanje_dolje_trece(){
   digitalWrite(motPin1, LOW);
  digitalWrite(motPin2, HIGH);
  for(int usd = 69; usd>50; --usd){
   analogWrite(speedPin, usd);
  }
   }
  void usporavanje_gore_prvo(){
    digitalWrite(motPin1, HIGH);
  digitalWrite(motPin2, LOW);
  for(int usg = 255; usg>140; --usg){
   analogWrite(speedPin, usg);
  }
  }
  void usporavanje_gore_drugo(){
    digitalWrite(motPin1, HIGH);
  digitalWrite(motPin2, LOW);
  for(int usg = 139; usg>100; --usg){
   analogWrite(speedPin, usg);
  }
  }
   void usporavanje_gore_trece(){
    digitalWrite(motPin1, HIGH);
  digitalWrite(motPin2, LOW);
  for(int usg = 99; usg>80; --usg){
   analogWrite(speedPin, usg);
  }
  }
  //UBRZAVANJE MOTOROM
  void ubrzavanje_prvo_gore(){
   digitalWrite(motPin1, HIGH);
  digitalWrite(motPin2, LOW);
  for(int usg_1 = 0; usg_1<109; usg_1++){
   analogWrite(speedPin, usg_1);
  }
  }
  void ubrzavanje_drugo_gore(){
   digitalWrite(motPin1, HIGH);
  digitalWrite(motPin2, LOW);
  for(int usg_2 = 110; usg_2<255; usg_2++){
   analogWrite(speedPin, usg_2);
  }
  }

  void ubrzavanje_prvo_dolje(){
   digitalWrite(motPin1, LOW);
  digitalWrite(motPin2, HIGH);
  for(int usg_1 = 0; usg_1<89; usg_1++){
   analogWrite(speedPin, usg_1);
  }
  }
  void ubrzavanje_drugo_dolje(){
   digitalWrite(motPin1, LOW);
  digitalWrite(motPin2, HIGH);
  for(int usg_2 = 90; usg_2<255; usg_2++){
   analogWrite(speedPin, usg_2);
  }
  }
// ZVUCNIK
  void zvucnik(){
   digitalWrite(buzzer, HIGH);
   delay(2000);
      digitalWrite(buzzer, LOW);
  }

  void upozorenje(){
   digitalWrite(buzzer, HIGH);
   digitalWrite(led_kat1_vani, HIGH);
  digitalWrite(led_kat1_unutra, HIGH);
  digitalWrite(led_kat2_vani, HIGH);
  digitalWrite(led_kat2_unutra, HIGH);
  digitalWrite(led_kat3_vani, HIGH);
  digitalWrite(led_kat3_unutra, HIGH);
   digitalWrite(led_kat4_vani, HIGH);
  digitalWrite(led_kat4_unutra, HIGH);
  digitalWrite(led_kat5_vani, HIGH);
  digitalWrite(led_kat5_unutra, HIGH);
   delay(200);
   digitalWrite(buzzer, LOW);
   digitalWrite(led_kat1_vani, LOW);
  digitalWrite(led_kat1_unutra, LOW);
  digitalWrite(led_kat2_vani, LOW);
  digitalWrite(led_kat2_unutra, LOW);
  digitalWrite(led_kat3_vani, LOW);
  digitalWrite(led_kat3_unutra, LOW);
   digitalWrite(led_kat4_vani, LOW);
  digitalWrite(led_kat4_unutra, LOW);
  digitalWrite(led_kat5_vani, LOW);
  digitalWrite(led_kat5_unutra, LOW);
   delay(200);
  }
////////////////////////////////////////////////////////////////////////////
 void setup() {
  lcd.begin(16, 2);
  Serial.begin(9600);
 lcd.createChar(1, Chardown);
 lcd.createChar(2, Charup);
 lcd.createChar(3, Charright);

 poc_vrijeme = millis();
 poc_vrijeme =  konacno_vrijeme;

  poc_vrijeme_dva = millis();
 poc_vrijeme_dva =  konacno_vrijeme_dva;

  pinMode(buzzer, OUTPUT);
```

```
 pinMode(tip_otvori_vrata, INPUT);
 pinMode(tip_zatvori_vrata, INPUT);
 pinMode(provjera_vrata, INPUT);
 pinMode(motPin1, OUTPUT);
 pinMode(motPin2, OUTPUT);
 pinMode(speedPin, OUTPUT);
 pinMode(kat1_unutra, INPUT);
 pinMode(kat2_unutra, INPUT);
 pinMode(kat3_unutra, INPUT);
 pinMode(kat4_unutra, INPUT);
 pinMode(kat5_unutra, INPUT);
 pinMode(kat1_vani, INPUT);
 pinMode(kat2_vani, INPUT);
 pinMode(kat3_vani, INPUT);
 pinMode(kat4_vani, INPUT);
 pinMode(kat5_vani, INPUT);
 pinMode(led_kat1_unutra, OUTPUT);
 pinMode(led_kat2_unutra, OUTPUT);
 pinMode(led_kat3_unutra, OUTPUT);
 pinMode(led_kat4_unutra, OUTPUT);
 pinMode(led_kat5_unutra, OUTPUT);
 pinMode(led_kat1_vani, OUTPUT);
 pinMode(led_kat2_vani, OUTPUT);
 pinMode(led_kat3_vani, OUTPUT);
 pinMode(led_kat4_vani, OUTPUT);
 pinMode(led_kat5_vani, OUTPUT);
 pinMode(senzor_kat1, INPUT);
 pinMode(senzor_kat2, INPUT);
 pinMode(senzor_kat3, INPUT);
 pinMode(senzor_kat4, INPUT);
 pinMode(senzor_kat5, INPUT);
 digitalWrite(senzor_kat1, HIGH);
 digitalWrite(senzor_kat2, HIGH);
 digitalWrite(senzor_kat3, HIGH);
 digitalWrite(senzor_kat4, HIGH);
 digitalWrite(senzor_kat5, HIGH);
 pinMode(senzor_medju_1_2, INPUT);
 pinMode(senzor_medju_2_3, INPUT);
 pinMode(senzor_medju_3_4, INPUT);
 pinMode(senzor_medju_4_5, INPUT);
digitalWrite(senzor_medju_1_2, HIGH);
digitalWrite(senzor_medju_2_3, HIGH);
digitalWrite(senzor_medju_3_4, HIGH);
digitalWrite(senzor_medju_4_5, HIGH);

//Početni položaj -> npr ako je na na 3 katu spušta se u prizemlje
senzor_kat1_status = digitalRead(senzor_kat1);
senzor_kat2_status = digitalRead(senzor_kat2);
senzor_kat3_status = digitalRead(senzor_kat3);
senzor_kat4_status = digitalRead(senzor_kat4);
senzor_kat5_status = digitalRead(senzor_kat5);

if(senzor_kat2_status == LOW || senzor_kat3_status == LOW || senzor_kat4_status == LOW || senzor_kat5_status == LOW){

       motor_dolje();

      if(senzor_kat1_status == LOW){

      motor_stop();
  }
  }

 }

void loop() {

 int c=Serial.read();
 vaga = analogRead(loadCell);

unsigned long currentMillis = millis();

senzor_kat1_status = digitalRead(senzor_kat1);
senzor_kat2_status = digitalRead(senzor_kat2);
senzor_kat3_status = digitalRead(senzor_kat3);
senzor_kat4_status = digitalRead(senzor_kat4);
senzor_kat5_status = digitalRead(senzor_kat5);


senzor_medju_1_2_status=digitalRead(senzor_medju_1_2);
senzor_medju_2_3_status=digitalRead(senzor_medju_2_3);
senzor_medju_3_4_status=digitalRead(senzor_medju_3_4);
senzor_medju_4_5_status=digitalRead(senzor_medju_4_5);


kat1_unutra_status = digitalRead(kat1_unutra);
kat2_unutra_status = digitalRead(kat2_unutra);
kat3_unutra_status = digitalRead(kat3_unutra);
kat4_unutra_status = digitalRead(kat4_unutra);
kat5_unutra_status = digitalRead(kat5_unutra);

kat1_vani_status = digitalRead(kat1_vani);
kat2_vani_status = digitalRead(kat2_vani);
kat3_vani_status = digitalRead(kat3_vani);
kat4_vani_status = digitalRead(kat4_vani);
kat5_vani_status = digitalRead(kat5_vani);

stanje_provjera_vrata= digitalRead(provjera_vrata);
stanje_otvori_vrata = digitalRead(tip_otvori_vrata);
stanje_zatvori_vrata=digitalRead(tip_zatvori_vrata);
////////////////////////////////////////////////////////////////////////////

///  4KAT
if(stanje_provjera_vrata==HIGH&&
kat5_vani_status == HIGH || kat5_unutra_status == HIGH && stanje_provjera_vrata == HIGH || c=='5' && stanje_provjera_vrata == HIGH)
   {
while(i<5){
    i++;
    ubrzavanje_prvo_gore();
   digitalWrite(led_kat5_vani, HIGH);
  digitalWrite(led_kat5_unutra, HIGH);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(2));
  lcd.setCursor(4, 0);
  lcd.print("4 kat");
   kretanje = 'G';
  kat = '5';
  vrata = 'Z';
  }
    }
   if(i == 5 && senzor_medju_1_2_status == LOW || i == 5 && senzor_medju_2_3_status == LOW || i==5 && senzor_medju_3_4_status == LOW){
     ubrzavanje_drugo_gore();
    }
    if( i==5 && senzor_medju_3_4_status == LOW){
      usporavanje_gore_prvo();
    }
    if( i==5 && senzor_kat4_status == LOW){
      usporavanje_gore_drugo();
    }
    if( i==5 && senzor_medju_4_5_status == LOW){
      usporavanje_gore_trece();
    }
  if( senzor_kat5_status != zadnje_stanje_kat5){
  if (i==5 && senzor_kat5_status == LOW){
    i=5;
  motor_stop();
  digitalWrite(led_kat5_vani, LOW);
  digitalWrite(led_kat5_unutra, LOW);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(3));
```

```
  lcd.setCursor(4, 0);
  lcd.print("4 kat");
  zvucnik();
  otvoren_lift();
  poc_vrijeme_dva = millis();
  }
     }
     zadnje_stanje_kat5 = senzor_kat5_status;
////////////////////////////////////////////////////////////////////////
```

/// **KAT3**

```
  if(stanje_provjera_vrata == HIGH &&
kat4_vani_status == HIGH || kat4_unutra_status ==
HIGH && stanje_provjera_vrata == HIGH || c=='4'
&& stanje_provjera_vrata == HIGH ){
  if(i>4){
     while(i>4) {
     i--;
     ubrzavanje_prvo_dolje();
  digitalWrite(led_kat4_vani, HIGH);
  digitalWrite(led_kat4_unutra, HIGH);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(1));
  lcd.setCursor(4, 0);
    lcd.print("3 kat");
     kretanje = 'D';
     kat = '4';
     vrata = 'Z';
 }
   }
    if(i<4){
     while(i<4){
       i++;
       ubrzavanje_prvo_gore();
  digitalWrite(led_kat4_vani, HIGH);
  digitalWrite(led_kat4_unutra, HIGH);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(2));
  lcd.setCursor(4, 0);
  lcd.print("3 kat");
   kretanje = 'G';
  kat = '4';
  vrata = 'Z';
    }
    }
    }
    if(i==4 && senzor_medju_4_5_status == LOW){
      ubrzavanje_drugo_dolje();
    }
    if(i==4 && senzor_medju_1_2_status == LOW ||
i==4 && senzor_medju_2_3_status == LOW ){
      ubrzavanje_drugo_gore();
    }
    if(i==4 && senzor_kat5_status == LOW){
      usporavanje_dolje_drugo();
    }
    if(i==4 && senzor_medju_4_5_status == LOW){
      usporavanje_dolje_trece();
    }
    if(i==4 && senzor_medju_2_3_status == LOW){
      usporavanje_gore_prvo();
    }
    if(i==4 && senzor_kat3_status == LOW){
      usporavanje_gore_drugo();
    }
    if(i==4 && senzor_medju_3_4_status == LOW){
      usporavanje_gore_trece();
    }
    if( senzor_kat4_status != zadnje_stanje_kat4){
     if (i==4 && senzor_kat4_status == LOW){
          i=4;
      motor_stop();
  digitalWrite(led_kat4_vani, LOW);
  digitalWrite(led_kat4_unutra, LOW);
  lcd.clear();

  lcd.setCursor(2, 0);
  lcd.print(char(3));
  lcd.setCursor(4, 0);
    lcd.print("3 kat");
    zvucnik();
     otvoren_lift();
  poc_vrijeme_dva = millis();
  }
  }
     zadnje_stanje_kat4 = senzor_kat4_status;

////////////////////////////////////////////////////////////////////////
```

/// **KAT 2**

```
if(stanje_provjera_vrata == HIGH &&
kat3_vani_status == HIGH || kat3_unutra_status ==
HIGH && stanje_provjera_vrata == HIGH || c=='3'
&& stanje_provjera_vrata == HIGH){
  if(i>3){
     while(i>3) {
     i--;
     ubrzavanje_prvo_dolje();
    // motor_dolje();
  digitalWrite(led_kat3_vani, HIGH);
  digitalWrite(led_kat3_unutra, HIGH);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(1));
  lcd.setCursor(4, 0);
  lcd.print("2 kat");
   kretanje = 'D';
  kat = '3';
  vrata = 'Z';
    }
    }
    if(i<3){
     while(i<3){
       i++;
       ubrzavanje_prvo_gore();
       //motor_gore();
  digitalWrite(led_kat3_vani, HIGH);
  digitalWrite(led_kat3_unutra, HIGH);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(2));
  lcd.setCursor(4, 0);
  lcd.print("2 kat");
   kretanje = 'G';
  kat = '3';
  vrata = 'Z';
    }
    }
   }
   if(i==3 && senzor_medju_1_2_status == LOW){
    ubrzavanje_drugo_gore();
   }
   if(i==3 && senzor_medju_4_5_status == LOW){
    ubrzavanje_drugo_dolje();
   }
   if(i==3 && senzor_medju_4_5_status == LOW){
     usporavanje_dolje_prvo();
   }
   if(i==3 && senzor_kat4_status == LOW){
     usporavanje_dolje_drugo();
   }
    if(i==3 && senzor_medju_3_4_status == LOW){
     usporavanje_dolje_trece();
   }
    if(i==3 && senzor_medju_1_2_status == LOW){
     usporavanje_gore_prvo();
   }
    if(i==3 && senzor_kat2_status == LOW){
     usporavanje_gore_drugo();
```

```
   }
   if(i==3 && senzor_medju_2_3_status == LOW){
    usporavanje_gore_trece();
   }
 if( senzor_kat3_status != zadnje_stanje_kat3){
 if (i==3 && senzor_kat3_status == LOW){
     i=3;
    motor_stop();
 digitalWrite(led_kat3_vani, LOW);
 digitalWrite(led_kat3_unutra, LOW);
 lcd.clear();
 lcd.setCursor(2, 0);
 lcd.print(char(3));
 lcd.setCursor(4, 0);
 lcd.print("2 kat");
    zvucnik();
 otvoren_lift();
 poc_vrijeme_dva = millis();
}
  }
 zadnje_stanje_kat3 = senzor_kat3_status;

//////////////////////////////////////////////////////////////////

//  KAT 1

 if(stanje_provjera_vrata == HIGH &&
kat2_vani_status == HIGH || kat2_unutra_status ==
HIGH && stanje_provjera_vrata == HIGH || c=='2'
&& stanje_provjera_vrata == HIGH)
 {
    if(i>2){
    while(i>2) {
    i--;
    ubrzavanje_prvo_dolje();
    //motor_dolje();
 digitalWrite(led_kat2_vani, HIGH);
 digitalWrite(led_kat2_unutra, HIGH);
 lcd.clear();
 lcd.setCursor(2, 0);
 lcd.print(char(1));
 lcd.setCursor(4, 0);
 lcd.print("1 kat");
  kretanje = 'D';
 kat = '2';
 vrata = 'Z';
  }
  }
  if(i<2){
   while(i<2){
     i++;
    ubrzavanje_prvo_gore();
    motor_gore();
 digitalWrite(led_kat2_vani, HIGH);
 digitalWrite(led_kat2_unutra, HIGH);
 lcd.clear();
 lcd.setCursor(2, 0);
 lcd.print(char(2));
 lcd.setCursor(4, 0);
 lcd.print("1 kat");
  kretanje = 'G';
 kat = '2';
 vrata = 'Z';
  }
  }
  }
  if(i==2 && senzor_medju_4_5_status == LOW){
   ubrzavanje_drugo_dolje();
  }
  if(i==2 && senzor_medju_3_4_status == LOW){
   usporavanje_dolje_prvo();
  }
  if(i==2 && senzor_kat3_status == LOW){
   usporavanje_dolje_drugo();
  }
   if(i==2 && senzor_medju_2_3_status == LOW){
    usporavanje_dolje_trece();
   }

  if(i==2 && senzor_kat1_status == LOW){
   usporavanje_gore_drugo();
  }
  if(i==2 && senzor_medju_1_2_status == LOW){
   usporavanje_gore_trece();
  }

   if( senzor_kat2_status != zadnje_stanje_kat2){
   if (i== 2 && senzor_kat2_status == LOW){
   i=2;
   motor_stop();
 digitalWrite(led_kat2_vani, LOW);
 digitalWrite(led_kat2_unutra, LOW);
 lcd.clear();
 lcd.setCursor(2, 0);
 lcd.print(char(3));
 lcd.setCursor(4, 0);
 lcd.print("1 kat");
    zvucnik();
 otvoren_lift();
 poc_vrijeme_dva = millis();
}
    }
    zadnje_stanje_kat2 = senzor_kat2_status;

//////////////////////////////////////////////////////////////////////////////


 // PRIZEMLJE

if(stanje_provjera_vrata == HIGH &&
kat1_vani_status == HIGH || kat1_unutra_status ==
HIGH && stanje_provjera_vrata == HIGH || c=='1'
&& stanje_provjera_vrata == HIGH){
  if(i>1){
while(i>1){
   i--;
   ubrzavanje_prvo_dolje();
      //motor_dolje();
 digitalWrite(led_kat1_vani, HIGH);
 digitalWrite(led_kat1_unutra, HIGH);
 lcd.clear();
 lcd.setCursor(2, 0);
 lcd.print(char(1));
 lcd.setCursor(3, 0);
 lcd.print("prizemlje");
  kretanje = 'D';
 kat = '1';
 vrata = 'Z';
  }
  }
  }
  if(i==1 && senzor_medju_4_5_status == LOW ||
i==1 && senzor_medju_3_4_status == LOW || i==1
&& senzor_medju_2_3_status == LOW){
    ubrzavanje_drugo_dolje();
  }
  if(i==1 && senzor_medju_2_3_status == LOW){
   usporavanje_dolje_prvo();
  }
  if(i==1 && senzor_kat2_status == LOW){
   usporavanje_dolje_drugo();
  }
   if(i==1 && senzor_medju_1_2_status == LOW){
   usporavanje_dolje_trece();
  }
  if( senzor_kat1_status != zadnje_stanje_kat1){
  if (i==1 && senzor_kat1_status == LOW){
       i=1;
   motor_stop();
 digitalWrite(led_kat1_vani, LOW);
 digitalWrite(led_kat1_unutra, LOW);
 digitalWrite(led_kat2_vani, LOW);
```

```
  digitalWrite(led_kat2_unutra, LOW);
  digitalWrite(led_kat3_vani, LOW);
  digitalWrite(led_kat3_unutra, LOW);
   digitalWrite(led_kat4_vani, LOW);
  digitalWrite(led_kat4_unutra, LOW);
  digitalWrite(led_kat5_vani, LOW);
  digitalWrite(led_kat5_unutra, LOW);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(char(3));
  lcd.setCursor(3, 0);
  lcd.print("prizemlje");
     zvucnik();
  otvoren_lift();
  poc_vrijeme_dva = millis();
}
   }
   zadnje_stanje_kat1 = senzor_kat1_status;
////////////////////////////////////////////////////////////////////////
```

//Zatvaranje vrata nakon 7sek kad timer odbroji, kad dodje na kat

```
   if(poc_vrijeme_dva > 0 && millis() - poc_vrijeme_dva > 5000)
         {
          zvucnik();
         zatvoren_lift();
          poc_vrijeme_dva = 0;
         }
////////////////////////////////////////////////////////////////////////
```

// Pritiskom tipkala na katu otvaraju se vrata i sama zatvaraju nakon 7sec

```
if(kat1_vani_status == HIGH && senzor_kat1_status == LOW){
  zvucnik();
otvoren_lift();
vrata = 'O';
poc_vrijeme = millis();
}
if(kat2_vani_status == HIGH && senzor_kat2_status == LOW){
  zvucnik();
otvoren_lift();
vrata = 'O';
poc_vrijeme = millis();
}
if(kat3_vani_status == HIGH && senzor_kat3_status == LOW){
   zvucnik();
otvoren_lift();
vrata = 'O';
poc_vrijeme = millis();
}
if(kat4_vani_status == HIGH && senzor_kat4_status == LOW){
   zvucnik();
otvoren_lift();
vrata = 'O';
poc_vrijeme = millis();
}
if(kat5_vani_status == HIGH && senzor_kat5_status == LOW){
   zvucnik();
otvoren_lift();
vrata = 'O';
poc_vrijeme = millis();
}
if(poc_vrijeme > 0 && millis() - poc_vrijeme > 5000)
         {
          zvucnik();
         zatvoren_lift();
         vrata= 'Z';
          poc_vrijeme = 0;
         }
```

//prikaz na lcd-u jesu li vrata otvorena ili zatvorena

```
 if(stanje_provjera_vrata == HIGH ){
     lcd.setCursor(0, 1);
     lcd.print("Vrata zatvorena");
        vrata= 'Z';
  }
    else{
     lcd.setCursor(0, 1);
   lcd.print("Vrata  otvorena");
          vrata = 'O';

 }

////////////////////////////////////////////////////////////////////////
```

/// Tipkala koja otvaraju i zatvaraju vrata

```
 if(stanje_otvori_vrata == HIGH || c=='O'){
  otvoren_lift();
   }
 if(stanje_zatvori_vrata == HIGH || c=='Z'){
  zatvoren_lift();
 }

////////////////////////////////////////////////////////////////////////

         if((currentMillis - previousMillis >= OnTime))
          {
           int tezina = vaga * 8;
           if(tezina<200){
            tezina = 200;
           }
           if(tezina>=400){
            tezina=400;
            if(stanje_provjera_vrata == LOW && senzor_kat1_status == LOW || senzor_kat2_status == LOW || senzor_kat3_status == LOW || senzor_kat4_status == LOW || senzor_kat5_status == LOW){
             upozorenje();
             motor_stop();
            }
           }
          // SLANJE PODATAKA NA SERIJSKI PORT
Serial.print("<"),Serial.print(kretanje),Serial.print(kat),Serial.print(vrata),Serial.print(">"),Serial.println(tezina);
           previousMillis = currentMillis;
          }
           }
```

Programski kod korišten u ovom projektu (C#)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.Timers;

namespace dizalo
{
   public partial class dizalo : Form
   {
      string line;
      string tez;
      private class Item
      {
         // definiranje da bi mogli popunjavati Comboboxeve
         public string Name;
         public int Value;
         public Item(string name, int value)
         {
            Name = name; Value = value;

         }
         public override string ToString()
         {
            return Name;
         }

      }
      enum Pozicija
      {
         Gore, Dolje
      }
      private Pozicija pozicija_objekta;
      int x = 410;
      int y = 555;
      //(y) kat
      string prizemlje = "555";
      string kat_1 = "445";
      string kat_2 = "340";
      string kat_3 = "235";
      string kat_4 = "130";
      int width = 90;
      int height = 127;

      Image kabina = Image.FromFile("kabina.jpg");


      public dizalo()
      {

         InitializeComponent();

         broj_kata.Text = " P";
         vrata.Text = "Zatvorena";

         combo_serial.Items.Add(new Item("COM1", 1));
         combo_serial.Items.Add(new Item("COM2", 2));
         combo_serial.Items.Add(new Item("COM3", 3));
         combo_serial.Items.Add(new Item("COM4", 4));
         combo_serial.Items.Add(new Item("COM5", 5));
         combo_serial.Items.Add(new Item("COM6", 6));
         combo_serial.Items.Add(new Item("COM7", 7));
         combo_serial.Items.Add(new Item("COM8", 8));
         combo_serial.Items.Add(new Item("COM9", 9));
         combo_serial.Items.Add(new Item("COM10", 10));
         combo_serial.Items.Add(new Item("COM11", 11));
         combo_serial.Items.Add(new Item("COM12", 12));
         combo_serial.Items.Add(new Item("COM13", 13));
         combo_serial.Items.Add(new Item("COM14", 14));
         combo_serial.Items.Add(new Item("COM15", 15));

      }

      private void dizalo_Load(object sender, EventArgs e)
      {

      }

      private void dizalo_Paint(object sender, PaintEventArgs e)
      {
         e.Graphics.DrawImage(kabina, x, y, width, height);
      }

      private void timer1_Tick(object sender, EventArgs e)
      {

         if (pozicija_objekta == Pozicija.Dolje)
         {
            y += 5;
         }
         else if (pozicija_objekta == Pozicija.Gore)
         {
            y -= 5;

         }

          Invalidate();
      }

      private void odaberi_port_Click(object sender, EventArgs
e)
      {

            serialPort1.PortName =
combo_serial.SelectedItem.ToString();
            serialPort1.BaudRate = 9600;
            serialPort1.DtrEnable = true;

         try
         {
         if (!serialPort1.IsOpen)
         {
            serialPort1.DataReceived +=
serialPort1_DataReceived;
            serialPort1.Open();
            connect_textBox.Text = "  Connect!";
            connect_textBox.ForeColor = Color.Green;
         }
         }
         catch (Exception ex)
         {
            connect_textBox.Text = "  Disconnect!";
            connect_textBox.ForeColor = Color.Red;
         }

      }
      private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
      {
         line = serialPort1.ReadLine();
         this.Invoke(new EventHandler(timer2_Tick));


      }
      private void timer3_Tick(object sender, EventArgs e)
      {

         if (pozicija_textBox.Text == prizemlje.ToString())
```

```
        {
            broj_kata.Text = " P";
        }
        if (pozicija_textBox.Text == kat_1.ToString())
        {
            broj_kata.Text = " 1";

        }
        if (pozicija_textBox.Text == kat_2.ToString())
        {
            broj_kata.Text = " 2";

        }
        if (pozicija_textBox.Text == kat_3.ToString())
        {
            broj_kata.Text = " 3";

        }
        if (pozicija_textBox.Text == kat_4.ToString())
        {
            broj_kata.Text = " 4";

        }
    }

    private void timer2_Tick(object sender, EventArgs e)
    {

        pozicija_textBox.Text = y.ToString();
        timer2.Start();

        paket.Text = line.ToString();
        timer1.Enabled = true;

        char slovo_0 = line[0];
        char slovo_1 = line[1];
        char slovo_2 = line[2];
        char slovo_3 = line[3];
        char slovo_4 = line[4];
        string tez = line.Split('>').Last();
        string G = "G";
        string D = "D";
        string prvi = "1";
        string drugi = "2";
        string treci = "3";
        string cetvrti = "4";
        string peti = "5";
        string pocetni = "<";
        string zavrsni = ">";
        string otvorena = "O";
        string zatvorena = "Z";

        // progress bar za tezinu
         this.BeginInvoke(new LineReceivedEvent(LineReceived),
tez);


        // Parsiranje
        if (slovo_0.ToString() == pocetni.ToString() &&
slovo_4.ToString() == zavrsni.ToString())
        {
            ispravnost.Text = "Ispravan paket";
            ispravnost.ForeColor = Color.Green;
        }
        else
        {
            ispravnost.Text = "Neispravan paket";
            ispravnost.ForeColor = Color.Red;
            timer1.Stop();
        }


        if (slovo_1.ToString() == G.ToString())
        {
            pozicija_objekta = Pozicija.Gore;
        }
        else if (slovo_1.ToString() == D.ToString())
        {
            pozicija_objekta = Pozicija.Dolje;

        }
        ////////////////////////////////////////////
        if (slovo_2.ToString() == prvi.ToString())
        {
            if (y.ToString() == prizemlje.ToString())
            {
                timer1.Stop();
            }
        }

        else if (slovo_2.ToString() == drugi.ToString())
        {
            if (y.ToString() == kat_1.ToString())
            {

                timer1.Stop();
            }
        }
        else if (slovo_2.ToString() == treci.ToString())
        {
            if (y.ToString() == kat_2.ToString())
            {

                timer1.Stop();

            }
        }
        else if (slovo_2.ToString() == cetvrti.ToString())
        {
            if (y.ToString() == kat_3.ToString())
            {

                timer1.Stop();

            }
        }
        else if (slovo_2.ToString() == peti.ToString())
        {
            if (y.ToString() == kat_4.ToString())
            {

                timer1.Stop();

            }
        }
        if (slovo_3.ToString() == zatvorena.ToString())
        {
            vrata_picture.Image = Properties.Resources.vrata1;
            vrata.Text = "Zatvorena";
        }
        else if (slovo_3.ToString() == otvorena.ToString())
        {
            vrata_picture.Image = Properties.Resources.vrata2;
            vrata.Text = "Otvorena";
        }

    }
    private delegate void LineReceivedEvent(string tez);
    private void LineReceived(string tez)
    {
        prog_bar_tezina.Value = int.Parse(tez);
        tezina.Text = " " + tez.ToString();
        int percent = (int)(((double)(prog_bar_tezina.Value -
prog_bar_tezina.Minimum) /
      (double)(prog_bar_tezina.Maximum -
prog_bar_tezina.Minimum)) * 100);
        postotak.Text ="  " + percent.ToString();

        if (prog_bar_tezina.Value < prog_bar_tezina.Maximum)
        {
            tezina_text.Text = " Dozvoljena tezina";
            tezina_text.ForeColor = Color.Green;
```

```
            }
            if (prog_bar_tezina.Value ==
prog_bar_tezina.Maximum)
            {
                prog_bar_tezina.Value = prog_bar_tezina.Maximum;
                tezina_text.Text = " Prevelika tezina";
                tezina_text.ForeColor = Color.Red;

            }
        }

        private void tipkalo_prizemlje_Click(object sender,
EventArgs e)
        {
            serialPort1.Write("1");
        }

        private void tipkalo_kat_jedan_Click(object sender,
EventArgs e)
        {
            serialPort1.Write("2");
        }

        private void tipkalo_kat_dva_Click(object sender,
EventArgs e)
        {
            serialPort1.Write("3");

        }

        private void tipkalo_kat_tri_Click(object sender, EventArgs
e)
        {
            serialPort1.Write("4");

        }

        private void tipkalo_kat_cetiri_Click(object sender,
EventArgs e)
        {
            serialPort1.Write("5");
        }

        private void tipkalo_otvori_vrata_Click(object sender,
EventArgs e)
        {
            serialPort1.Write("O");
        }

        private void tipkalo_zatvori_vrata_Click(object sender,
EventArgs e)
        {
            serialPort1.Write("Z");
        }
    }

  }
```

Ovom prilikom želim se zahvaliti mentoru doc.dr.sc. Tomislavu Keseru na stručnoj pomoći i razumijevanju tijekom izrade diplomskog rada.

Posebna zahvala mojim roditeljima koji su mi omogućili ovaj studij, zahvaljujem im se na pomoći i strpljenju koje su imali tijekom mog studiranja.