Maketa dizala zasnovana na LED tehnologiji

Turkalj, Krešimir

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

Sveučilišni diplomski studij računarstva

MAKETA DIZALA ZASNOVANA NA LED TEHNOLOGIJI

Diplomski rad

Krešimir Turkalj

Osijek, 2020

SADRŽAJ

1. UVOD

Dizalo je neizostavan dio svakog nebodera jer smanjuje vrijeme potrebno za dolazak na traženi kat, pri čemu putnik ne obavlja zahtjevan fizički rad. Kako su neboderi rasli u visini, došlo je do potrebe uvođenja dužih dizala s većom pouzdanošću. Samim povećanjem udaljenosti koja je pokrivena jednim dizalom, povećao se broj potencijalnih putnika koji će koristiti dizalo u isto vrijeme i prosječan pređeni put, čime se odziv pojedinog dizala znatno usporava. U svrhu rješavanja navedenog problema, uvedeni su sustavi koji upravljaju paralelnim dizalima, kako bi se postigla veća učinkovitost. Ti sustavi mogu biti jako složeni, što je vidljivo u primjeru najviše zgrade u glavnom grada Ujedinjenih Arapskih Emirata, World Trade Center u Abu Dhabiju, koja je visoka 381.2 m, ima 88 katova iznad tla i 5 ispod, sadrži 13 dizala, pri čemu nijedno dizalo ne pokriva svih 93 katova [1].

Cilj ovog diplomskog rada je napraviti jednostavan sustav koji upravlja dvama dizalima, od kojih svaki od njih je prikazan s dva niza svjetlećih dioda. Rješenje za upravljanje maketom dizala je izvedeno u Arduino razvojnom okruženju, unutar kojeg postoji biblioteka za komunikaciju s nizom svjetlosnih dioda. Za upravljanje s dizalima postavljene su 24 tipke koje su raspodijeljene u 5 grupa, pri čemu svaka grupa ima drugačiju funkcionalnost. Te funkcionalnosti su dodjela zahtjeva lijevom ili desnom dizalu, dodjela zadataka s gornjim ili donjim usmjerenjem, te tipke za promjenu načina rada. Kako bi se upravljalo raspodjelom zadataka, za maketu dizala su napravljena četiri algoritma: algoritam prvog odziva, algoritam najmanje udaljenosti, algoritam najkraćeg odziva i algoritam s usmjerenjem.

Nakon uvodnog poglavlja opisana su postojeća rješenja i usporedbe s ovim rješenjem. Zatim je u trećem poglavlju opisano koji su elektrotehnički dijelovi korišteni u sklopovlju, kako su spojeni, koje razvojno okruženje i koje biblioteke su korištene za programsku podršku. Potom je u četvrtom poglavlju objašnjeno kako rade sva četiri algoritama. U petom poglavlju je prikazana procjena rada makete dizala i njenih osnovnih funkcija, te procjena rada svakog algoritma s navedenim prednostima i nedostacima. U šestom poglavlju je napisan zaključak.

2. POSTOJEĆA RJEŠENJA

U završnom radu „Maketa dizala s upravljanjem realiziranim u Arduinu“ [2] opisano je kako je izvedena fizička maketa dizala napravljenog u Arduino sustavu. Za prikaz rada, korištena je fizička kabina dizala s protutežom koja se kreće uz pomoć električnih motora i koja također ima pokretna vrata pogonjena električnim motorom.

U završnom radu „Upravljanje dizalom pomoću programirljivog logičkog kontrolera“ [3] opisana je izrada fizičke makete dizala s četiri kata, u kojoj se koristi upravljanje bez povratne veze. Za upravljanje pozicijom kabine korišten je PLC. Za upravljanje se koriste tri načina rada, ručni u kojem se držanjem tipke za podizanje i spuštanje izvodi željena akcija, automatski u kojem se izvodi predefinirane naredbe, te poluautomatski u kojem se korištenjem prilagođenog Karpovog algoritma upravlja sa zadacima dobivenim preko naredbi.

U diplomskom radu „Upravljanje dizalom“ [4] opisana je izrada fizičke makete dizala s četiri kata. Maketa koristi tipkala, senzore, pokretna vrata koja se pokreću servo motorima i svjetlosnu signalizaciju sa svjetlećim diodama i sedam segmentnim pokazivačem. Za potrebe prilagodbe signala korištena je prilagođena upravljačka ploča.

U znanstvenom radu „A Genetic Algorithm based elevator dispatching method for waiting time optimization.“ [5] opisano je korištenje genetičkog algoritma za učinkovito upravljanje grupom dizala u istom sustavu. Glavni kriteriji za odabir u generaciji su bili što kraće vrijeme vožnje i vrijeme čekanja dizala. Izvođenje algoritma je prikazano u simulaciji zgrade s 20 katova i četiri dizala.

U znanstvenom radu „Improving Elevator Performance Using Reinforcement Learning“ [6] opisano je kako se korištenjem strojnog učenja može poboljšati učinkovitost algoritma raspodjele zadataka unutar sustava dizala. Kako ponašanje ljudi unutar zgrade i mogući ishodi odziva u sustavu s više dizala su složeni, teško je napraviti sveobuhvatno rješenje. Za takve slučajeve se koristi strojno učenje. Strojno učenje je korišteno za algoritam raspodjele zadataka u simuliranoj zgradi s 10 katova i četiri dizala, s postavljenim konstantama za kretanje dizala, kapacitet kabine i vrijeme potrebno za ulazak i izlazak putnika.

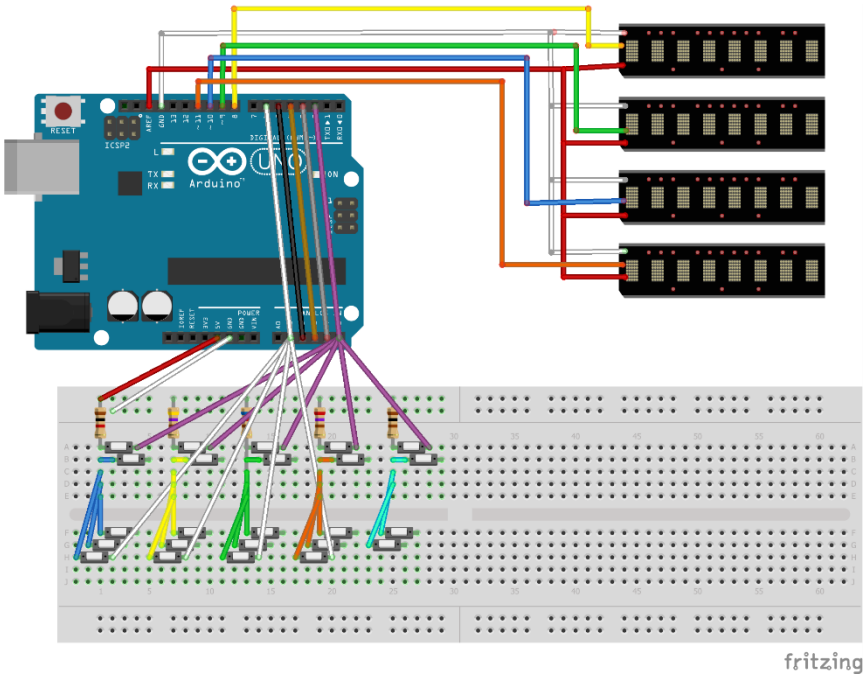
U ovom diplomskom radu cilj je bio na jednostavan način preko makete prikazati više različitih algoritama raspodjele zadataka i njihove razlike. Zbog prenosivosti i razumljivosti makete, broj katova je ograničen na pet. Kako na preciznost prikaza algoritma ne bi utjecala nepouzdanost mehaničkih dijelova, odlučeno je, za razliku od [2], [3] i [4] rada koji su koristili mehaničke makete dizala, prikazati kretanje dizala pomoću LED tehnologije.

3. MAKETA DIZALA

U nastavku će biti opisano kako je izvedena maketa dizala. Prvo je opisano kako su spojene komponente u modelu, zatim su opisani bitni elektrotehnički dijelovi koji su korišteni u maketi. Na kraju je je napisan opis razvojnog okruženja i korištenih biblioteka.

3.1. Sklopovski model dizala

Sklopovski model dizala sadrži Arduino razvojnu ploču na koju je spojeno uzemljenje i ulazni napon od 5V, četiri LED niza koji osim istog uzemljenja i ulaznog napona su također spojeni na digitalnu nožicu. Preko navedene nožice se prima signal za promjenu stanja niza dioda. Osim niza svjetlećih dioda, na Arduino ploču spojena je tiskana pločica čiji cilj je povezati tipkala kako bi se mogla razlikovati na analognoj nožici (slika 3.1.). Kako bi bilo moguće a-sinkrono očitati kad je pritisnuta tipka, svaka korištena analogna nožica je spojena na digitalnu nožicu, na kojoj se očitava prekid. Napajanje za sve dijelove se dobiva iz AC/DC pretvornika za napajanje, koji je spojen na Vcc i GND na tiskanoj pločici.



Slika 3.1 Prikaz spajanja tipkala na analogne i digitalne nožice

3.2. Sklopovlje

Maketa je izvedena korištenjem Arduino razvojne ploče na koju je spojeno 24 tipkala i 4 niza po 28 svjetlećih dioda. Kako Arduino Uno nema dovoljno digitalnih nožica, 24 tipkala treba spojiti na 5 analognih nožica pri čemu svako tipkalo treba dovoditi drugačiji napon na ulaz kako bi ih se moglo razlikovati. Različiti naponi se postignu tako da svako tipkalo u grupi jedne analogne nožice ima drugi omjer ulaznog i izlaznog otpornika. Izlazni otpornik je 1000Ω, dok ulazni ima jednu od 5 vrijednosti: 100Ω, 270Ω, 470Ω, 620Ω i 1000Ω. Svaka grupa otpornika predstavlja jednu funkciju, zapovijed odlaska za lijevo ili desno dizalo, zahtjev za određeni s gornjim ili donjim usmjerenjem, te odabir načina rada.

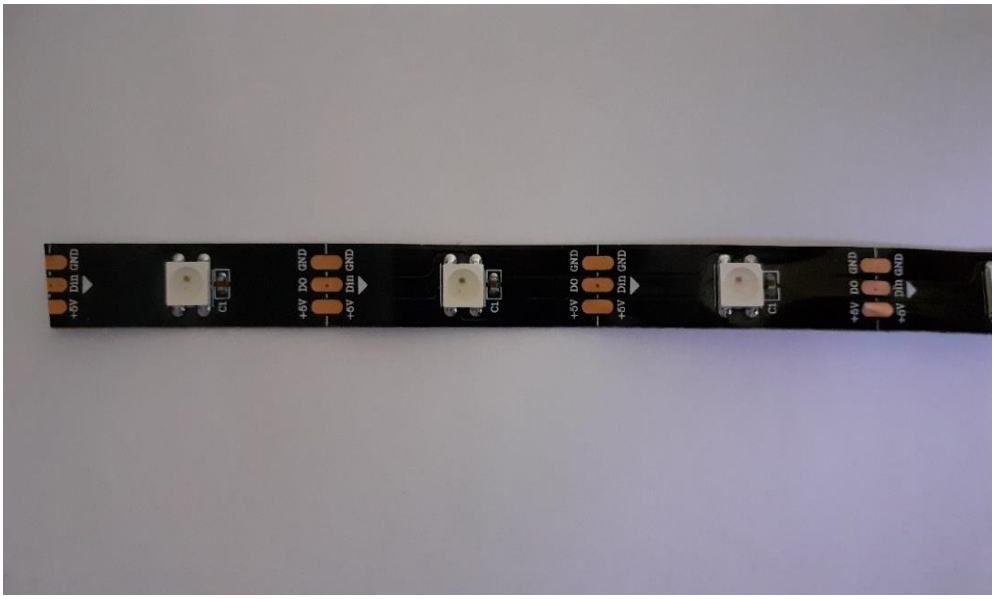
3.2.1. ATMega328P razvojni sustav

ATMega328P je mikro upravljač koji koristi Arduino Uno sklopovlje [7]. Arduino Uno koristi 8-bitni registar, ima 14 digitalnih i 6 analognih nožica koje se mogu koristiti za primanje ili slanje signala. Sadrži 32 KB *flash* memorije od kojih 2 KB se koristi za SRAM. Unutarnji EEPROM sadrži 1 KB memorije. Ima unutarnji keramički rezonator koji radi na frekvenciji od 16 MHz. Za pravilan rad preporuča se ulazni napon između 7 i 12 V. Arduino radi na naponu od 5V, te ima maksimalnu dopuštenu struju od 20 mA pri 5 V po nožici. Izgled Arduino sklopovlja je prikazan na slici 3.2.

***Slika** 3.2 Arduino Uno*

3.2.2. Niz svjetlećih dioda 5050-30LED/M

U ovom radu su korišteni nizovi svjetlećih dioda tipa 5050-30LED/M. 5050 u imenu opisuje dimenzije paketa u kojem se nalaze svjetleće diode, 50mm x 50mm [8]. 30LED/M ukazuje koliko ima paketa svjetlećih dioda u jednom metru, pri čemu su svi jednako udaljeni. Svaki paket sadrži 3 svjetleće diode, crvenu, zelenu i plavu. Svakoj se može zasebno namjestiti jačina svjetlosti i boja. Na niz svjetlećih dioda treba spojiti žice s naponom od 5V, uzemljenje, te žicu koja prenosi upravljački signal. Izgled spajanja nizova na digitalne nožice može se vidjeti na slici 3.1, dok izgled niza svjetlećih dioda je vidljiv na slici 3.3.



Slika 3.3 Izgled LE niza dioda 5050-30LED/M

3.3. Programska podrška za upravljanje radom makete

U sklopu ovog diplomskog rada je potrebno izraditi programsko rješenje koje će upravljati nizom svjetlećih dioda kako bi se prikazala trenutna pozicija i stanje dizala. Objekt dizala sadrži tri informacije, trenutnu poziciju, brzinu i stanje vrata. Trenutno stanje dizala se mijenja u funkciji `updateCurrentState(double interval)`, gdje interval predstavlja vrijeme proteklo od prošlog poziva, čime je postignuto praćenje stvarnog prolaska vremena. Trenutna pozicija dizala je prikazana na nizu svjetlećih dioda s četiri diode u stacionarnom stanju i s pet u pokretu. Kako se dizalo kreće, zadnja svjetleća dioda se postupno gasi, a dodatna peta se pali. Crvena boja predstavlja zatvorena vrata, zelena boja predstavlja otvorena vrata, a prijelaz predstavlja otvaranje ili zatvaranje vrata. U nastavku je opisano razvojno okruženje u kojem je

pisano programsko rješenje, zatim su opisane biblioteke korištene za upravljanje nizom svjetlećih dioda i postizanje prekidnog način rada, s ciljem smanjenja učestalosti čitanja vrijednosti analogne nožice. Kako bi se odredila raspodjela zadataka u sustavu dva dizala korištena su 4 algoritma raspodjele zadataka opisana u zadnjem dijelu.

3.3.1. Arduino razvojno okruženje

Arduino razvojno okruženje je program uz pomoć kojeg je napisan kod zapisan na odabrani Arduino uređaj. Može se koristiti C i C++ jezik, s time da C++ ima određene preinake radi jednostavnosti Arduino okruženja. Programski kod se sastoji od dijela koji se pokreće samo jednom kroz `setup()` funkciju i dijela koji se stalno ponavlja unutar `loop()` funkcije. Program napisan unutar Arduino razvojnog okruženja se zove *sketch*. Njegova najveća veličina ovisi o mikro upravljaču koji određuje veličinu prostora za programski kod i varijable.

3.3.2. Adafruit NeoPixel

`Adafruit NeoPixel` je biblioteka koja je namijenjena upravljanju adresabilnim svjetlećim diodama. Za rad s funkcijama biblioteke potrebno je klasi predati broj svjetlećih dioda kojima se upravlja, redni broj korištene digitalne nožice i tip korištenog niza na temelju postojećih konstanti. Prije korištenja je potrebno pozvati funkciju `begin()` koja pripremi objekt za rad. Za postavljanje boje nizu svjetlećih dioda koriste se funkcije `fill(...)` za dodjelu iste naredbe cijelom nizu ili njegovoj sekciji, te `setPixelColor(...)` za dodjelu boje svjetlećoj diodi na određenoj poziciji. Funkcija `fill(...)` kao argumente prima boju, početni redni broj svjetleće diode u nizu, te broj dioda kojim se želi upravljati nakon prve. Ako se ne preda nijedan argument funkciji, cijeli niz se ugasi, ekvivalentno funkciji `clear()`. Funkcija `setPixelColor(...)` prima poziciju svjetleće diode i traženu boju kojom će se obojati pozicija. Vrijednost tražene boje se dobije iz statičke funkcije `Color(uint8_t r, uint8_t g, uint8_t b)` kojoj se preda željena RGB vrijednost.

3.3.3. PinChangeInterrupt

Arduino razvojna ploča ima samo jednu nožicu koja preko sklopovlja određuje je li stigao uvjet za prekid. Ostale digitalne nožice mogu očitati prekid na programski način, zbog čega se poziva isti prekid za bilo koju nožicu unutar grupe od četiri nožice. Postoje tri grupe od četiri digitalne nožice, {2,3,4,5}, {6,7,8,9} i {10,11,12,13} [9]. `PinChangeInterrupt` biblioteka sadrži brzo rješenje koje razlikuje prekid za svaku digitalnu nožicu, neovisno o tome pripadaju li istoj

grupi. Za korištenje je prvo potrebno postaviti željenu nožicu u ulazni način rada s unutarnjim otpornikom pomoću naredbe `pinMode(uint8_t pin, INPUT_PULLUP)`. Poslije postavljanja nožice poziva se naredba `attachPinChangeInterrupt(...)` kojoj se prvo preda povratna vrijednost funkcije `digitalPinToPinChangeInterrupt(uint8_t pin)` kojoj se prosljeđuje redni broj digitalne nožice. Na drugo mjesto se stavlja naziv prekidne funkcije koja će se pozvati kada prekid bude zabilježen i kao zadnji argument se predaje konstanta koja predstavlja uvjet za koji će se pokrenuti prekidna rutina. Uvjet može biti niski, visoki, silazni i uzlazni signal.

4. ALGORITMI ZA RASPOREĐIVANJE ZADATAKA

Dizalo može dobiti poziv na određeni kat na dva načina. Prvi način je zapovijed za odabrano dizalo koje stigne od tipki unutar kabine dizala. Ta zapovijed se mora izvesti za određeno dizalo, to jest pritisak tipke unutar kabine jednog dizala ne utječe na drugo dizalo. Drugi način je zahtjev za dolazak na određeni kat. Ta funkcionalnost predstavlja pritisak na mjestu ulaska u dizalo, na što se oba dizala mogu odazvati. Za drugi način sustav treba odrediti kome treba dodijeliti poziv, pri čemu su korištena četiri različita algoritma dodjele zadataka, gdje svaki sljedeći algoritam nadopunjuje prethodni i povećava njegovu učinkovitost. U nastavku je opisan osnovni algoritam s najjednostavnijom provjerom. Cilj prvog algoritma je dodijeliti zadatak prvom slobodnom dizalu, pri čemu desna strana ima prednost. Zatim je opisan algoritam najmanje udaljenosti, unapređenje osnovnog algoritma s dodatnom provjerom. Cilj drugog algoritma je dodijeliti zadatak najbližem slobodnom dizalu. Potom je opisan algoritam najkraćeg odziva, čiji cilj je predati zadatak dizalu koje može prije doći na željeno odredište. I na kraju je opisan algoritam s usmjerenjem koji u algoritam najkraćeg odziva dodaje informaciju smjera u kojem putnik želi krenuti, čime se provodi procjena budućeg stanja.

4.1. Algoritam prvog odziva

Algoritam prvog odziva ili algoritam pričuve je osnovni algoritam koji dodjeljuje zadatak prvom slobodnom dizalu. S obzirom da provjera ide uvijek istim redoslijedom, postoji prioritet u odabiru desnog dizala. Prilikom dodjele zadataka potrebno je pratiti samo dva uvjeta, postoji li zadatak koji se može dodijeliti, te postoji li slobodno dizalo. Zbog svoje jednostavnosti, jako se brzo izvodi, no zato što glavno dizalo odrađuje i zahtjeve koje bi bili efikasnije odrađeni s drugim dizalom, sustav koji koristi ovaj algoritam pruža najmanju učinkovitost.

4.2. Algoritam najmanje udaljenosti

Algoritam najmanje udaljenosti je unaprjeđenje osnovnog algoritma s dodatnom provjerom koje je od trenutno slobodnih dizala bliže. U slučaju kada su jednako udaljeni, desno dizalo i dalje ima prednost, no s obzirom da to nije česti slučaj, ovaj algoritam puno bolje upošljava obje strane. U navedeni algoritam može se uz dodatne provjere ubaciti mogućnost dodavanja zadataka zauzetom dizalu. Te provjere će biti opisane u sljedećem poglavlju.

4.3. Algoritam najkraćeg odziva

Algoritam najkraćeg odziva ima cilj za svaki zadatak pronaći dizalo koje može prvo doći na traženu visinu. Provjera se izvršava na svim dizalima neovisno o njihovoj trenutnoj zaposlenosti. Prva provjera je za slučaj ako se zadatak može postaviti na prvo mjesto. Zadatak je moguće postaviti na prvo mjesto ako je dizalo slobodno ili ako zauzeto dizalo može usputno stati na traženo mjesto. Ako je moguće staviti zadatak na prvo mjesto, onda se koristi vrijeme potrebno do traženog mjesta. U suprotnom je potrebno izračunati vrijeme izvođenja svakog zadatka prije kojeg se ne može izvoditi traženi zadatak. Vrijeme izvođenja zadatka se izračuna kao zbroj vremena dolaska na traženu visinu, te vremena potrebnog za ciklus otvaranja i zatvaranja vrata. Uvjet za mogućnost postavljanja željenog zadatka prije trenutno promatranog, je kretanje u pravcu promatranog kata, te da dizalo nije prošlo točku iza koje se ne može zaustaviti na vrijeme (jednadžba (4-4)). Kako bi se izračunalo vrijeme odziva, korištene su tri osnovne jednadžbe vezane za udaljenost, vrijeme, brzinu i ubrzanje:

$$v = \frac{s}{t}, \tag{4-1}$$

$$v = at, \tag{4-2}$$

$$s = \frac{v^2}{2a}. \tag{4-3}$$

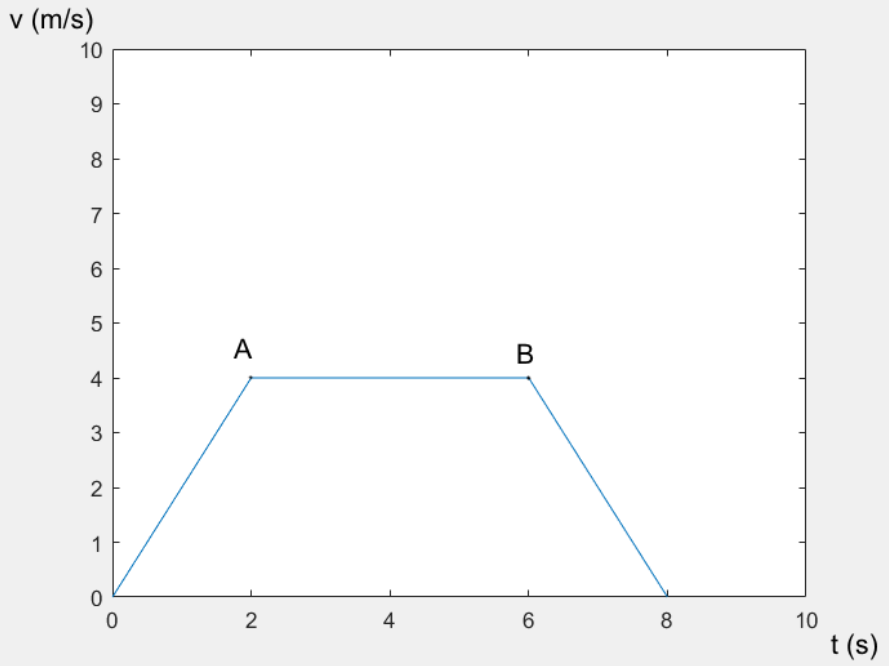
Drugi uvjet za mogućnost postavljanja je da trenutna udaljenost s_{tr} ne smije biti manja od udaljenosti s_a potrebne kako bi se zakočilo na vrijeme:

$$s_{tr} \geq s_a \tag{4-4}$$

Uvrštavanjem jednadžbe (4-3) na mjesto s_a u jednadžbu (4-4) dobije se izraz drugog uvjeta za mogućnost postavljanja zadatka u obliku jednadžbe (4-5).

$$s_{tr} \geq \frac{v^2}{2a} \tag{4-5}$$

Na slici 4.1. je prikazana ovisnost brzine o vremenu u slučaju kada se postigne najveća dopuštena brzina.



Slika 4.1 Promjena brzine dizala u ovisnosti o vremenu

Kako bi se odredilo vrijeme potrebno za dolazak, prvo se određuje koji dio grafa opisuje trenutno stanje dizala. Ako je dostupno dizalo na udaljenosti manjoj od potrebne za kočenje s najveće brzine, onda se nalazi iza točke B. Korištenjem jednadžbe (4-2), za navedeni dio grafa se izračuna vrijeme kočenja prema jednadžbi (4-6), pri čemu v je trenutna brzina:

$$t = \frac{v}{a} \tag{4-6}$$

Ako dizalo ne koči, treba provjeriti je li trenutna brzina dizala jednaka najvećoj, što je prikazano područjem na slici 4.1. između točki A i B. Potrebno je odrediti ukupno vrijeme kretanja, koje je jednako zbroju vremena kretanja t_{max} pri najvećoj brzini v_{max} i vremena kočenja t_a, prikazano jednadžbom (4-7):

$$t = t_a + t_{max} \tag{4-7}$$

Vrijeme i udaljenost pređena pri kočenju od najveće brzine je stalna, zbog čega korištenjem jednadžbi (4-2) i (4-3) se prvo izračunaju te dvije vrijednosti pomoću jednadžbi (4-8) i (4-9) :

$$t_a = \frac{v_{max}}{a} \tag{4-8}$$

$$s_a = \frac{{v_{max}}^2}{2a} \tag{4-9}$$

Put koji se pređe pri najvećoj brzini s_{max} je jednak rezultatu oduzimanja trenutne udaljenosti s_{tr} i putu pređenom pri kočenju s_a prikazano jednadžbom (4-10):

$$s_{max} = s_{tr} - s_a \tag{4-10}$$

Uvrštavanjem jednadžbe (4-9) na mjesto s_a u jednadžbi (4-10) dobije se konačan oblik koji prikazuje udaljenost pređenu pri najvećoj brzini (jednadžba (4-11)):

$$s_{max} = s_{tr} - \frac{{v_{max}}^2}{2a} \tag{4-11}$$

Vrijeme koje dizalo provede pri najvećoj brzini dobije se uvrštavanjem jednadžbe (4-11) na mjesto s u jednadžbi (4-1), s čime se dobije jednadžba (4-12):

$$t_{max} = \frac{s_{tr} - \frac{{v_{max}}^2}{2a}}{v_{max}} \tag{4-12}$$

Uvrštavanjem jednadžbi (4-8) i (4-12) u jednadžbu (4-7) dobije se jednadžba (4-13), koja prikazuje potrebno vrijeme do tražene točke:

$$t = \frac{v_{max}}{a} + \frac{s_{tr} - \frac{{v_{max}}^2}{2a}}{v_{max}} \tag{4-13}$$

Nakon sređivanja dobije se jednadžba (4-14):

$$t = \frac{v_{max}}{2a} + \frac{s_{tr}}{v_{max}} \tag{4-14}$$

Ako prve dvije provjere nisu prošle, dizalo je u stanju prije točka A na slici 4.1. Za područje na slici ispred točke A potrebno je odrediti može li se postići najveća dopuštena brzina. Uvjet je ispunjen ako je udaljenost do kata jednaka ili veća zbroju udaljenosti potrebnoj za ubrzanje do najveće brzine s_a i udaljenosti potrebnoj za usporenje od najveće brzine s_{da} prikazano jednadžbom (4-15).

$$s_{tr} \geq s_a + s_{da} \tag{4-15}$$

Udaljenost s_{da} je poznata iz jednadžbe (4-9), dok udaljenost s_a je jednaka rezultatu oduzimanja s_{da} i udaljenosti potrebnoj za ubrzanje do trenutne brzine v. Udaljenost potrebna za ubrzanje je prikazana jednadžbom (4-16):

$$s_a = \frac{{v_{max}}^2}{2a} - \frac{v^2}{2a} \tag{4-16}$$

Nakon što se u jednadžbu (4-15) uvrsti (4-7) i (4-16), dobije se jednadžba (4-17):

$$s \geq \frac{{v_{max}}^2}{2a} + \frac{{v_{max}}^2}{2a} - \frac{v^2}{2a} \tag{4-17}$$

Nakon sređivanja jednadžbe (4-17), dobije se izraz provjere za mogućnost postizanja najveće brzine (jednadžba (4-18)):

$$s \geq \frac{{v_{max}}^2}{a} - \frac{v^2}{2a} \tag{4-18}$$

Ako uvjet u jednadžbi (4-18) nije ispunjen, nije moguće postići najveću brzinu. U tom slučaju potrebno vrijeme jednako vremenu akceleracije i deceleracije do najveće moguće brzine. Na najveću moguću brzinu v' utječe udaljenost potrebna za ubrzanje do najveće moguće brzine $s_{a'}$, udaljenosti potrebne za usporenje od najveće moguće brzine $s_{da'}$ i udaljenost pređena pri ubrzanju do trenutne brzine s_a. Zbog jednake vrijednosti akceleracije, moguće je analogno jednadžbi (4-16) prikazati put pređen pri ubrzanju v' kao rezultat oduzimanja $s_{da'}$ *i* s_a (jednadžba (4-20) i jednadžba):

$$s_{a'} = s_{da'} - s_a \tag{4-19}$$

$$s_{a'} = \frac{{v'}^2}{2a} - \frac{v^2}{2a} \tag{4-20}$$

Ukupna udaljenost je jednaka vremenu akceleracije i deceleracije do najveće moguće brzine, prikazano jednadžbom (4-21):

$$s_{tr} = s_{da'} + s_{a'} \tag{4-21}$$

Uvrštavanjem jednadžbi (4-6) i (4-20) u jednadžbu (4-21) dobije se jednadžba (4-22) u kojoj je nepoznanica najveća moguća brzina v'.

$$s_{tr} = \frac{{v'}^2}{2a} + \frac{{v'}^2}{2a} - \frac{v^2}{2a} \tag{4-22}$$

Sređivanjem jednadžbe (4-21) s ciljem izražavanja nepoznanice v' dobije se jednadžba (4-23):

$$v' = \sqrt{\mathrm{as} - \frac{v^2}{2}} \tag{4-23}$$

Potrebno vrijeme, analogno jednadžbi za udaljenost (4-20), je jednako zbroju vremena za $t_{a'}$ akceleraciju i $t_{da'}$ deceleraciju, pri čemu akceleraciju, analogno jednadžbi (4-20) se može zapisati kao rezultat oduzimanja $t_{da'}$ i t_a vremena potrebnog za ubrzanje do trenutne brzine (jednadžba (4-24)):

$$t = t_{da'} + t_{da'} - t_a \tag{4-24}$$

Korištenjem jednadžbe (4-6) za zamjenu vrijednosti $t_{da'}$ i t_a, pri čemu kod $t_{da'}$ vrijednost brzine je v', dobije se jednadžba (4-25):

$$t = \frac{v'}{a} + \frac{v'}{a} - \frac{v}{a} \tag{4-25}$$

Dodatnim sređivanjem jednadžbe (4-25) i ubacivanjem jednadžbe (4-23) za v', dobije se jednadžba (4-26) potrebnog vremena:

$$t = \frac{2\sqrt{\mathrm{as} - \frac{v^2}{2}}}{a} - \frac{v}{a} \tag{4-26}$$

Ako je moguće postići najveću brzinu, potrebno je izračunati vrijeme ubrzanja do najveće brzine, vrijeme provedeno pri najvećoj brzini, te vrijeme kočenja. Kako bi se izračunalo vrijeme provedeno pri najvećoj brzini, prvo je potrebno odrediti kolika je udaljenost pređena pri najvećoj brzini. Ukupni pređeni put je jednak zbroju udaljenosti pređenih pri ubrzanju s_a, kočenju s_{da} i najvećoj brzini s_{max} (jednadžba (4-27)):

$$s = s_a + s_{da} + s_{max} \tag{4-27}$$

Uvrštavanjem jednadžbi (4-9) i (4-16) u jednadžbu (4-27), dobije se jednadžba (4-28) u kojoj je moguće izraziti nepoznanicu s_{max}:

$$s_{max} = s - \frac{{v_{max}}^2 - v^2}{2a} - \frac{{v_{max}}^2}{2a} \tag{4-28}$$

Uređivanjem jednadžbe (4-28) dobije se jednadžba (4-29) za izračun puta pri najvećoj brzini:

$$s_{max} = s + \frac{v^2 - 2{v_{max}}^2}{2a} \tag{4-29}$$

Uvrštavanjem jednadžbe (4-29) u jednadžbu (4-1), može se izračunati vrijeme provedeno pri najvećoj brzini t_{max} kao jednadžba (4-30), pri čemu brzina je v_{max}:

$$t_{max} = \frac{s + \frac{v^2 - 2{v_{max}}^2}{2a}}{v_{max}} \tag{4-30}$$

Sveukupno vrijeme je jednako zbroju vremena potrebnog za ubrzanje t_a , kočenje t_{da} i vremenu t_{max} (jednadžba (4-31)):

$$t = t_a + t_{da} + t_{max} \tag{4-31}$$

Analogno jednadžbi (4-25), t_a i t_{da} se mogu napisati kao $t_{a'}$ i $t_{da'}$, pri čemu se v' zamjeni s v_{max}, a umjesto t_{max} se uvrsti jednadžba (4-31), čime se dobije jednadžba (4-32):

$$t = \frac{v_{max}}{a} - \frac{v}{a} + \frac{v_{max}}{a} + \frac{s + \frac{v^2 - 2{v_{max}}^2}{2a}}{v_{max}} \tag{4-32}$$

Sređivanjem jednadžbe (4-32) dobije se jednadžba (4-33):

$$t = \frac{2as + (v_{max} - v)^2 + {v_{max}}^2}{2a v_{max}} \tag{4-33}$$

Ako nije moguće staviti zadatak na prvo mjesto, iduća točka određivanja kreće u trenutku u kojem dizalo stoji. Taj slučaj je podskup gore navedenih gdje se za trenutnu brzinu stavlja vrijednost 0. Za slučaj kada nije moguće postići najveću brzinu u jednadžbi (4-26) dobije se jednadžba (4-34):

$$t = 2\sqrt{\frac{s}{a}} \tag{4-34}$$

A u slučaju kada se postigne najveća brzina prikazano jednadžbom (4-33), postavljanjem vrijednosti v u 0 dobije se jednadžba (4-35):

$$t = \frac{2as + {v_{max}}^2 + {v_{max}}^2}{2a v_{max}} \tag{4-35}$$

Sređivanjem jednadžbe (4-35) dobije se jednostavnija jednadžba (4-36):

$$t = \frac{s}{v_{max}} + \frac{v_{max}}{a} \tag{4-36}$$

Vrijeme se zbraja dok god nije moguće smjestiti zadatak između druga dva zadatka ili je došao kraj listi zadataka. Potom slijedi provjera najkraćeg vremena i dodjele zadatka dizalu s bržim odzivom. Kako ne bi bilo potrebno ponovo tražiti povoljnu poziciju, pozicija zadatka za koju je dobiveno vrijeme prosljeđuje se zajedno s izračunatim vremenom.

4.4. Algoritam s usmjerenjem

Posebnost algoritma s usmjerenjem naspram dosadašnjih algoritama je procjena budućeg stanja. Algoritmi najmanje udaljenosti i najkraćeg odziva su davali prednost na temelju informacija do zadanog kata, dok zadnji algoritam još koristi dodatnu informaciju u kojem smjeru se očekuje zahtjev poslije odlaska na određeni kat. Kao i u algoritmu najkraćeg odziva, prvo se odredi potrebno vrijeme odlaska na određeni kat. Zatim se koristi informacija usmjerenja i određuje se dodatno očekivano vrijeme za svaki kat koji se nalazi u određenom smjeru. Na primjer, ako je na drugom katu pritisnuta tipka gornjeg usmjerenja, dodaje se vrijeme potrebno za svaki kat zasebno koji se nalazi iznad drugog kata, što su treći, četvrti i peti. Nakon toga kao i u prijašnjem algoritmu slijedi provjera najkraćeg vremena i ubacivanje na željeno mjesto. Određivanje budućeg potrebnog vremena izračunava se na isti način kao i u trećem algoritmu, pri čemu se kat s kojeg je poslan zahtjev gleda kao početna točka, a mogući katovi kao konačne točke.

5. EVALUACIJA

U nastavku je prikazan izgled makete, provjera prikaza dizala i dodjela zadataka unutar kabine, i na samom kraju su prikazani osnovni slučajevi koji će demonstrirati razlike između predloženih algoritma.

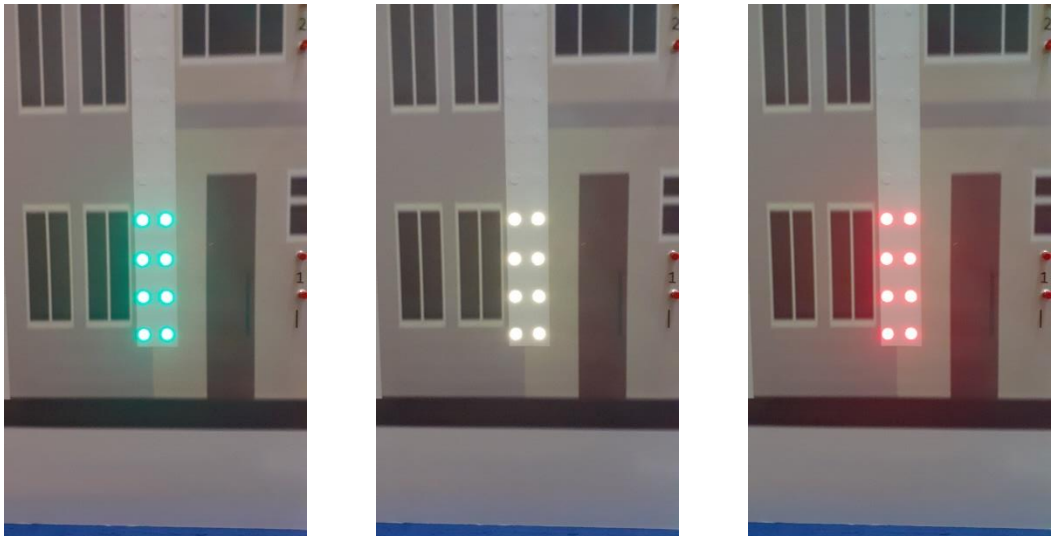
5.1. Evaluacija rada makete

Dizalo je prikazano na ploči sa slikom zgrade i dvije kabine dizala. Kretanje dizala se izvodi unutar bijele trake koje propuštaju svjetlost. Izgled ploče je vidljiv na slici 5.1.

***Slika** 5.1 Prikaz prednje strane makete*

S desne bočne strane su postavljena tipkala za odabir između četiri načina rada koja će biti korištena u sljedećem poglavlju.

Kada se pošalje zahtjev dizalu za kat na kojem se već nalazi, odabrano dizalo će otvoriti vrata. Pri svakom dolasku vrata se otvore, drže otvorenima kratko vrijeme, te se onda zatvore. Trajanje navedenih akcija je određeno konstantama `DOOR_SLIDE_TIME` i `DOOR_STOP_TIME`, pri čemu njihove vrijednosti predstavljaju sekunde. Prijelaz iz zelene u crvenu boju, odnosno zatvaranje vrata, je prikazano na slici 5.2.



5.2 a) *5.2 b)* *5.2 c)*

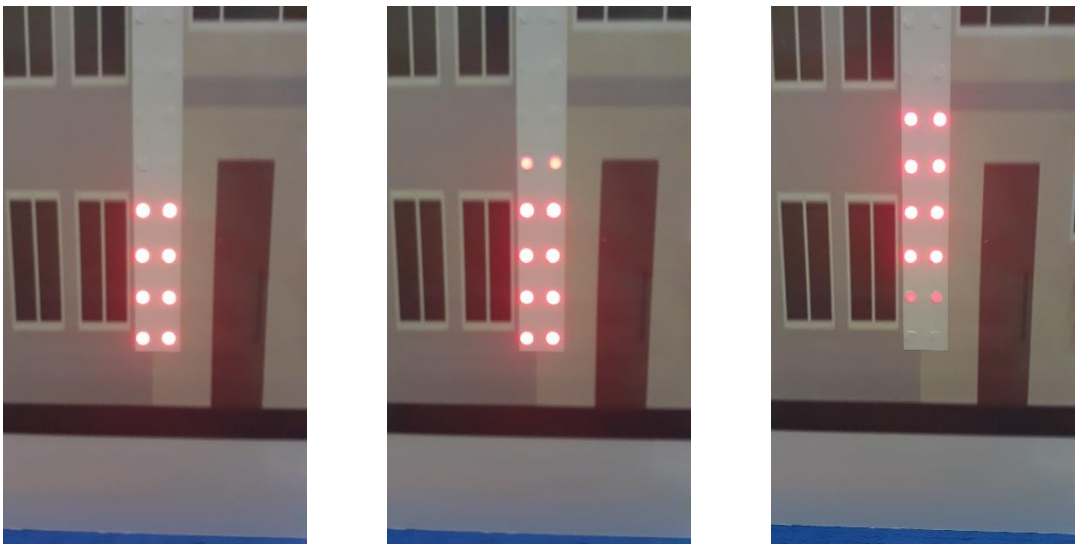
Slika 5.2 Zatvaranje vrata

Pri svakom pozivanju funkcije `sendDataToStrip(…)` prosljeđuje se trenutna pozicija i trenutno stanje vrata i duljina kabine dizala. Efekt pomicanja vrata se postiže postavljanjem boje svake točke uz pomoć programskog koda 5.1:

```
#define DOOR_SLIDE 255*(1-doorPosition), 255*doorPosition, 0
for(int j = 0; j < elevatorLength; j++){
  ledStrips[i].setPixelColor(lowestPosition+j,Adafruit_NeoPixel::Color(DOOR_SLIDE));
}
```

Programski kod 5.1 Učinak zatvaranja vrata

Kada stigne zahtjev za kat na kojem se trenutno ne nalazi, dizalo krene ubrzavati i pri dolasku usporava. Radi glađeg prijelaza zadnja točka u nizu se lagano gasi, dok iduća u redu se lagano pali. Pomicanje dizala prema gore je prikazano na slici 5.3.



5.3 a) *5.3 b)* *5.3 c)*

Slika 5.3 Kretanje dizala prema gore

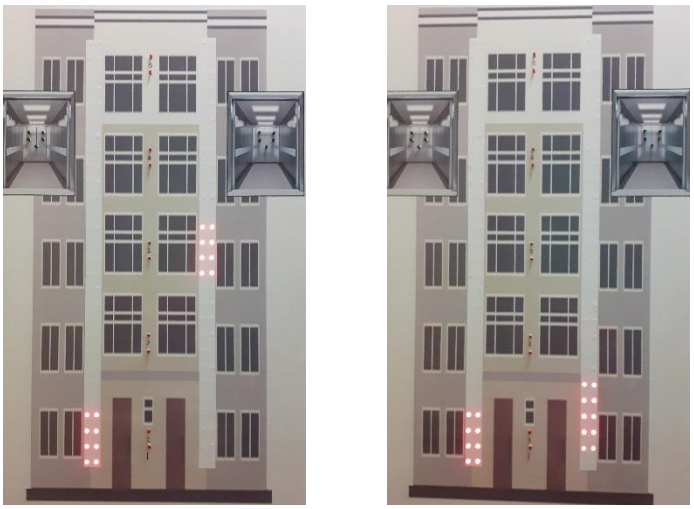
Klasa `ElevatorData` koja sadrži varijablu trenutne pozicije zapisana je kao *`double`* vrijednost. Kako su RGB vrijednosti u `Adafruit_NeoPixel::Color()` cjelobrojne, potrebno je transformacijom iz decimalne vrijednosti dobiti jačinu svjetlosti prve i zadnje LED. Prvo se odvoji decimalna vrijednost, podijeli s $\frac{1000}{256}$, a zatim se množi s 1000 kako se ne bi izgubila vrijednosti prelaskom u cjelobrojni tip. Postupak je prikazan u programskom kodu 5.2.

```
#define MOVE_COLOR_LAST   decimalPart, 0, 0
#define MOVE_COLOR        255, 0, 0
#define MOVE_COLOR_FIRST  255-decimalPart, 0, 0
...
else{
  int intPart = (int)lowestPosition;
  int decimalPart = 1000 * (double)((lowestPosition - intPart)/3.91);//<0,1> <0,255]
  ledStrips[i].setPixelColor(intPart, Adafruit_NeoPixel::Color(MOVE_COLOR_FIRST));
  for(int j = 1; j < elLength; j++){
    ledStrips[i].setPixelColor(intPart+j, Adafruit_NeoPixel::Color(MOVE_COLOR));
  }
ledStrips[i].setPixelColor(intPart+elLength,Adafruit_NeoPixel::Color(MOVE_COLOR_LAST));
}
```

Programski kod 5.2 Učinak pomicanja dizala

5.2. Evaluacija rada algoritama za raspoređivanje zadataka

U radu su korištena četiri algoritma odabira dizala. Prvi prikazani algoritam je algoritam prvog odziva u kojem se uzima prvo slobodno dizalo. Provjera se uvijek izvodi istim redoslijedom, zbog čega je najmanje učinkovit algoritam s obzirom na prosječni pređeni put. Prioritet desne strane je vidljiv na slikama 5.4 a) i 5.4 b), gdje je pritisnuta tipka na prvom katu, a javlja se desna strana.



5.4 a) *5.4 b)*

Slika 5.4 Prikaz dodjele prvog algoritma

Drugi algoritam, algoritam najmanje udaljenosti, odabire trenutno najbliže slobodno dizalo. Za isti slučaj kao u gornjem zadatku (Slika 5.4) sada se javlja lijeva strana prikazano na slici 5.5.

5.5 a) *5.5 b)*

Slika 5.5 Prikaz rada drugog algoritma

Potreba za slobodnim dizalom je vidljiva na slici 5.6., gdje je desno dizalo prije stiglo na prvi kat, ali pošto je zauzeto u trenutku pritiska, ne uzima se u obzir.

5.6 a) *5.6 b)* *5.6 c)*

Slika 5.6 Prikaz rada drugog algoritma - nedostatak

Treći algoritam, algoritam najkraćeg odziva, određuje vrijeme potrebno do određenog kata, uzimajući u obzir mogućnost zaustavljanja i ubacivanja zadatka. Zbog svoje mogućnosti dodavanja zadatka zauzetom dizalu, prethodni slučaj (Slika 5.6.) može se izvesti brže, što je vidljivo na slici 5.7. u kojoj se koristi lijeva strana.

5.7 a) *5.7 b)*

Slika 5.7 Prikaz rada trećeg algoritma

Na slici 5.8. prikazan je slučaj u kojem se na mjesto prvog zadatka stavi novi zadatak. Nakon kretanja dizala prema petom katu, pritisnuta je tipka na četvrtom katu gdje se zaustavlja, nakon čega nastavlja prema petom.

5.8 a) *5.8 b)*

Slika 5.8 Prikaz ubacivanja zadatka

Algoritam s usmjerenjem određuje vrijeme do kata na isti način kao i prijašnji algoritam, pri čemu još ima dodatak procjene vremena potrebnog za buduće zahtjeve. Prednost je vidljiva na slici 5.9., gdje je prikazan slučaj u kojem lijevo dizalo ima zadatak prvi kat, a desno peti kat. Iako lijevo dizalo prije stigne do traženog četvrtog kata, dio koji je isti trećem i četvrtom algoritmu, pošto je budući zadatak gornjeg usmjerenja duže će se izvoditi, zbog čega je zadatak prepušten desnoj strani.

5.9 a) *5.9 b)* *5.9 c)*

Slika 5.9 Prikaz rada četvrtog algoritma

6. ZAKLJUČAK

U ovom radu detaljno je opisana izrada makete dizala i procjena rada predstavljenih algoritama. Na početku je razmatrana mogućnost izrade fizičke makete dizala, no nakon kratkog razmatranja zaključeno je kako ispravnost fizičke makete donosi nepotrebne probleme s pouzdanošću, zbog čega je prihvaćena ideja prikaza dizala preko LED niza.

Za izradu makete dizala korišteni su sljedeći dijelovi : mikroupravljač Arduino Uno, 24 tipkala podijeljenih u 5 različitih grupa s drugačijim funkcionalnostima, otpornici za raspoznavanje signala na analognoj nožici, te tiskana pločica na kojoj je sve spojeno.

Programska podrška je napisana u Arduino razvojnom okruženju. Arduino razvojno okruženje sadrži biblioteku `Adafruit_NeoPixel` u kojoj je izvedeno rješenje za upravljanje nizovima svjetlećih dioda. Za asinkroni način rada upotrijebljena je biblioteka `PinChangeInterrupt`.

Kako bi se naglasila potreba za učinkovitijim upravljanjem dizala napisana su četiri algoritma, svaki napredniji od prijašnjeg. Zadnji algoritam koristi funkcionalnost dvije tipke kojima se određuje željeni smjer kretanja dizalom. Sva četiri algoritma imaju mogućnost rada s jednim ili više dizala, čime je omogućena proširivost sustava. Postoji mogućnost prikazivanja dizala sa samo jednom trakom, zbog čega je moguće uz manje promjene na maketi prikazati sustav s četiri dizala.

U budućnosti će biti ubačeno postavljanje liste zadataka i promjena vrijednosti u stvarnom vremenu za konstante koje opisuju sustav, kao što su najveća brzina i akceleracija. Navedene funkcionalnosti će biti proslijeđene putem serijske komunikacije. Također je razmotreno dodavanje prozirne plastike kako bi se zamaglio prikaz dva stupca svjetlećih dioda, s ciljem sakrivanja dva odvojena stupca za prikaz dizala.

LITERATURA

[1] The Skyscraper Center, https://www.skyscrapercenter.com/complex/282 [25.9.2020]

[2] Bulić, M. „Maketa dizala s upravljanjem realiziranim u Arduinu“, Završni rad, Sveučilište u Rijeci, Tehnički fakultet ,rujan 2017., https://urn.nsk.hr/urn:nbn:hr:190:962086 [25.9.2020]

[3] V. Šimundić, "Upravljanje dizalom pomoću programirljivog logičkog kontrolera", Završni rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, rujan 2019. https://urn.nsk.hr/urn:nbn:hr:200:127098 [25.9.2020]

[4] B. Rakazović, Upravljanje dizalom, Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, srpanj 2014. https://www.bib.irb.hr/830414?rad=830414 [25.9.2020]

[5] E. O. Tartan, C. Ciftlikli. "A Genetic Algorithm based elevator dispatching method for waiting time optimization." *IFAC-PapersOnLine* , (no. 49.3), (pp. 424-429), siječanj 2016., https://doi.org/10.1016/j.ifacol.2016.07.071 [25.9.2020]

[6] R.H. Crites i A.G. Barto, „Improving elevator performance using reinforcement learning“, *Advances in neural information processing systems* ,(pp. 1017-1023)., 1996., http://papers.nips.cc/paper/1073-improving-elevator-performance-using-reinforcement-learning.pdf [25.9.2020]

[7] Arduino Store, https://store.arduino.cc/arduino-uno-rev3 [25.9.2020]

[8] Flexfire LEDs, https://www.flexfireleds.com/comparison-between-3528-leds-and-5050-leds/ [25.9.2020]

[9] Arduino StackExchange, https://arduino.stackexchange.com/questions/1784/how-many-interrupt-pins-can-an-uno-handle [25.9.2020]

SAŽETAK

U ovom diplomskom radu detaljno je opisano kako je napravljen sustav dizala prikazanih preko LED niza. Maketa se sastoji od 4 LED niza, pri čemu se dva niza koriste za prikaz jednog dizala. Za upravljanje dizalima koriste se tipke raspodijeljene po funkcionalnosti u pet različitih grupa. Za podjelu neovisnih zadataka napisana su četiri algoritma, algoritam prvog odziva, algoritam najmanje udaljenosti, algoritam najkraćeg odziva, te algoritam s usmjerenjem.

Ključne riječi: Arduino, niz svjetlećih dioda, model dizala, algoritmi za raspoređivanje zadataka

ABSTRACT

This paper describes in detail process of creating elevator system presented through LED strips. Model consists of 4 LED strips in which pair of LED strips are used to represent one elevator. Managing elevators is done through buttons that are separated by function, of which there are five. Four different algorithms were written for the purpose of assigning tasks, First Response Algorithm, Shortest Distance Algorithm, Quickest Response Algorithm and Algorithm with Orientation.

Keywords: Arduino, LED strip, elevator model, algorithms for managing tasks

ŽIVOTOPIS

Krešimir Turkalj rođen je u Vinkovcima, Republika Hrvatska 7. veljače 1996. godine. Pohađao je osnovnu školu "Ivan Mažuranić" u Vinkovcima. U sedmom i osmom razredu sudjeluje na državnom natjecanju iz kemije. Nakon osnovne škole, 2011. godine upisuje Gimnaziju "Matije Antuna Reljkovića". U prvom razredu sudjeluje na državnom natjecanju iz kemije. Maturira 2015. godine i iste godine upisuje Elektrotehnički fakultet u Osijeku, smjer računarstvo, kojeg završava 2018. godine., čime stječe zvanje prvostupnik inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij računarstva kojeg trenutno pohađa. Na drugoj godini diplomskog studija radi kao demonstrator na laboratorijskim vježbama u sklopu kolegija Dizajn Računalnih Sustava.

Vlastoručni potpis

_____

Krešimir Turkalj