Pneumatska preša upravljana putem WEB-a

Dabčević, Zvonimir

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:482589

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-03**

Repository / Repozitorij:

Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Zvonimir Dabčević

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Željko Šitum, dipl. ing.

Student:

Zvonimir Dabčević

Zagreb, 2020.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Prije svega, želio bih se zahvaliti asistentu Juraju Beniću, mag. ing. mech. na svim korisnim savjetima, odvojenom vremenu i pruženoj pomoći tijekom izrade rada.

Također, zahvaljujem se svojem bivšem mentoru dr. sc. Tihomiru Žiliću, dipl. ing. na ugodnoj suradnji tokom studija, kao i prof. dr. sc. Željku Šitumu na prihvaćanju mentorstva te uvijek prisutnoj spremnosti na pomoć.

Naposljetku, zahvaljujem se svojoj obitelji i prijateljima na pruženoj potpori tijekom preddiplomskog studija, posebice nećakinjama Alice i Adriani na velikoj dozi pozitivne energije.

Zvonimir Dabčević

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **ZVONIMIR DABČEVIĆ** Mat. br.: 0035202218

Naslov rada na hrvatskom jeziku: **PNEUMATSKA PREŠA UPRAVLJANA PUTEM WEB-a**

Naslov rada na engleskom jeziku: **WEB CONTROLLED PNEUMATIC PRESS**

Opis zadatka:

Pneumatska preša izrađena u Laboratoriju za automatiku i robotiku FSB-a ima mogućnost spajanja na internet preko mikrokontrolera. Mikrokontroler je povezan s konverterom naponskog u strujni signal te s tlačnim ventilom koji regulira tlak u komorama cilindra. Na isti mikrokontroler spojen je senzor sile, a bit će postavljen i senzor pomaka. Pneumatska preša tlači metalnu oprugu koja služi kao opteretna sila. U mikrokontroleru treba napisati program za akviziciju podataka sa senzora, prikupljanje podataka s Web portala, izračun upravljačke veličine i njezino slanje na konverter, kao i slanje povratnih informacija korisniku na Web portal. Za programiranje mikrokontrolera potrebno je koristiti C++ programski jezik, a za Web portal koristiti web programske jezike uključivo Python i Web2Py.

U radu je potrebno:

- osmisliti WEB portal za korištenje pneumatske preše u koji će se korisnici ulogirati i upravljati s prešom (korisnički kalendar + WEB upravljačka stranica),
- izraditi elektroničku pločicu za smještaj senzora sile i pomaka preše s ulazno/izlaznim priključcima, kao i priključkom za njezino umetanje između Arduino pločice i Arduino shield pločice,
- matematički modelirati pneumatsku prešu i realizirati PID regulaciju sile i pomaka, napisati upravljački program za Arduino mikrokontroler u C++ programskom jeziku te omogućiti korisniku da preko Web portala odabire zadatak regulacije sile ili pomaka, kao i referentne vrijednosti sile ili pomaka,
- implementirati neuronsku mrežu s ciljem prepoznavanja govora pomoću Python TensorFlow modula, tako da se govorom može zadavati referentna vrijednost sile ili pomaka.

Zadatak zadan:
28. studenog 2019.

Datum predaje rada:
1. rok: 21. veljače 2020.
2. rok (izvanredni): 1. srpnja 2020.
3. rok: 17. rujna 2020.

Predviđeni datumi obrane:
1. rok: 24.2. – 28.2.2020.
2. rok (izvanredni): 3.7.2020.
3. rok: 21.9. - 25.9.2020.

Zadatak zadao:

Prof. dr. sc. Željko Šitum

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Opis
K_p	pojačanje proporcionalnog djelovanja
$e(t)$	signal greške
T_i	integracijsko vrijeme
T_d	derivacijsko vrijeme
$u(t)$	upravljački signal
$s(t)$	vremska domena
$S(\omega)$	frekvencijska domena
f_s	frekvencija uzorkovanja
f_0	Maksimalna frekvencija kontinuiranog signala
y	Izlazn neurona
$\hat{y}$	trenutačni izlaz neurona
θ	prag
w	težina
z	skup ulaznih signala

SAŽETAK

Što je to pametna pneumatska preša i što je potrebno za implementaciju takvog sustava? Koje su prednosti, a koji nedostaci interneta stvari? Odgovore na ta pitanja moguće je pronaći u ovom radu. Mrežno programiranje, implementacija PID regulatora, razvijanje sustava klasifikacije riječi primjenom neuronske mreže te u konačnici izrada same WEB aplikacije područja su koja ovaj rad obuhvaća.

Ključne riječi: internet stvari, pneumatska preša, mrežno programiranje, PID regulacija, neuronska mreža, WEB programiranje

SUMMARY

What steps are involved in creating a smart pneumatic press system? What are advantages and disadvantages of Internet of things? The answers to those questions are presented in this paper which contains the following topics: socket programming, PID controller implementation, word classification using neural networks and finally WEB application development.

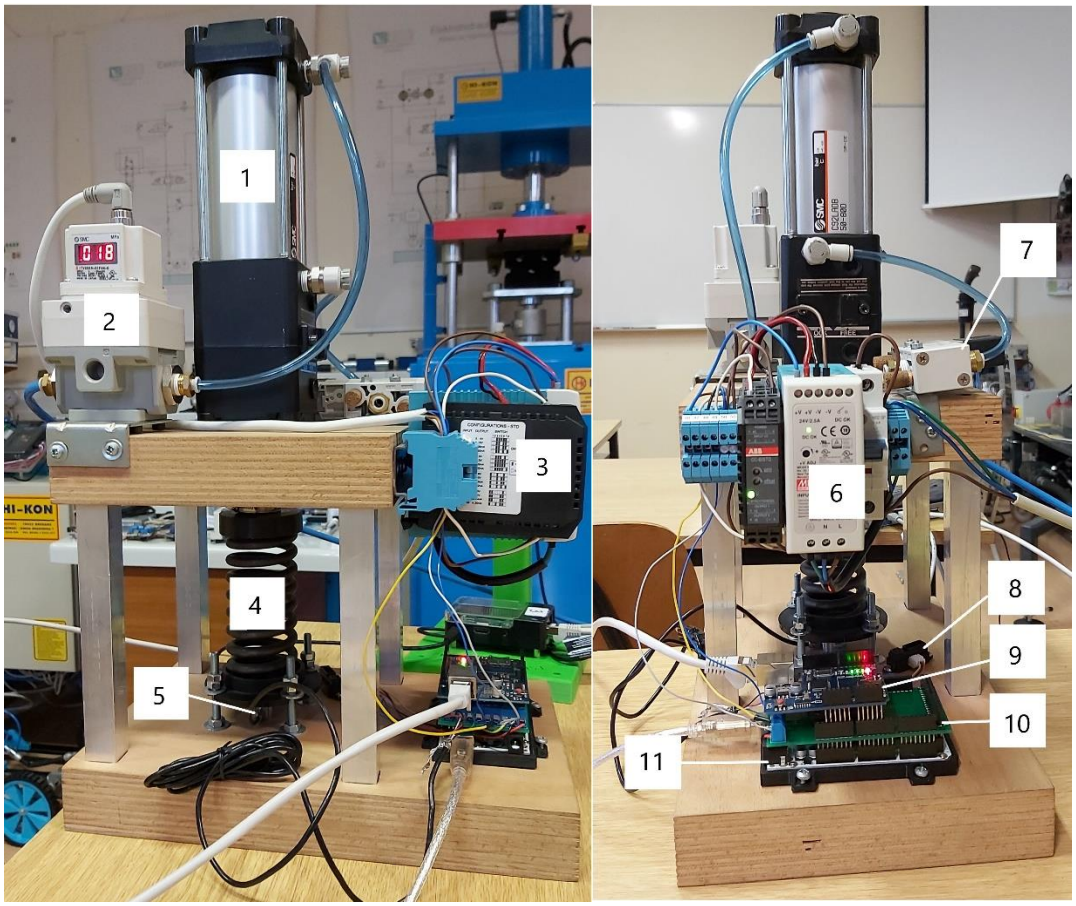
Key words: Internet of things, pneumatic press, socket programming, PID control, neural networks, WEB development

1. UVOD

Internet više ne uključuje samo računala. Danas se na internet mogu povezati različiti uređaji koje svakodnevno koristimo uključujući automobile, lampe, hladnjake, pa čak i žlice. Ovakav sustav međusobne povezanosti različitih uređaja poznat je pod nazivom internet stvari (engl. Internet of things). Međutim, nije li malo apsurdno povezati žlicu na internet? Žlica može prikupljati podatke poput koliko često jedemo, a dobivene informacije možemo pregledati pak pomoću drugog uređaja povezanog na internet kao što je pametni telefon. Svi podaci prikupljeni pomoću žlice mogu biti analizirani kako bismo saznali koliko kalorija smo unijeli u organizam, dobili osobne zdravstvene savjete i mnogo više. Ali, internet stvari nije ograničen na proizvode namijenjene potrošačima. Primjerice u gradovima postoje kante za smeće koje obavještavaju kada trebaju biti ispražnjene, senzori na mostovima koji ustanovljuju opterećenje ili strukturalna oštećenja, senzori lokacije omogućuju praćenje staništa životinja... Kako jedan klasičan mehatronički uređaj postaje član interneta stvari te koje sve benificije takav sustav posjeduje, teme su ovog završnog rada.

2. KOMPONENTE SUSTAVA

Na slici 1 prikazana je pneumatska preša zajedno sa svim korištenim komponentama. [1]



Slika 1. Pneumatska preša

1-Pneumatski cilindar, 2-Proporcionalni tlačni regulator, 3-Pretvarač signala, 4-Opruga, 5-Senzor sile, 6-Napajanje, 7-Monostabilni razvodnik, 8-Senzor pozicije, 9-Arduino shield pločica, 10-Tiskana pločica, 11-Arduino MEGA mikrokontroler

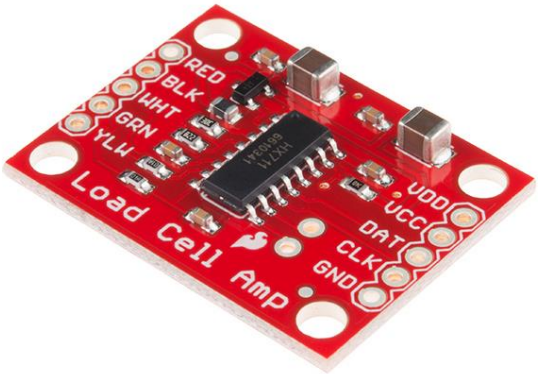
2.1. Senzor sile

Senzor sile vijcima je učvršćen za donju ploču pneumatske preše. Senzor ima sposobnost pretvaranja sile u iznosu do 2000 N u naponski signal, a sastoji se od 4 mjerača naprezanja međusobno spojenih u Wheatstonov most. Svaki mjerač ima sposobnost mjerenja električnog otpora koji je proporcionalan naprezanju na kaloti.

Slika 2. Senzor sile

2.2. Pojačalo

Kako bi se pojačao izlazni signal senzora u cilju povećanja točnosti mjerenja sile koristi se pojačalo. Pojačalo koristi 24 bitni A/D čip za precizno mjerenje težine s dvokanalnim analognim ulazom i 128x programabilnim pojačalom signala, a ulazni signal može davati mostni ('bridge') senzor.



Slika 3. Pojačalo HX711

2.3. Senzor pozicije

Raspon mjerenja ovog senzora je 4 do 30 cm što je za ovaj sustav pneumatske preše pogodno s obzirom na visinu same preše kao i hod opruge. Senzor radi na jednostavnom principu: infracrveni signal poslan sa senzora nailazi na prepreku, a analogni napon koji se vraća definira koliko je prepreka udaljena. U pravilu što je prepreka bliže, napon je veći.

Slika 4. Senzor pomaka Sharp GP2Y0A41SK0F

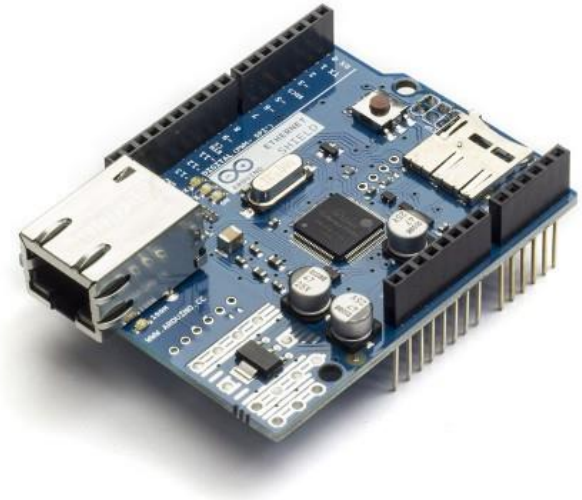
2.4. Mikrokontroler Arduino MEGA

Arduino Mega je mikrokontroler baziran na Atmega2560 čipu. Sadrži 54 digitalnih izlazno/ulaznih pinova, 16 analognih ulaza, 4 UART pinova, 16 MHz kristalni oscilator, USB konektor, napajanje, ICSP zaglavlje i dugme za reset. Može se napajati putem USB veze ili pomoću vanjskog napajanja. Sadrži 256 kB flash memorije za pohranu programskog koda, 8 kB statičkog RAM-a i 8 kB EEPROM-a (električno izbrisive programibilne ispisne memorije).

Slika 5. Mikrokontroler Arduino MEGA

2.5. Arduino ethernet shield

Arduino Ethernet Shield omogućuje Arduino mikrokontroleru spajanje na internet pomoću standardne RJ-45 konekcije. Baziran je na Wiznet W5100 Ethernet čipu. Wiznet W5100 podržava do 4 istovremenih socket konekcija (odnosi se i na TCP i UDP protokol). Također, na pločici se nalazi utor za mikro-SD karticu koja se može koristiti za pohranjivanje i slanje podataka preko mreže te reset dugme.



Slika 6. Arduino ethernet shield

2.6. Usmjerivač (engl. Router)

Usmjerivač je mrežni uređaj zadužen za usmjeravanje podatkovnih paketa na njihovom putu kroz računalnu mrežu pri čemu se taj proces odvija na trećem (mrežnom) sloju OSI modela.

Slika 7. Usmjerivač tp-link AC 750

2.7. USB kamera

Za potrebe ovoga rada USB kamera je potrebna kako bi se omogućio video prijenos u realnome vremenu.

Slika 8. USB kamera Logitech C170

2.8. Raspberry PI 3B+

Raspberry PI jedan je od najpoznatijih mikroračunala na svijetu. Model 3B+ sadrži: 64bitni 4-jezgreni procesor koji radi na frekvenciji od 1.4 GHz, 1 GB RAM memorije, WiFi, Bluetooth, HDMI izlaz, 4 USB priključka, slot za SD karticu na koju je spremljen operacijski sustav (u ovom slučaju Raspbian) kao i ostali podaci te LAN priključak.

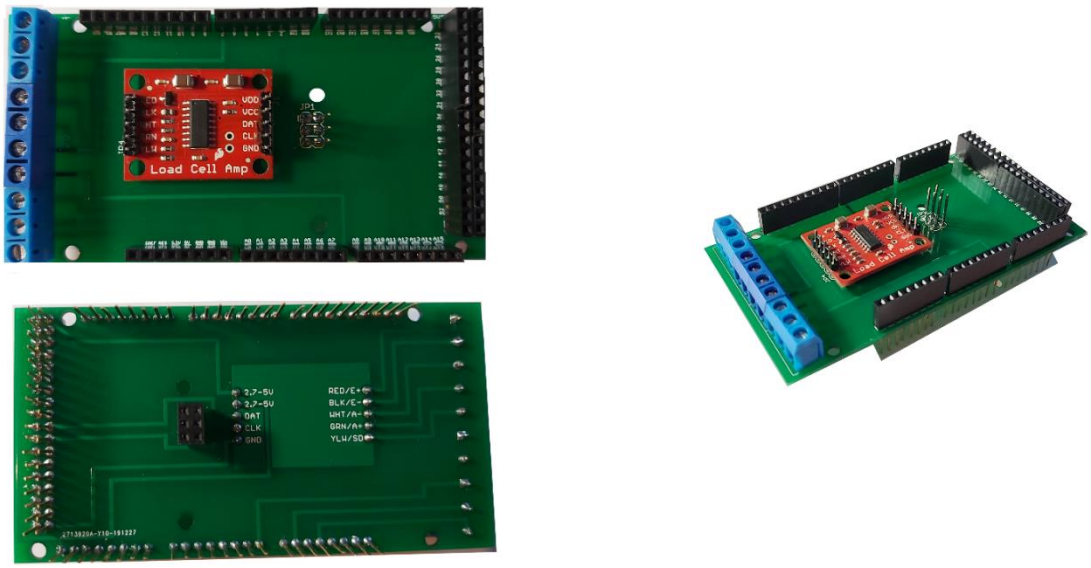
2.8.1. Webcam server

Funkcija Raspberry PI-a u ovom radu služi isključivo implementaciji video prijenosa uživo odnosno praćenje ponašanja samog uređaja (pneumatske preše). Za ostvarivanje video prijenosa potrebno je povezati Raspberry PI sa USB kamerom i usmjerivačem odnosno priključiti Raspberry PI lokalnoj mreži te mu pridružiti statičku lokalnu IP adresu. Također, potrebno je provesti instalaciju programskih paketa kao i konfiguraciju željenih postavki. Za omogućavanje pregleda video prijenosa preko globalne mreže (interneta) podignuti server preusmjerava se sa lokalnog porta na globalni.

Slika 9. Raspberry PI 3B+

2.9. Tiskana pločica

Za potrebe završnog rada bilo je potrebno izraditi tiskanu pločicu. Tiskana pločica smještena je između Arduino mikrokontrolera i Arduno Ethernet pločice koje međusobno komuniciraju preko tiskane pločice. Na samoj pločici smješteno je pojačalo, Arduino ICSP konektor, stezaljke za vodove te Arduino konektori.



Slika 10. Tiskana pločica

2.10. Pneumatski cilindar

Pneumatski cilindar izvršni je element u pneumatskom sustavu. Dvoradni cilindar ima jedan stupanj slobode gibanja (vertikalnu translaciju), a klip cilindra se izvlači i uvlači djelovanjem stlačenog zraka. Promjer cilindra iznosi D_c = 50mm, a hod klipa l=80mm.

Slika 11. Pneumatski cilindar

2.11. Proporcionalni tlačni regulator

Proporcionalni tlačni regulator je ventil sa implementiranim automatiziranim sustavom prekida dovoda zraka pri referentnoj vrijednosti tlaka pomoću strujnog signala na ulazu.

Slika 12. Proporcionalni tlačni regulator

2.12. Monostabilni razvodnik

Pneumatski razvodnik usmjerava tok stlačenog zraka propuštanjem, zatvaranjem i promjenom smjera toka. Tip razvodnika prikazan na slici 13 je 5/2 (pet kroz dva) odnosno riječ je o razvodniku s 5 priključaka i dva razvodna položaja. Karakteristika monostabilnog razvodnika jest da se poslije prestanka signala vraća u početni položaj.

Slika 13. Monostabilni razvodnik

2.13. Pretvarač signala

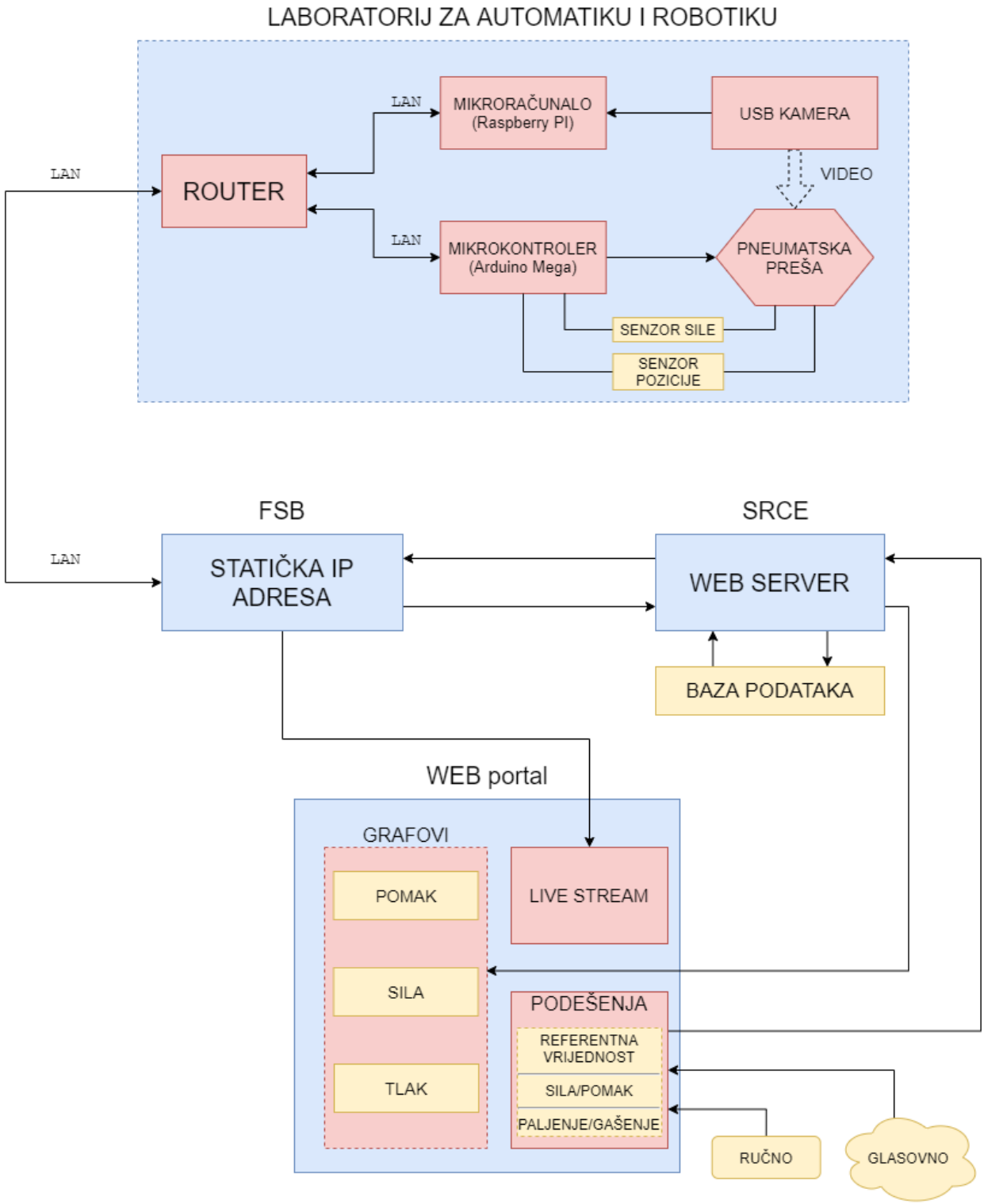
Funkcija pretvarača signala je pretvaranje izlaznog napona Arudino mikrokontrolera (PWM signala) od 0 do 5 V (0 do 255 PWM) u ulazni strujni signal proporcionalnog tlačnog regulatora u iznosu od 4 do 20 mA.

Slika 14. Pretvarač signala

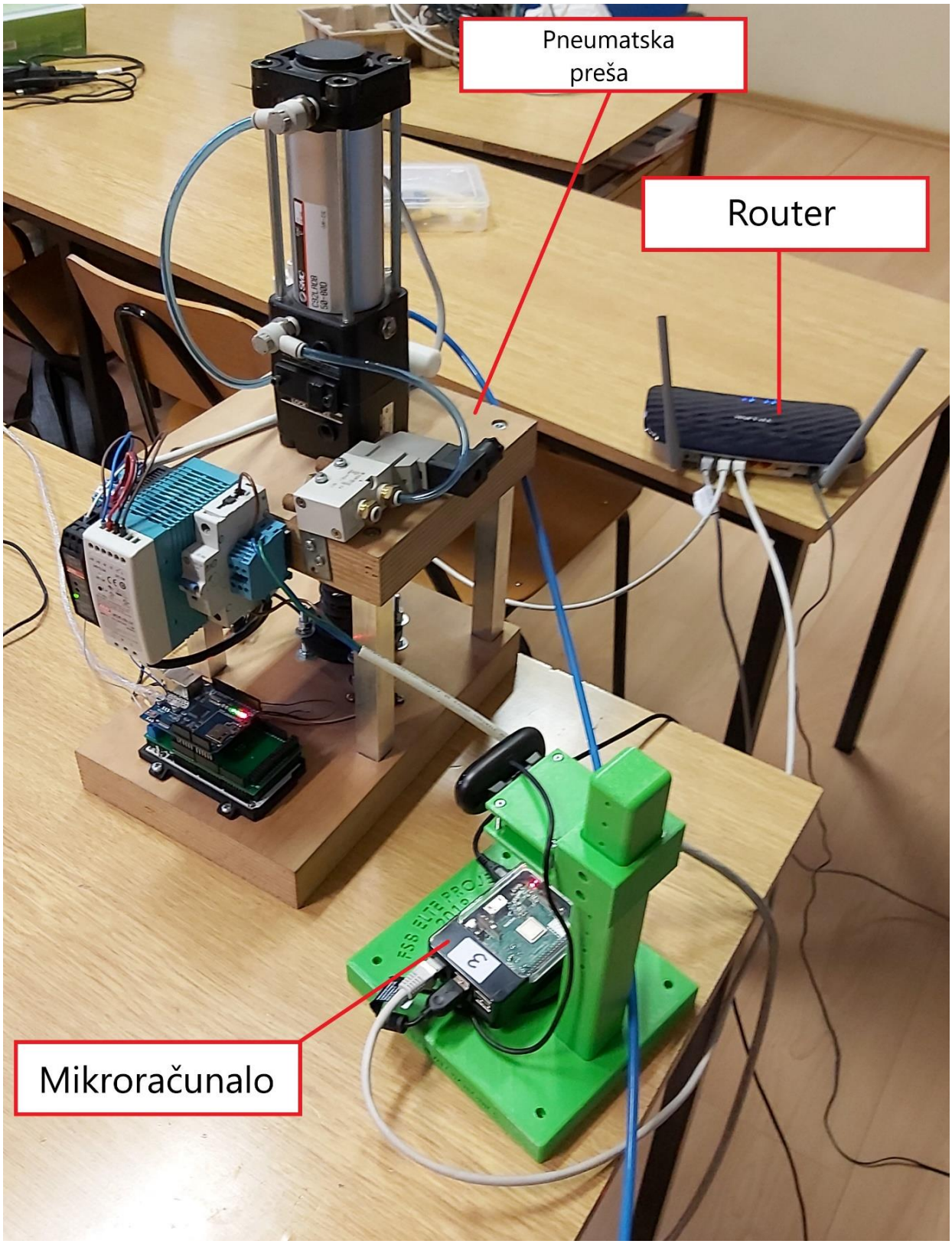
3. PRINCIP RADA SUSTAVA

Na WEB portalu implementirana su dva sučelja: administratorsko i korisničko. Administrator ima mogućnost definiranja termina pristupa upravljačkom sučelju WEB portala na način da isključivo jedan korisnik u jednome terminu može upravljati jednom pneumatskom prešom u vremenskom periodu određenom od strane administratora kako bi se izbjegla kolizija upravljanja. Mogućnosti koje se korisniku nude pri upravljanju su: uključivanje i isključivanje pneumatske preše, definiranje tipa regulacije (regulacija sile ili regulacija pomaka) te definiranje referentne vrijednosti (sile ili pomaka). Korisnik navedene opcije može definirati ručno (tipkovnica i miš) ili glasovno (implementirana neuronska mreža). Definirani podaci od strane korisnika šalju se na WEB server smješten u Sveučilišnom računskom centru (SRCE) koji preko socket komunikacije iste te podatke prosljeđuje vanjskoj statičkoj IP adresi servera lociranom na FSB-u (Fakultetu strojarstva i brodogradnje). Sa statičke IP adrese podaci se preko mrežnog kabela dostavljaju usmjerivaču (engl. Router). Usmjerivač dobivene podatke prenosi mikrokontroleru s kojim se nalazi u istoj lokalnoj mreži. Mikrokontroler podatke raspakirava i interpetira iščitavajući vrstu regulacije i referentnu vrijednost definiranu od strane korisnika. Na mikrokontroler spojeni su senzori sile i pozicije koji neprekidno mjere navedene fizikalne veličine pneumatske preše. S obzirom da mikrokontroler sada posjeduje informacije o referentnoj vrijednosti, vrsti regulacije i senzorskim vrijednostima, pomoću PID regulacije izračunava se naponski PWM signal. Izračunati PWM signal se pomoću pretvarača signala konvertira u ulazni strujni signal za elektro-pneumatski proporcionalni tlačni regulator kojim se regulira tlak u pneumatskom cilindru. Pneumatski cilindar sa jednim stupnjem slobode gibanja (vertikalnom translacijom) koristi se kao aktuator pretvaranjem komprimiranog zraka u mehaničku silu. Nakon dobivanja podataka, mikrokontroler počinje odgovarati na dobivenu poruku gdje prikupljene podatke sa senzora pozicije, sile i tlaka neprekidno šalje u suprotnome smjeru do WEB servera gdje se podaci spremaju u bazu podataka. Slanje podataka mikrokontroler prekida po nalogu korisnika odnosno odabirom opcije isključivanja preše na WEB portalu ili izlaskom iz WEB preglednika. Podaci iz baze podataka također se konstantno prikazuju korisniku na WEB portalu u obliku grafova čime korisnik dobiva povratnu informaciju vrijednosti sa senzora. Također, korisniku se nudi mogućnost praćenja ponašanja pneumatske preše tokom cijelog vremena odabranog termina. Na mikroračunalo spojena je USB kamera koja cijelo vrijeme snima pneumatsku prešu. Video signal mikroračunalo šalje

usmjerivaču koji dobiveni signal proslijeđuju FSB serveru te se u konačnici signal prikazuje u pregledniku korisnika.



Slika 15. Blokovska shema sustava



Slika 16. Uređaji spojeni u lokalnu mrežu

4. USPOSTAVLJANJE KOMUNIKACIJE

Za ostvarivanje komunikacije između programa, potrebno je definirati mrežne protokole. Koncept mrežnih protokola utvrđuje niz pravila da svaki sustav koristi jezik ostalih kako bi komunicirali. Protokoli opisuju format u kojem se poruka mora nalaziti, kao i način na koji se poruke razmjenjuju među programima.

4.1. OSI (Open System Interconnection) referentni model

OSI je model mrežne komunikacije koju je sastavio ISO (International Organization for Standardization) 1977. ISO organizacija stvorena je kao odgovor na potrebu standardizacije tehnologija i upravo joj je to posao.

OSI model dijeli mrežnu komunikaciju na 7 slojeva. Svaki od tih slojeva nosi određenu ulogu u prijenosu podataka mrežom. Podaci putuju slojevima OSI modela točno određenim redoslijedom. Tako aplikacijski sloj može komunicirati isključivo s aplikacijskim slojem na drugom računalu, dok je na istom računalu prezentacijski sloj jedini sloj kojemu može proslijediti podatke. Jednako tako, prezentacijski sloj može isključivo komunicirati s prezentacijskim slojem na drugom računalu, a na istom računalu može komunicirati s aplikacijskim i sesijskim slojevima. Pri slanju dokumenta preko mreže taj će dokument uvijek proći put od aplikacijskog sloja preko prezentacijskog sve do fizičkog, a na računalu koje prima taj dokument put će biti obrnut i ići će od fizičkog preko podatkovnog sloja do aplikacijskog [2].

Tablica 1. OSI model

7. APLIKACIJSKI SLOJ	To je sloj na kojem dolazi do spoja između aplikacije i mrežnog softvera. Na aplikacijskom sloju nalaze se programi koji omogućuju mrežnu komunikaciju.
6. PREZENTACIJSKI SLOJ	Sloj na kojem se nalazi softver koji uspostavlja konvenciju koja će se koristiti pri komunikaciji, poput jezika i sl., a glavna mu je namjena biti prevodilac tj. prevesti tekst koji želimo poslati u jezik razumljiv mrežnom dijelu softvera koji će taj tekst prenijeti na drugo računalo.
5. SLOJ SESIJE	Sloj sesije zadužen je za uspostavu i održavanje sesije između procesa koji komuniciraju.
4. TRANSPORTNI SLOJ	Na transportnom se sloju odlučuje hoće li se koristiti pouzdani (TCP) ili nepouzdani (UDP) protokol te se na temelju toga od podataka kreiraju segmenti na koje se dodaju određene informacije poput odredišnog i ishodišnog porta. Također transportni sloj se brine da poslani podaci budu isporučeni.
3. MREŽNI SLOJ	Mrežni sloj brine o tome da podaci budu usmjereni pravim putem do odredišta, odnosno da ne zalutaju. Paket na mrežnom sloju dobija ishodišnu i odredišnu IP adresu. Ovaj sloj nudi mogućnost usmjeravanja podataka preko mreže.
2. SLOJ VEZA	Sloj veza je sloj koji pakete stavlja u okvire i šalje ih na medij za prijenos podataka. U okvir se stavljaju ishodišna i odredišna fizička (MAC) adresa uređaja.
1. FIZIČKI SLOJ	Na ovom su sloju definirane fizičke osobine računala.

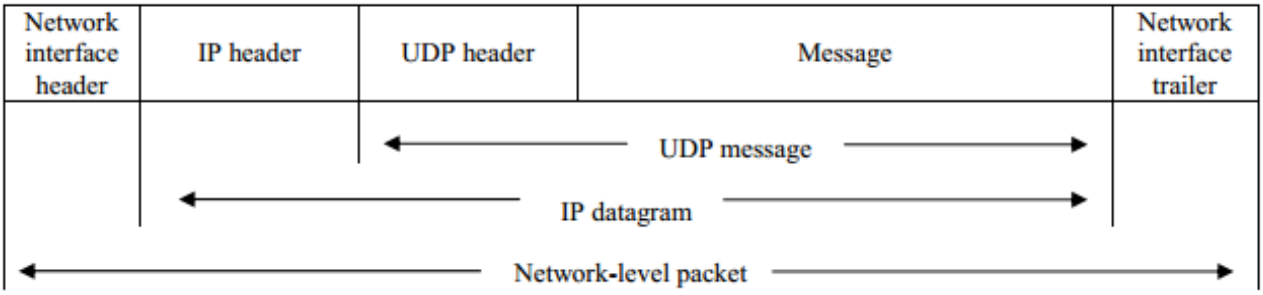
4.2. Protokoli transportnog sloja

4.2.1. User Datagram Protocol (UDP)

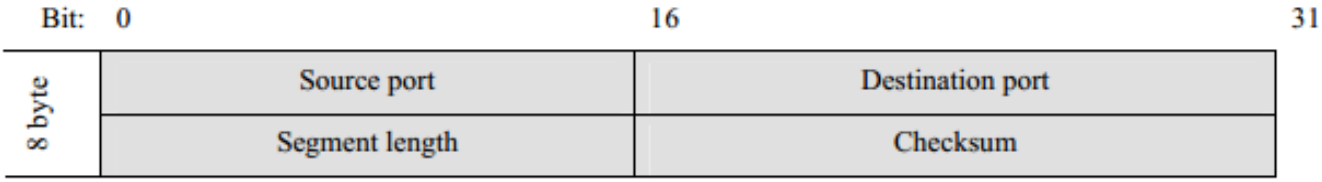
UDP je jednostavni protkol koji se nalazi na transportnoj razini OSI modela, te je zajedno sa TCP protokolom jedan od temeljnih internet protokola. UDP omogućuje slanje kratkih poruka (engl. datagram) između aplikacija na mrežnim računalima. Aplikacija šalje poruku UDP socketu koja se prvo enkapsulira u UDP poruku, a zatim i u IP poruku te u konačnici šalje na odredišnu adresu. Jedna od osnovnih karakteristika UDP protokola je njegova nepouzdanost odnosno ne postoji garancija da će se UDP poruka isporučiti traženoj destinaciji. Također, nije sigurno da će redoslijed same poruke biti sačuvan kao što ne postoji garancija da će poruka stići samo jednom.

Ukoliko se poruka isporuči traženoj destinaciji, a sustav za kontrolu detektira grešku pri isporuci ili ukoliko je poruka ispuštena negdje u mreži, poruka neće biti dostavljena UDP socketu. Ukoliko poruka nije dostavljena UDP socketu, ne dolazi do automatskog ponovnog slanja poruke. Stoga, želimo li osigurati ispostavu poruke odredištu potrebno je implementirati opcije unutar programa poput potvrde prijama sa drugog kraja komunikacijske veze, definiranje programskih pauza (engl. timeout), sustav za prisilno ponovno slanje...

Svaka UDP poruka ima svoju duljinu koja je zajedno sa podacima proslijeđena odredišnoj aplikaciji.



Slika 17. UDP poruka (datagram)



Slika 18. UDP zaglavlje (header)

Tablica 2. UDP zaglavlje

Source port	2 byte	Port sa kojeg se šalje UDP poruka, tj. definira koja aplikacija na aplikacijskom sloju šalje poruku. Polje je opcionalno te ako se ne koristi može biti 0.
Destination port	2 byte	Port na koji stiže UDP poruka, tj. definira se koja aplikacija aplikacijskog sloja prima poruku. Zajedno s IP adresom iz IP zaglavljem (engl. header) jedinstveno označava kojem procesu tj. aplikaciji je poruka poslana.
Length	2 byte	Minimalna duljina je 8 byte-ova a teoretska maksimalna 65,515 (duljina IP paketa minus zaglavlja). Prava maksimalna duljina je ograničena MTU-om linka preko kojeg se prenose paketi. Ovo polje je redudantno zato što se duljina paketa uvijek može izračunati iz duljine IP paketa.
Checksum	2 byte	Polje koje osigurava integritet podataka.

Još jedno važno svojstvo UDP protokola je pružanje bespojne usluge. Drugim riječima između klijenta i servera ne treba postojati nikakva dugoročna veza. Tako, primjerice, UDP klijent može kreirati socket i slati poruke jednom serveru, a zatim odmah poslati drugu poruku preko istoga socketa drugom serveru. Isto tako, UDP server može prihvatiti nekolicinu poruka na pojedinačnom UDP socketu od kojih svaka može biti od drugog klijenta [2].

4.2.2. Transimission Control Protocol (TCP)

Usluga pružena putem TCP protokola razlikuje se od one pružene UDP protokolom. TCP omogućuje uspostavu klijent-server komunikacije. TCP klijent prvo uspostavlja komunikaciju sa serverom, zatim se preko konekcije razmjenjuju podaci, a na kraju razmjene podataka konekcija se zatvara.

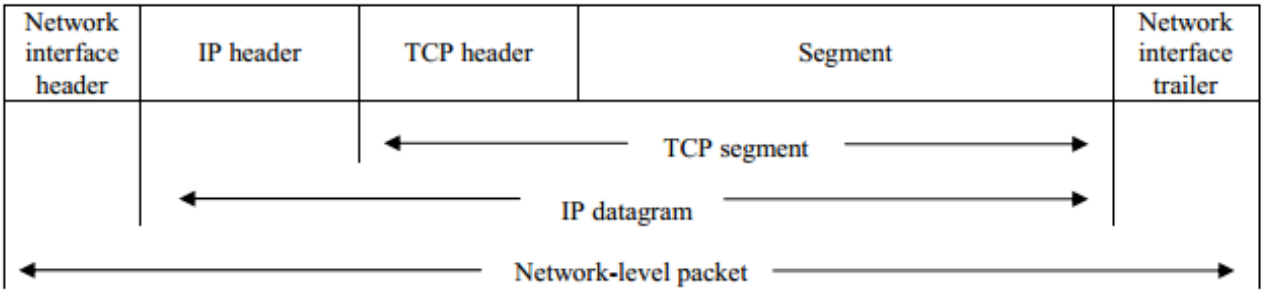
Za razliku od nepouzdanog UDP protokola, TCP protokol je pouzdan. Slanjem podataka putem TCP-a na drugi kraj komunikacijske veze zatražuje se potvrda prijama. Ukoliko potvrda nije primljena, TCP automatski ponovno šalje podatke te ovog puta čeka dulje vrijeme na potvrdu. Ovaj proces ponavlja se određeni broj puta nakon čega TCP odustaje od slanja podataka (obično u periodu između 4-10 minuta, ovisno o implementaciji).

Važno je napomenuti da TCP ne garantira uspješno dospijeće podataka stoga se pridjev pouzdani treba uzeti sa rezervom pošto se ne radi o 100% pouzdanom protokolu. Zadaća TCP-a je dostavljanje podataka odredištu ukoliko je to moguće, a ukoliko nije moguće korisnik se obvezno obavještava. Unutar TCP-a implementiran je algoritam dinamičke procjene vremena dostave podataka odnosno RTT algoritam (engl. round-trip time) između klijenta i servera. Na temelju RTT algoritma određuje se vrijeme čekanja potvrde o dospijeću podataka, i to vrijeme se konstantno ponovno procjenjuje. Tako primjerice u lokalnoj (LAN) mreži RTT može iznositi nekoliko milisekundi dok u globalnoj (WAN) mreži može doseći i vrijeme od par sekundi.

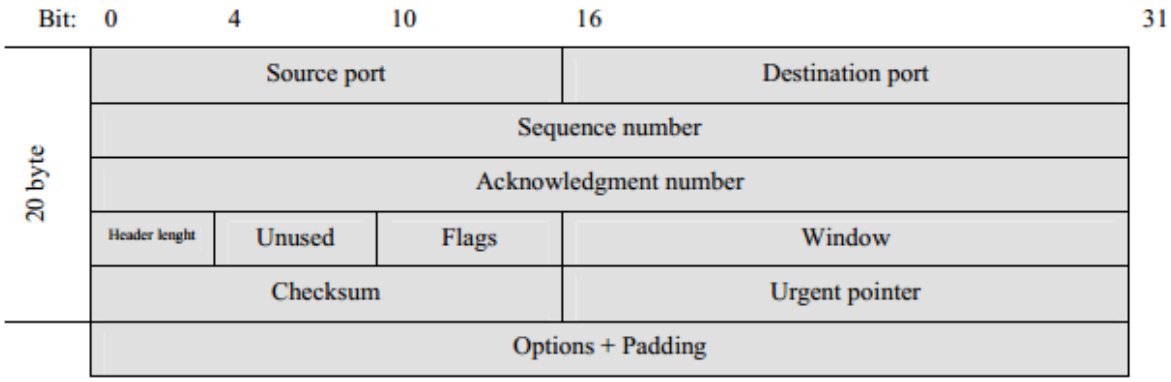
Podaci poslani putem TCP-a posloženi su na način da je svaki poslani byte povezan sa rednim brojem. Navedena tvrdnja lakše je razumljiva putem primjera: pretpostavimo li da aplikacija ispunjava TCP socket sa 2048 byte, što znači da TCP šalje 2 segmenta (segment je mjera podatka koje TCP proslijeđuje IP-u): prvi kojem su pridruženi redni brojevi od 1 do 1024 i drugi koji se sastoji od podataka sa rednim brojevima od 1025 do 2048. Ukoliko segmenti ne dođu na odredište točnim redoslijedom, odredišni TCP socket presložiti će dva segmenta s obzirom na redne brojeve prije nego što proslijedi podatke odredišnoj aplikaciji. Ukoliko TCP primi duplicirane podatke (npr. klijentska strana je dobila netočnu obavijest da je segment odbačen jer je zbog prebukiranosti mreže vrijeme čekanja premašeno), on pomoću rednih brojeva to može detektirati te odbaciti duplicirane podatke.

Također, TCP posjeduje ugrađenu kontrolu protoka. TCP uvijek šalje obavijest o broju byte-ova podataka koje je slobodan primiti u određenom vremenskom intervalu kako bi se spriječilo preopterećenje buffera.

Konačno, TCP konekcija je full-duplex, odnosno aplikacija može putem jedne konekcije istovremeno slati i primati podatke u oba smjera. U tom slučaju TCP mora cijelo vrijeme pratiti stanje određenih informacija (redni brojevi, popunjenost buffera) u oba smjera komunikacijske veze. Nakon što je uspostavljena full-duplex konekcija, po potrebi moguće je prijeći i na simplex konekciju [2].



Slika 19. TCP segment



Slika 20. TCP zaglavlje

Tablica 3. TCP zaglavlje

Source port	2 byte	Socket s kojeg se šalje TCP segment, tj. definira koja aplikacija na aplikacijskom sloju šalje segment. Source socket zajedno s IP adresom jedinstveno definira mjesto sa kojeg je segment poslan.
Destination port	2 byte	Port na koji stiže TCP poruka, tj. definira se koja aplikacija aplikacijskog sloja prima poruku. Zajedno s IP adresom i IP zaglavljem (engl. header) jedinstveno označava kojem procesu tj. aplikaciji je poruka poslana.
Sequence number	4 byte	Redni broj početnog okteta segmenta.
Acknowledgment number	4 byte	Broj sljedećeg okteta korisnikove poruke, ujedno i kumulativna potvrda.
Data offset	4 byte	Početak podataka u paketu, ujedno označava veličinu TCP zaglavlja.
Reserved	6 bit	Rezervirano za buduću uporabu (postavljeno na 0).
Flags	6 bit	URG, ACK, PSH, RST, SYN, FIN flagovi.
Window	2 byte	Veličina buffera pošiljatelja poruke koji služi za primanje poruka. TCP/IP stack drugog sudionika treba slati pakete veličine maksimalno do veličine prozora. Ako je poslana 0, poruke se dalje ne šalju sve dok se ne pošalje nova poruka sa vrijednošću većom od 0.
Urgent pointer	2 byte	Lokacija hitnih podataka u segmentu.
Options		Proizvoljne opcije koje se dodaju TCP headeru u paketima od 4 bytea.

4.2.3. Usporedba UDP I TCP protokola

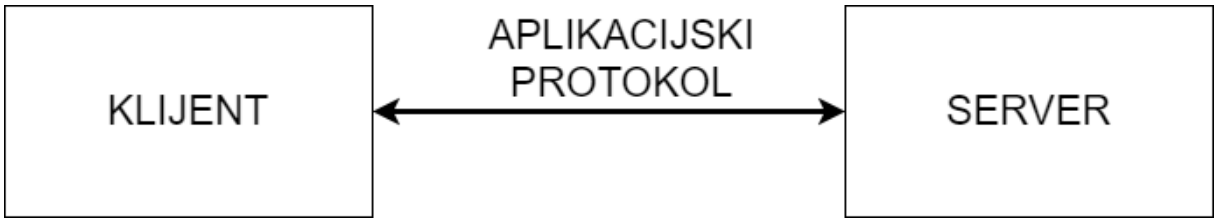
Tablica 4. Usporedba UDP i TCP protokola

	TCP	UDP
Konekcija	Aplikacijski procesi uspostavljaju konekciju prije same razmjene poruka.	Aplikacijski procesi izmjenuju poruke bez stvaranja konekcije.
Primjena	Pogodan za aplikacije koje iziskuju veliku pouzdanost, a vrijeme slanja je manje značajno.	Pogodan za aplikacije koje zahtjevaju brzi i efikasni prijenos a pouzdanost je manje značajna.
Pouzdanost	Garantira ili sigurno dospijeće poruke ili povratnu informaciju o njenom nedospijeću.	Ne postoji garancija dospijeća poruke niti povratna informacija ukoliko poruka nije dostavljena.
Raspoređivanje podataka unutar segmenta	Točno određeni redoslijed rasporeda podataka.	Segmenti su međusobno neovisni.
Potvrda prijama	Potvrda prijama segmenata.	Ne postoji potvrda prijama segmenata.
Kontrola protoka	Slanje obavijesti o broju byte-ova podataka koje je slobodan primiti u određenom vremenskom intervalu.	Ne postoji opcija kontrole protoka.
Provjera grešaka	Ponovno slanje segmenata s greškom.	Segmeti sa greškom se odbacuju. Ponovno slanje segmenata se ne izvršava.

Nakon kratke analize glavnih protokola transportnog sloja, u ovom završnom radu odabrana je komunikacija UDP protokolom sa implementiranom programskom pauzom (engl. timeout) na dolaznoj strani komunikacijske veze. Razlog tome je važnost brzine prijenosa podataka kod ovakvih sustava, a količina poslanih podataka je velika.

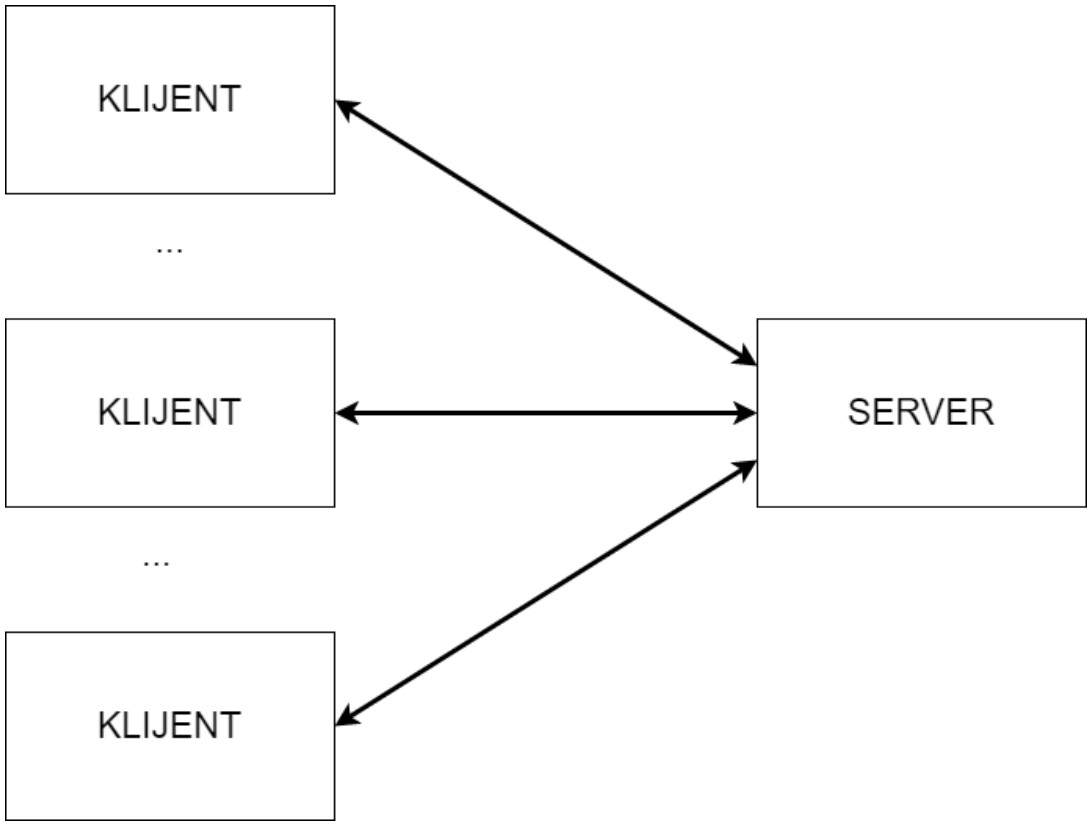
4.3. Mrežno programiranje

Pod pojmom mrežno programiranje podrazumijevamo komunikaciju između programa preko računalne mreže putem određenog protokola. Jedan program se uobičajeno naziva klijent, a drugi server. Komunikacija između klijenta i servera je na principu zahtjeva i odgovora gdje zahtjev (engl. request) za traženim sadržajem upućuje klijent dok odgovor na zahtjev (engl. response) šalje server. Ovakva klijent-server komunikacija koristi se kod većine mrežnih aplikacija, a odluka da klijent uvijek inicijalizira zahtjev znatno pojednostavljuje ne samo protokol već i sami program [3].



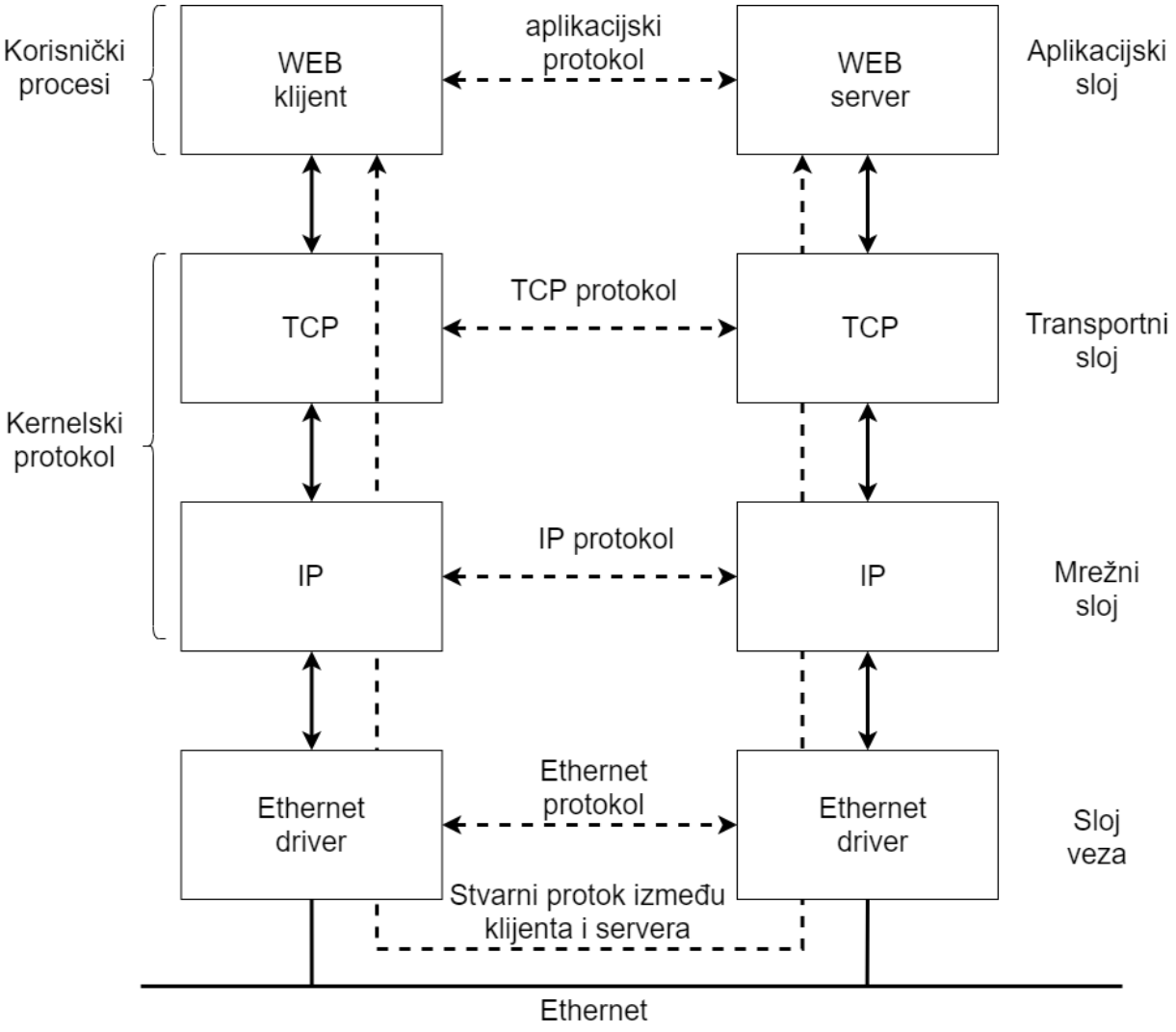
Slika 21. Mrežna aplikacija klijent-server

I dok je za klijente uobičajena komunikacija sa isključivo jednim serverom pojedinačno (iako koristeći WEB preglednik kao primjer, često uspostavljamo komunikaciju sa više različitih WEB servera u kratkom vremenskom periodu), serveri ostvaruju komunikaciju sa više klijenata istovremeno.



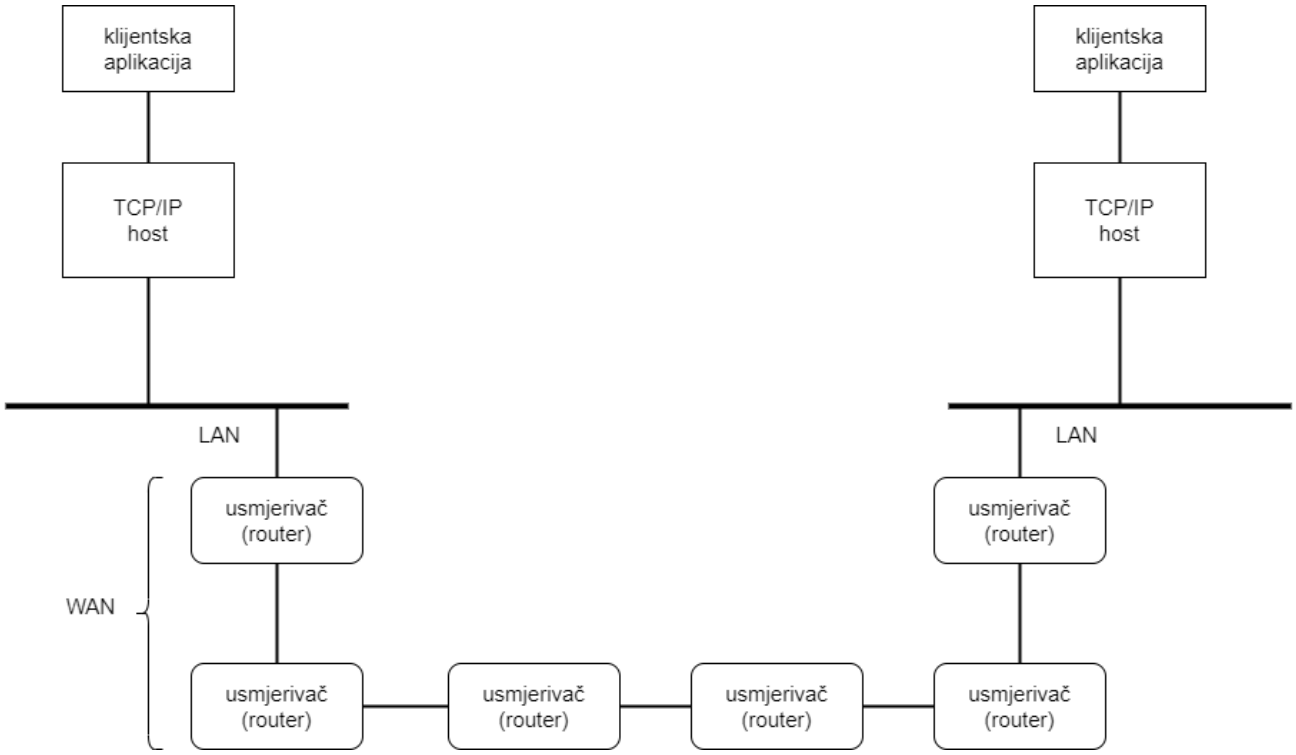
Slika 22. Upravljanje servera sa više klijenata istovremeno

Tijekom komunikacije između klijenta i servera uključeno je više protokola i to na različitim slojevima OSI modela. Tako na primjer WEB klijent i server komuniciraju koristeći TCP protokol. TCP, naizmjence, koristi Internet protokol (IP), a IP komunicira sa slojem veza. Blokovski prikaz ovakve komunikacije za slučaj da se server i klijent nalazi na istom Ethernetu prikazan je na slici 23.



Slika 23. Klijent i server u istom Ethernetu

S obzirom da je za potrebu ovoga rada potrebna uspostava komunikacije preko globalne (WAN) mreže bitno je naglasiti da klijent i server ne moraju biti nužno povezani u lokalnu strukturu. Primjer takve strukture gdje se klijent i server nalaze na različitim lokalnim (LAN) mrežama koje su povezane preko globalne (WAN) mreže putem usmjerivača prikazana je na slici 24.



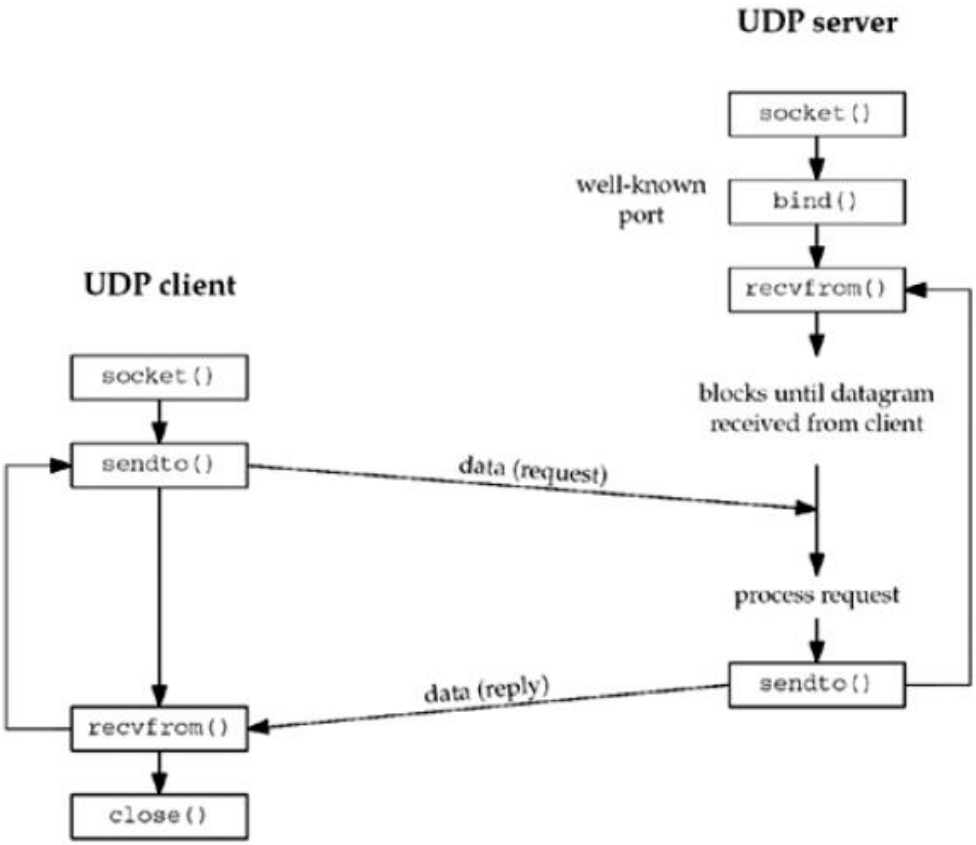
Slika 24. Klijent i server u različitim LAN mrežama povezani preko WAN-a

Usmjerivači su osnovni gradivni blokovi WAN mreže. Najveća WAN mreža današnjice svakako je internet iako mnoge tvrtke uspostavljaju osobne WAN mreže koje mogu, ali ne moraju biti povezane na internet.

4.4. UDP socket

Socket je jedan kraj dvosmjerne komunikacijske veze između programa koji se izvršavaju na istoj mreži. Socket je određen IP adresom i portom. Najčešća primjena socketa je u klijent-server arhitekturi gdje server na određenom portu sluša zahtjeve klijenta. Kao što je opisano u prethodnom potpoglavlju, UDP protokol karakteriziran je kao bespojni, nepouzdani protokol. Kod UDP protokola klijent ne uspostavlja konekciju sa serverom. Umjesto toga, klijent šalje poruke serveru koristeći *sendto* funkciju, čiji prametar je odredišna IP adresu (IP adresa servera). Slično tome, server ne prihvaća konekciju od strane klijenta. Server jednostavno poziva *recvfrom* funkciju koja iščekuje prijem podataka. Izlaz *recvfrom* funkcije jest klijentska adresa zajedno sa porukom kako bi server mogao poslati zahtjev ispravnome klijentu.

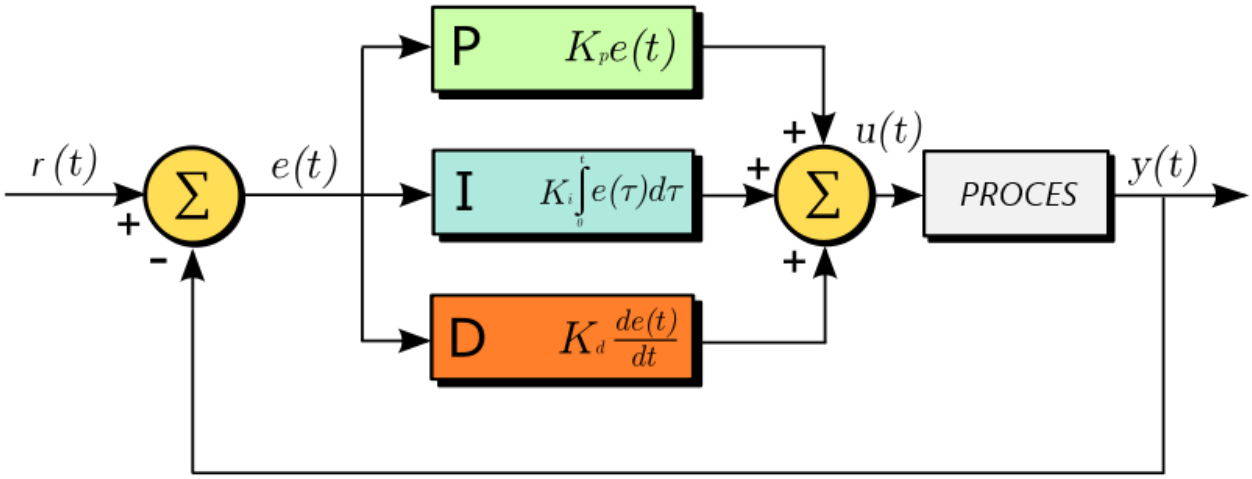
Na slici 25 prikazane su funkcije tipične UDP klijent-server komunikacije [3].



Slika 25. Osnovne UDP socket funkcije

5. PID REGULACIJA

Nakon uspostave komunikacije između korisnika i mikrokontrolera, tražene zahtjeve korisnika potrebno je realizirati. Drugim riječima, potrebno je postići referentnu vrijednost diktiranu od strane korisnika koja se komunikacijskom vezom šalje do mikrokontrolera. Upravo to je zadaća PID regulacije. Proporcionalno-integracijsko-derivacijska regulacija (PID-regulacija) temelji se na trikomponentnim PID regulatorima. PID regulatori dobijaju se paralelnim spojem proporcionalnog, integracijskog i derivacijskog regulatora. PID regulacija zbog svojeg robusnog djelovanja te jednostavnosti implementacije najpopularniji je regulacijski algoritam u industriji. Elementi PID regulatora prikazani su na slici 26 [4].



Slika 26. PID reguator

PID regulator ima sva potrebna dinamička ponašanja za potrebu kvalitetne regulacije:

- P član posjeduje odgovarajući energetski sadržaj unutar određenog područja regulacijskog odstupanja kako bi se eliminirale vlastite oscilacije,
- D član je odgovoran za brzu reakciju na naglu promjenu pogreške,
- I član je zadužen za povećanje upravljačkog signala kako bi se minimizirala pogreška.

Izraz za algoritam regulacije PID regulatora:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt} \right] \qquad [5.1]$$

Gdje su:

K_p – pojačanje proporcionalnog djelovanja

$e(t)$ – signal greške

T_i – integracijsko vrijeme

T_d – derivacijsko vrijeme

$u(t)$ – upravljački signal

6. PREPROCESIRANJE PODATAKA

Pod pojmom korisničko iskustvo (engl. user experience (UX)) podrazumijeva se sveukupni korisnički doživljaj prilikom interakcije sa proizvodom (poput WEB aplikacije). Cilj korisničkog iskustva je pružanje pozitivnog doživljaja koje će uzrokovati lojalnost samoga korisnika proizvodu. Vodeći se tom činjenicom, u sklopu završnog rada integriran je dodatni sustav glasovne interakcije korisnika sa WEB aplikacijom.

Klasifikacija zvukova trenutačno je veoma popularno područje istraživanja sa brojnim primjenama u stvarnome svijetu. Uzimajući u obzir nedavna ostvarenja u klasifikaciji slika pomoću konvolucijskih neuronskih mreža, isti pristup primjeniti će se na klasifikaciju zvučnih datoteka.

6.1. Prikupljanje podataka

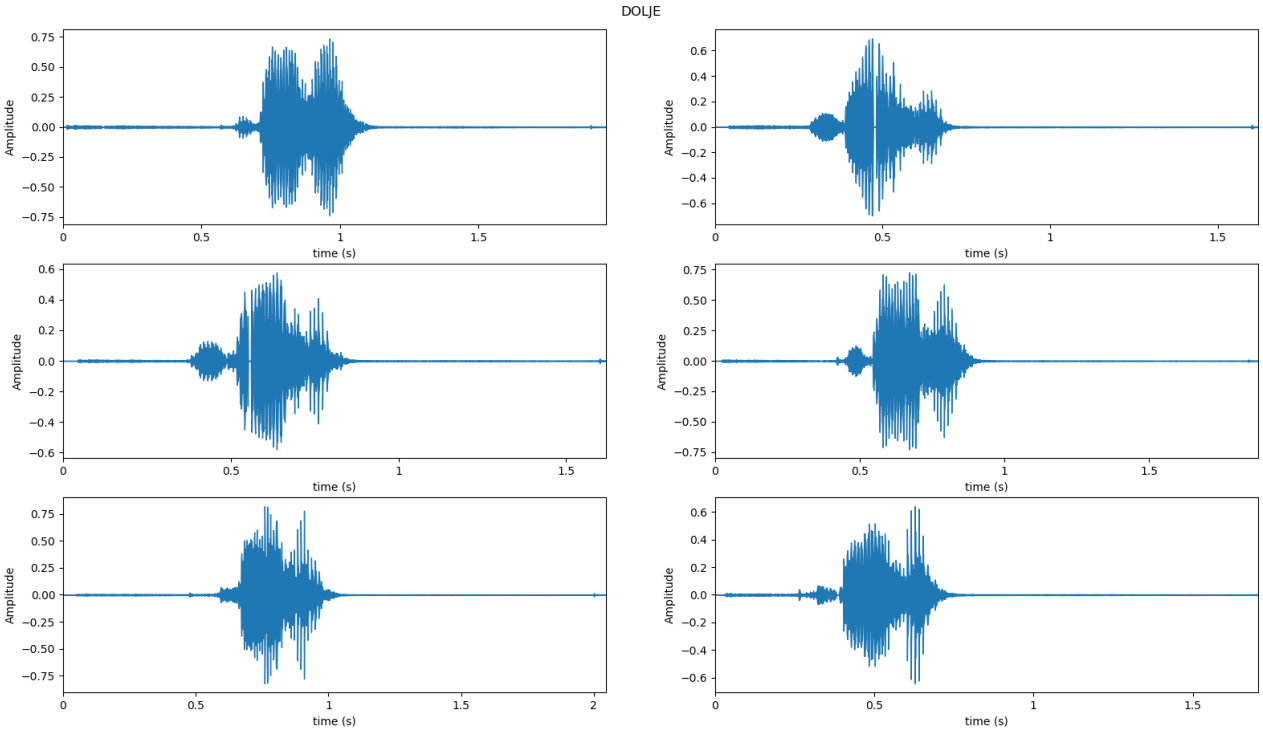
Cilj neuronske mreže je uspješno prepoznavanje određenih riječi. Prvi korak u treniranju neuronske mreže prikupljanje je uzoraka. U ovome radu definirano je 7 riječi koje će se klasificirati: „dolje“, „gore“, „pomak“, „sila“, „stani“, „ugasi“ i „upali. S obzirom da ne postoji dostupan set podataka zvučnih datoteke navedenih riječi, podaci su osobno prikupljeni. Za svaku riječ prikupljeno je 2000 zvučnih uzoraka, koje će činiti ulaz neuronske mreže.

6.2. Vizualizacija podataka

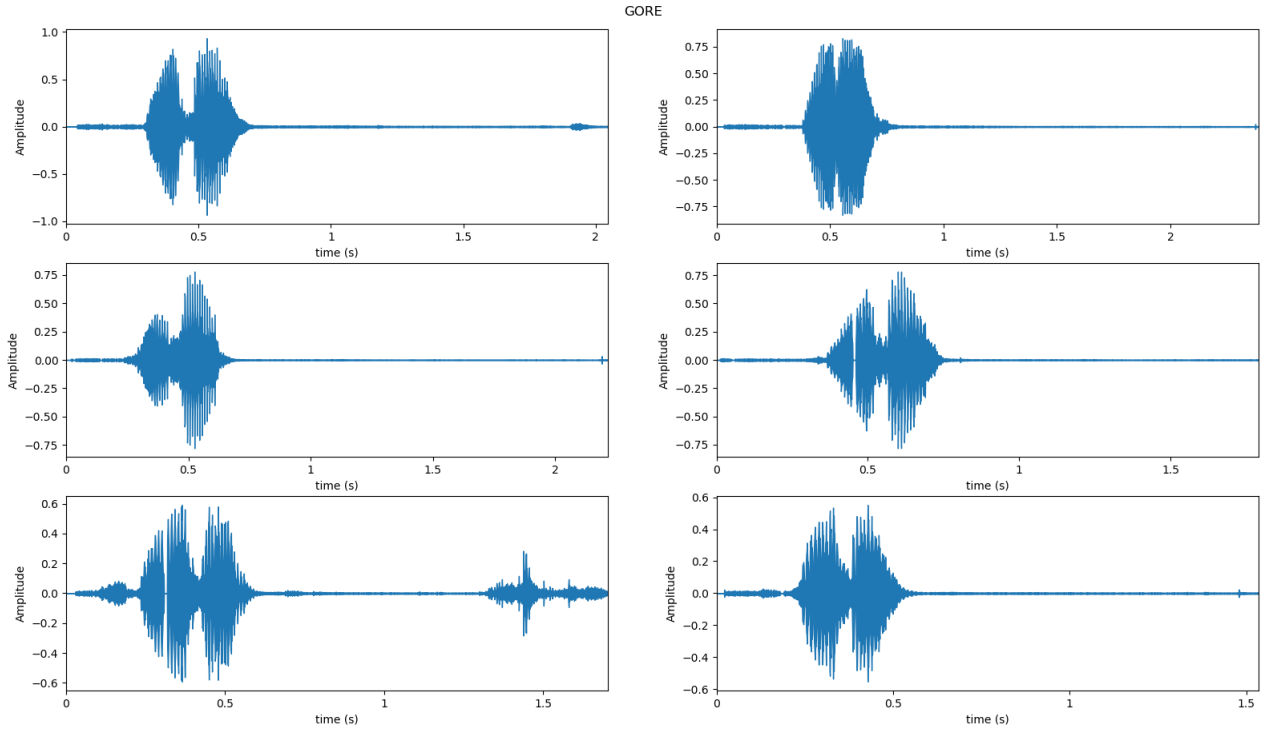
Nakon prikupljanja podataka, podatke je potrebno što bolje reprezentirati na način da uzorci istih riječi imaju s jedne strane što veću međusobnu sličnost, a s druge strane da budu što različitiji ostalim riječima. U potrazi za što boljim rješenjem zvučne datoteke će se vizualizirati.

6.2.1. Vremenska domena

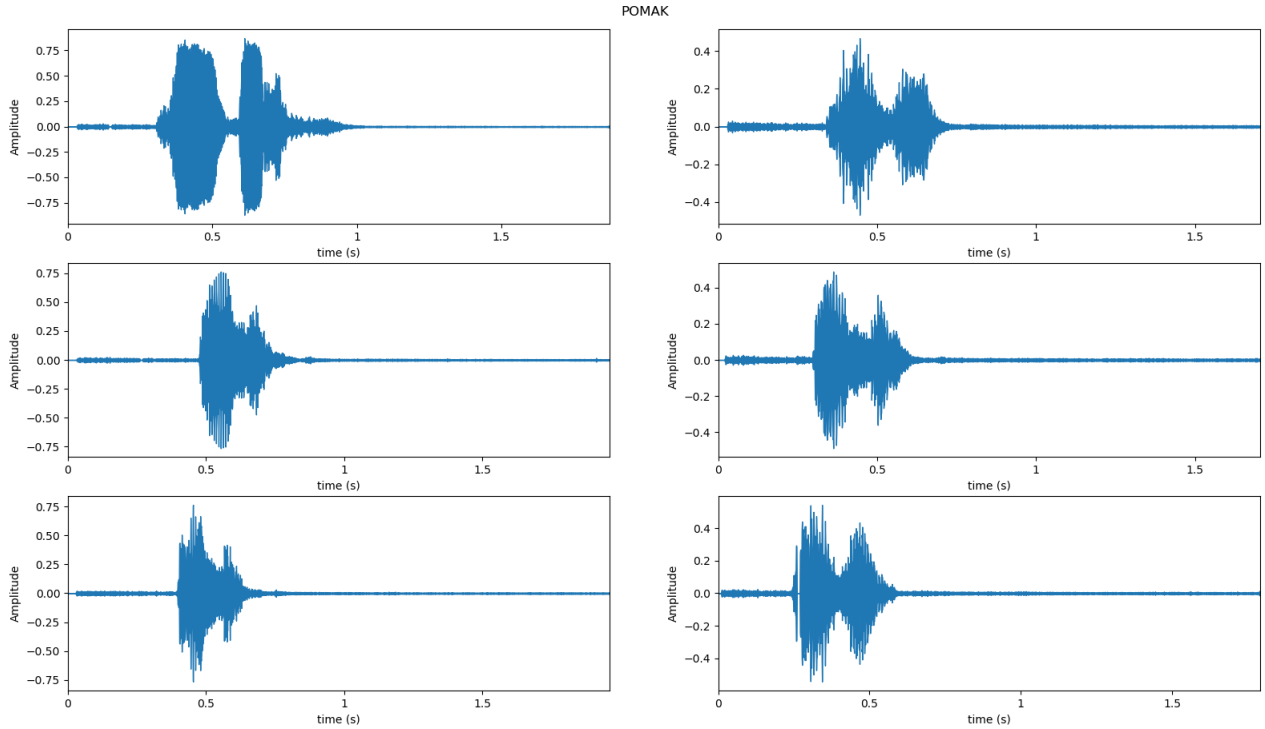
Prvi korak biti će prikaz podataka u vremenskoj domeni. Po šest primjeraka svake riječi prikazano je u vremenskoj domeni.



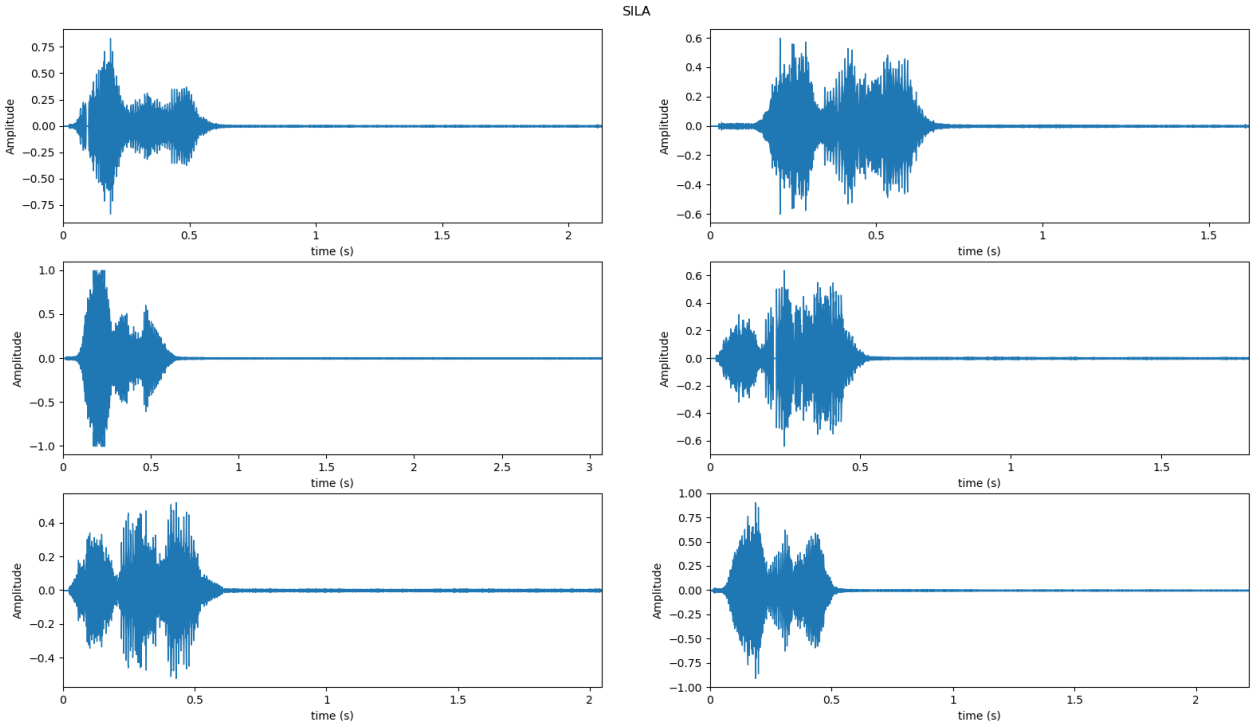
Slika 27. Uzorci riječi „dolje“ u vremenskoj domeni



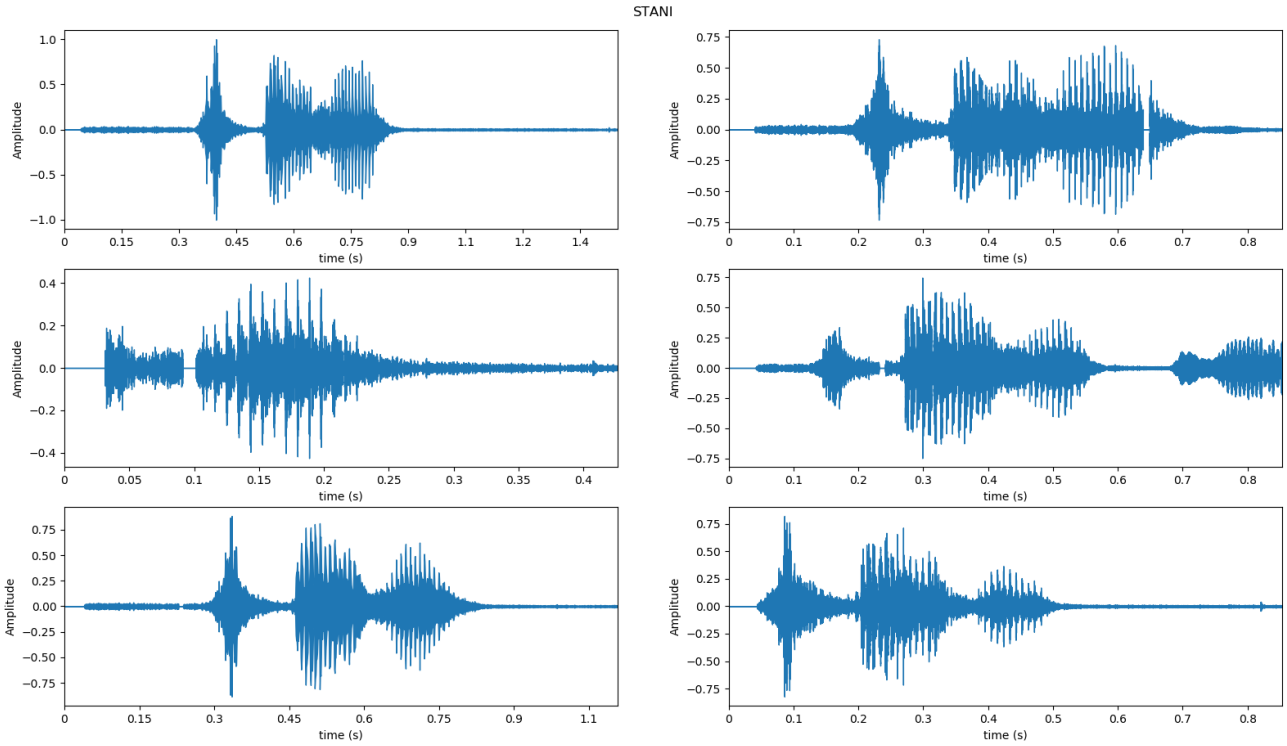
Slika 28. Uzorci riječi „gore“ u vremenskoj domeni



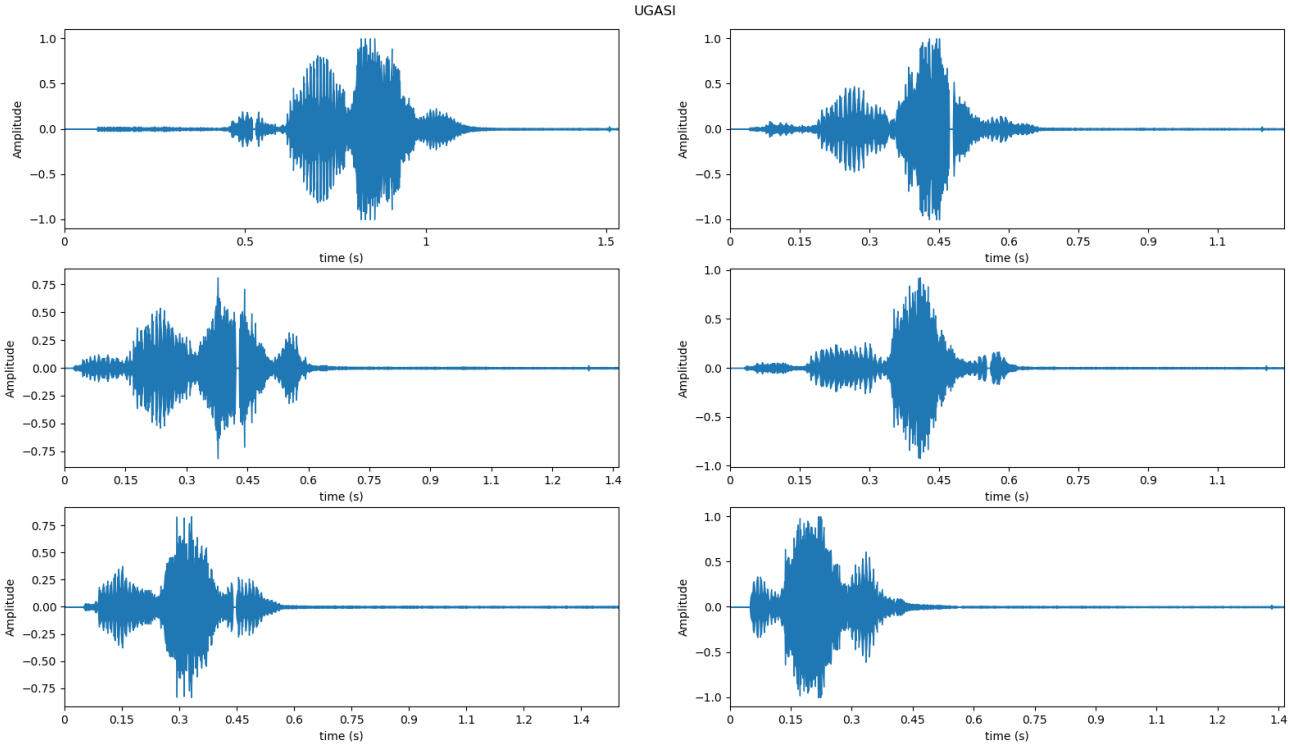
Slika 29. Uzorci riječi „pomak“ u vremenskoj domeni



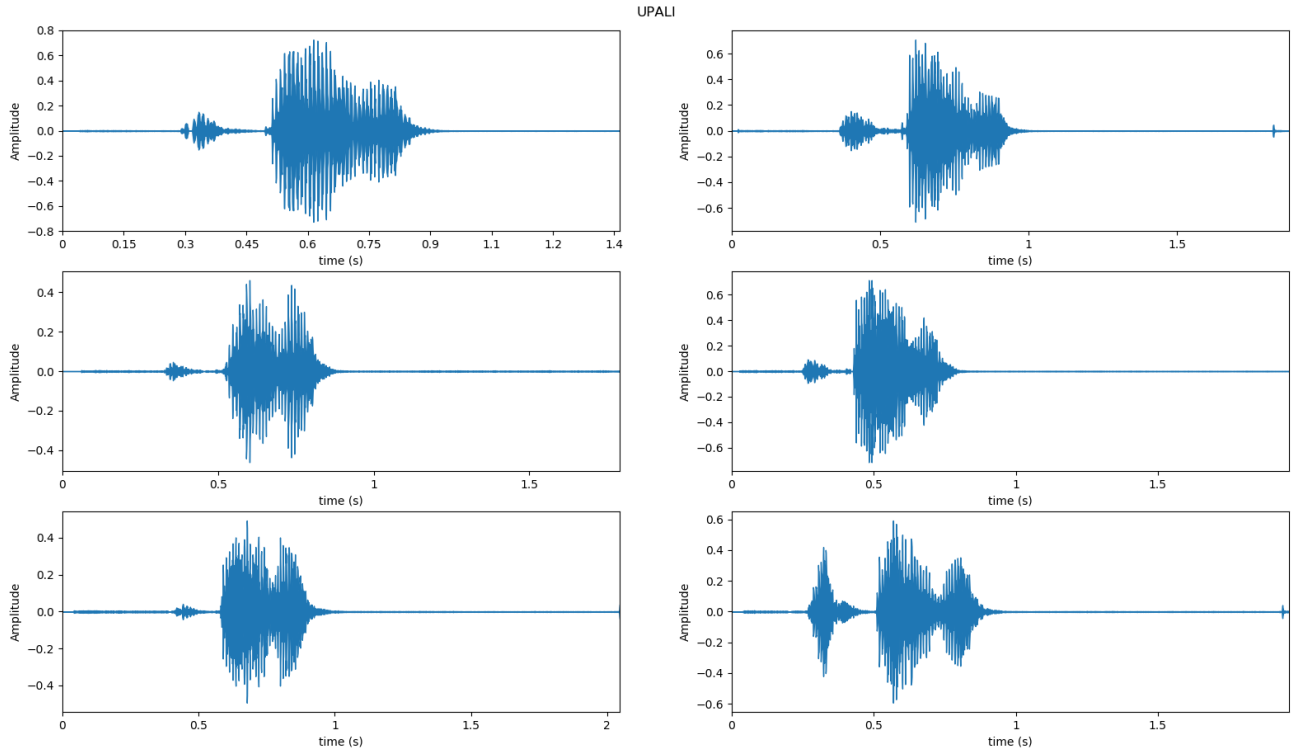
Slika 30. Uzorci riječi „sila“ u vremenskoj domeni



Slika 31. Uzorci riječi „stani“ u vremenskoj domeni



Slika 32. Uzorci riječi „ugasi“ u vremenskoj domeni



Slika 33. Uzorci riječi „upali“ u vremenskoj domeni

Reprezentacija signala u vremenskoj domeni (engl. time-domain) prikazuje odnos amplitude zvučnog vala s promjenom vremena pri čemu vrijednost amplitude jednake nuli označava tišinu.

Korak vizualizacije signala u vremenskoj domeni dobar je početni korak za daljnju analizu i poboljšanu reprezentaciju signala, međutim jasno je da odnos amplitude i vremena nije previše informativan s obzirom da iz njega jedino iščitavamo glasnoću zvučnog signala. Shodno tome, signale prikazujemo u frekvencijskoj domeni u kojoj tumačimo koje su sve frekvencije u signalu prisutne. Za prikaz signala u frekvencijskoj domeni nužna je određena matematička operacija koja će signal iz vremenske domene transformirati u frekvencijsku. Takva operacija naziva se Fourierova transformacija.

6.2.2. Fourierova transformacija

Zvučni (audio) signal kompleksan je signal koji se sastoji od više „jedno-frekvencijskih audio valova“ koji zajedno putuju kroz medij. Prilikom snimanja zvukova snimaju se jedino

rezultantne amplitude valova. Fourierova transformacija je reverzibilna, linearna matematička tehnika koja vrši transformaciju funkcije iz vremenske domene $s(t)$ u frekvencijsku $S(\omega)$ pri čemu razlaže signal na njegove sastavne frekvencije [4].

Definicija direktne Fourierove transformacije:

$$S(\omega) = \int_{-\infty}^{+\infty} s(t)e^{-j\omega t}dt \qquad [6.1]$$

Tranformacija koja ima učinak suprotan Fourierovoj naziva se inverzna Fourierova transformacija. Inverzna Fourierova transformacija funkciju iz frekvencijske domene S(ω) transformira u vremensku $s(t)$.

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} S(\omega)e^{j\omega t}d\omega \qquad [6.2]$$

6.2.3. Diskretna Fourierova transformacija

Diskretna Fourierova transformacija matematički je algoritam koji izračunava diskretnu Fourierovu transformaciju određenog slijeda signala. U odnosu na Fourierovu transformaciju koja uzima neprekidan signal, ulaz diskretne Fourierove tranformacije mora biti definirana diskretnim vrijednostima i analiza se provodi na ograničenom intervalu. S obzirom da se zvučne datoteke sastoje od diskretnih vrijednosti (amplituda) neprekidnog zvučnog signala diskretna Foureirova tranformacija idealna je za procesiranje informacija zvučnih signala.

Definicija: Dani skup od N kompleksnih brojeva $s_0, \dots, s_{N-1}$ je transformiran u niz od N kompleksnih brojeva $S_0, \dots, S_{N-1}$ prema formuli

$$S_k = \sum_{n=0}^{N-1} s_n e^{-\frac{2\pi kni}{N}} \quad k = 0, \dots, N-1 \quad i^2 = -1 \qquad [6.3]$$

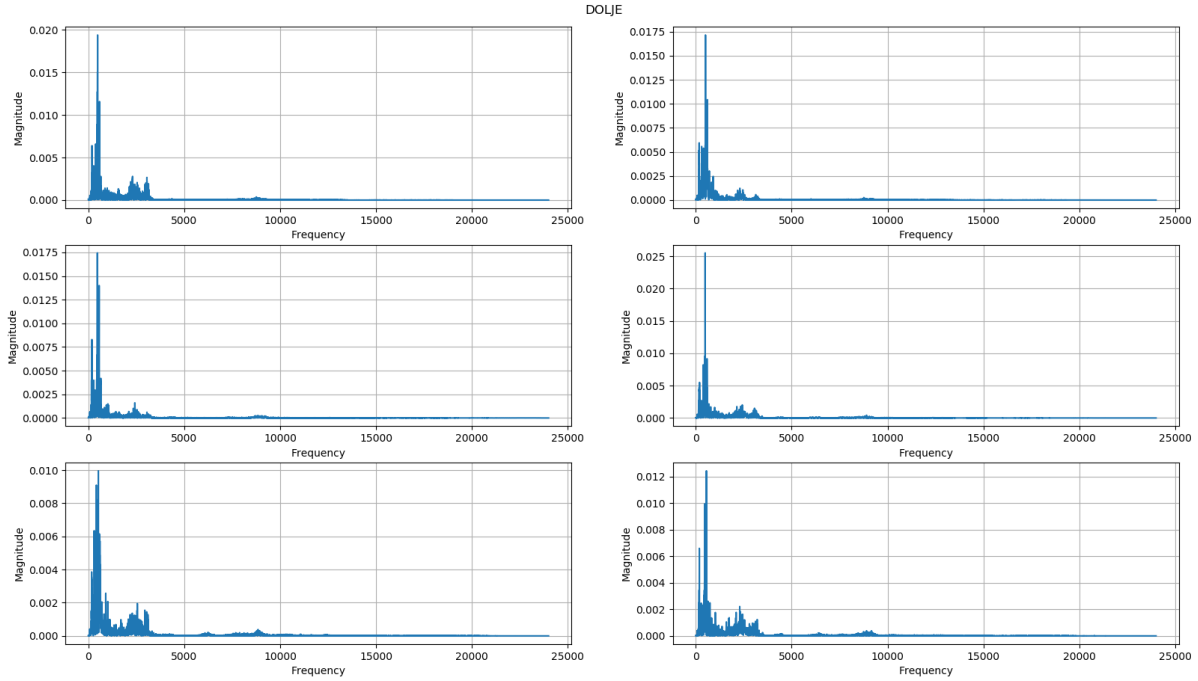
Takvu transformaciju nazivamo diskretna Fourierova transformacija od $s_0, \dots, s_{N-1}$.

Inverzna diskretna Fourierova transformacija dana je izrazom:

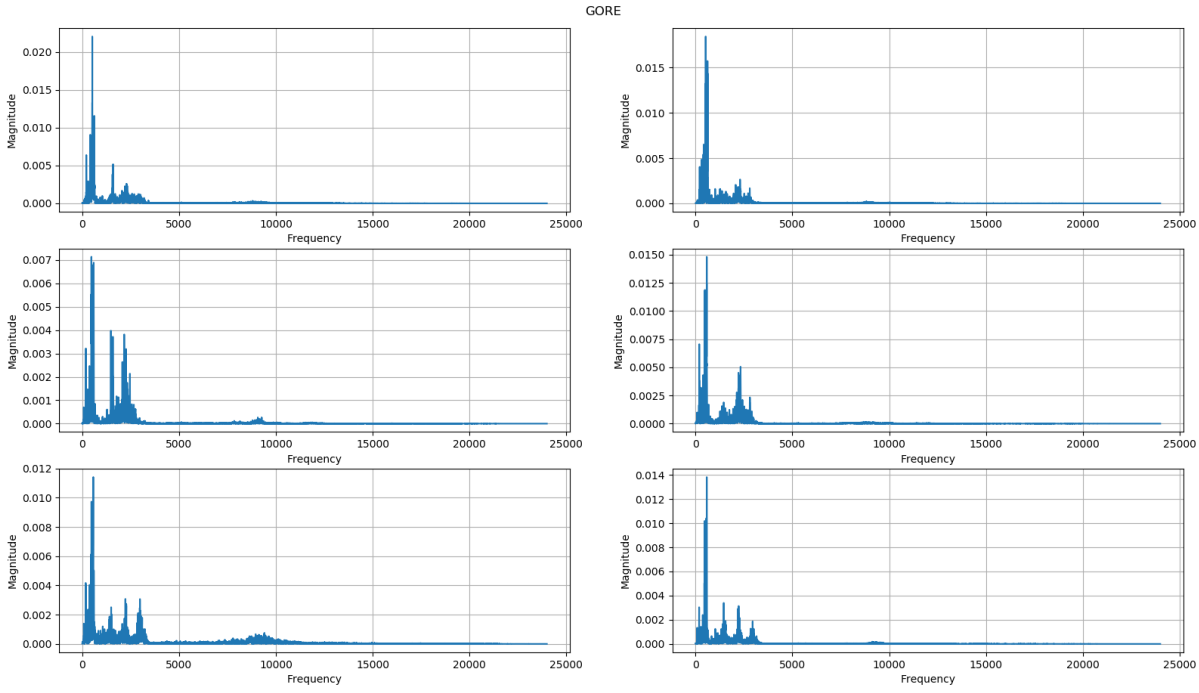
$$s_n = \frac{1}{N}\sum_{k=0}^{N-1} S_k\, e^{\frac{2\pi kni}{N}} \quad n = 0, \dots, N-1 \qquad [6.4]$$

6.2.4. Diskretna Fourierova transformacija zvučnih datoteka

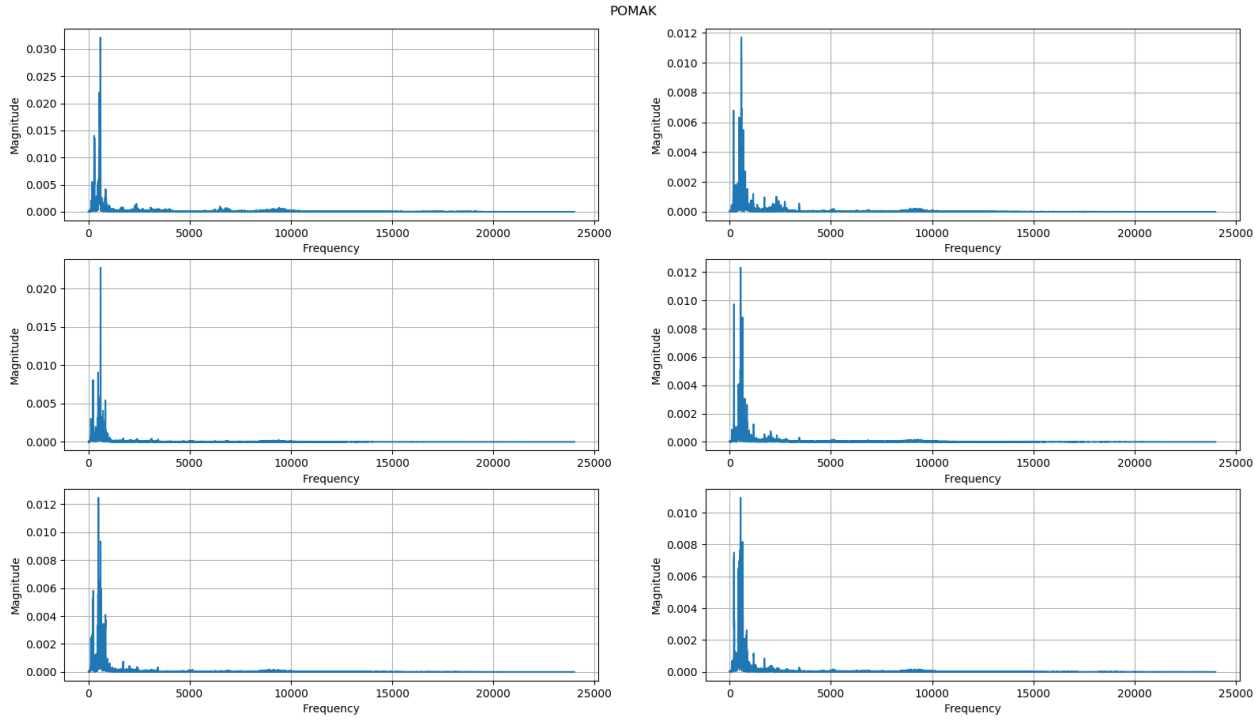
Nakon teorijskog opisa, diskretna Fourierova transformacija primjenit će se na setu zvučnih podataka. Na sljedećim slikama zvučne datoteke prikazane su u frekvencijskoj domeni. Brzina uzorkovanja (engl. sampling rate) zvučnih datoteka iznosi 48000 Hz.



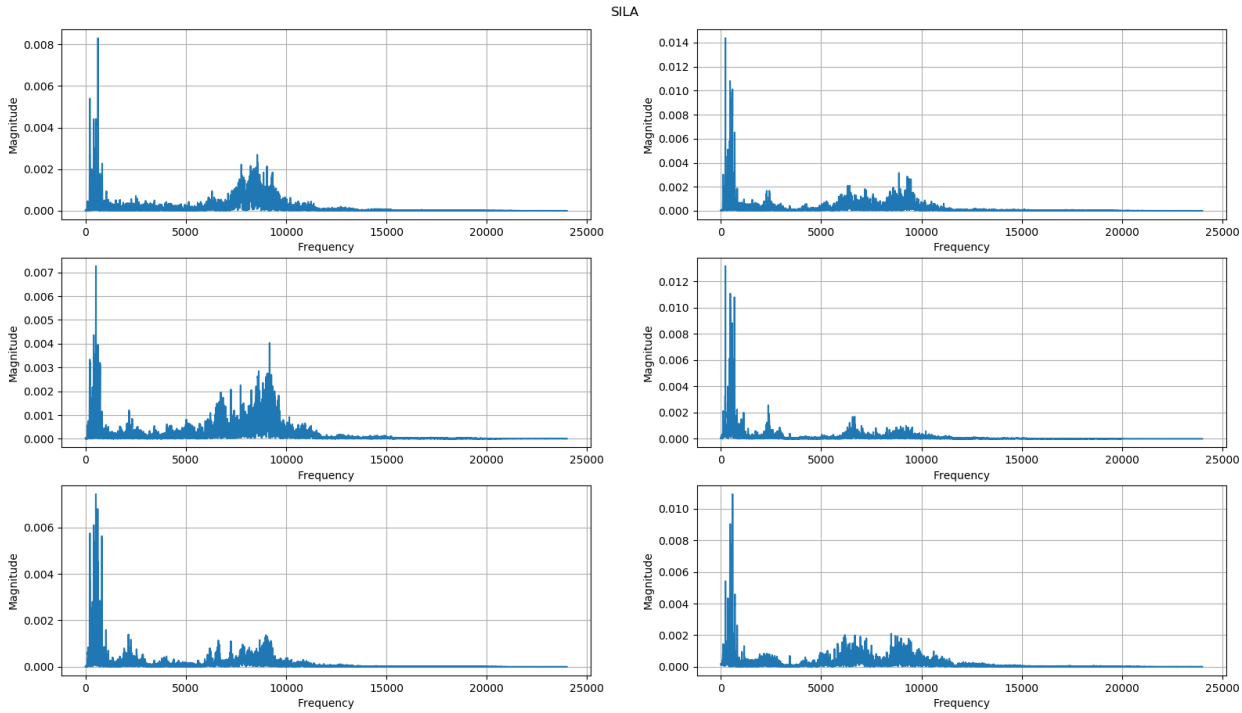
Slika 34. Uzorci riječi „dolje“ u frekvencijskoj domeni



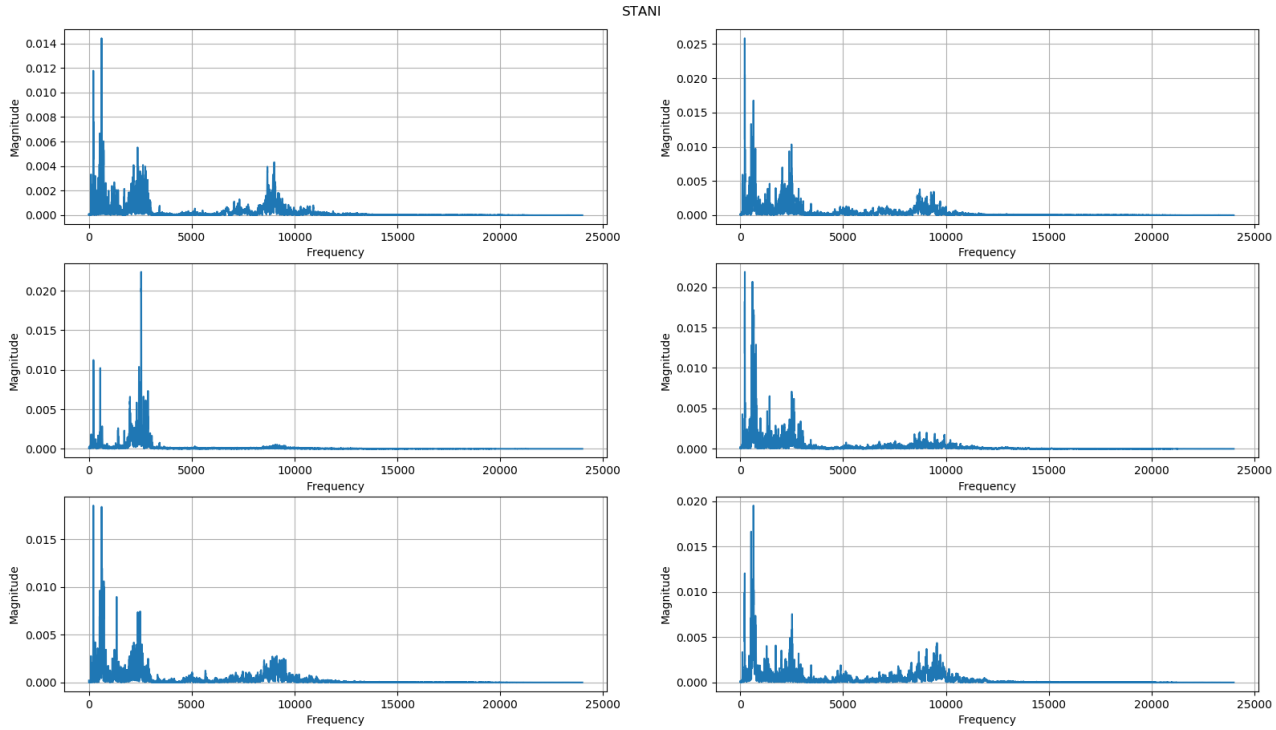
Slika 35. Uzorci riječi „gore“ u frekvencijskoj domeni



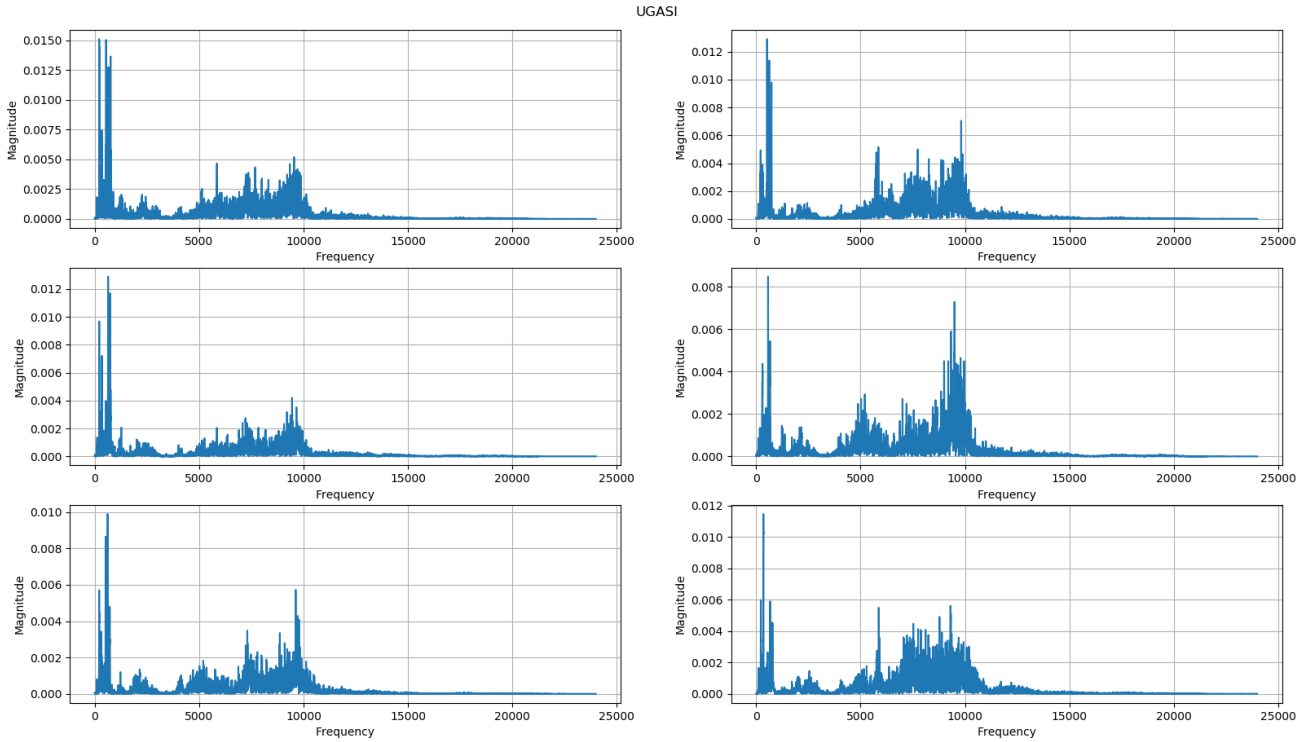
Slika 36. Uzorci riječi „pomak“ u frekvencijskoj domeni



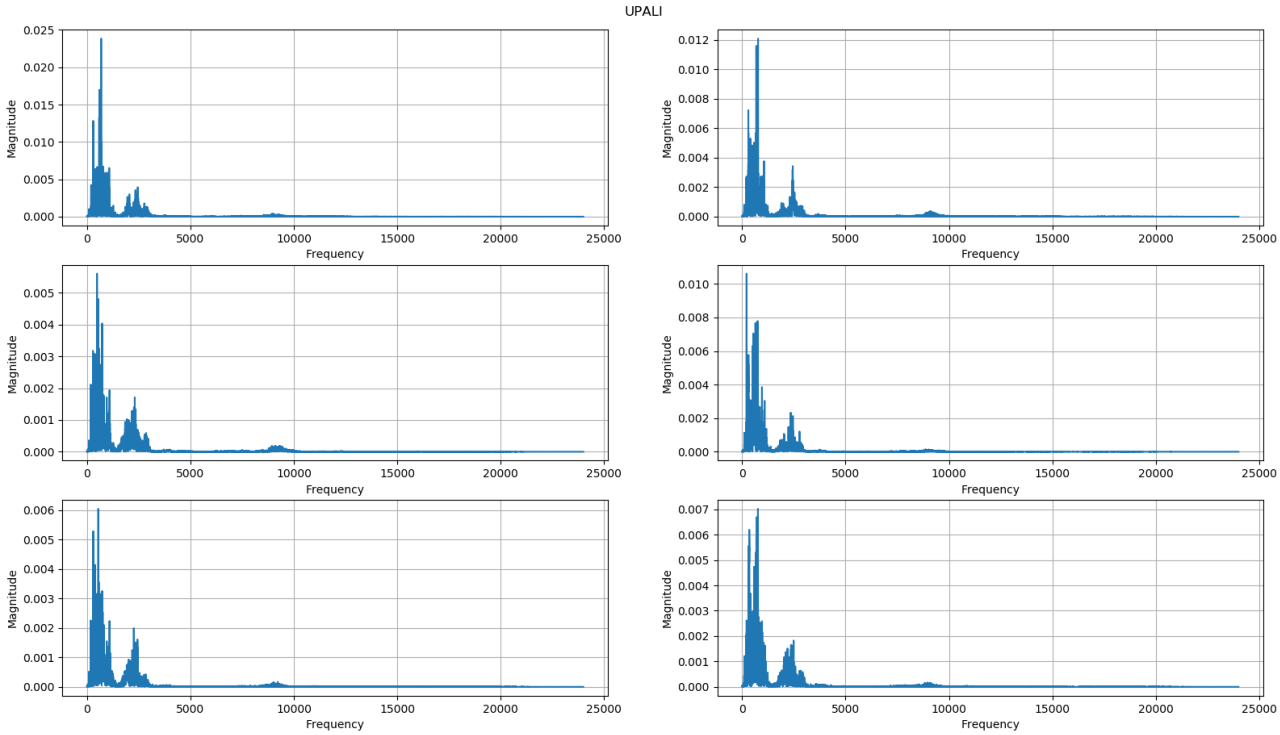
Slika 37. Uzorci riječi „sila“ u frekvencijskoj domeni



Slika 38. Uzorci riječi „stani“ u frekvencijskoj domeni



Slika 39. Uzorci riječi „ugasi“ u frekvencijskoj domeni



Slika 40. Uzorci riječi „upali“ u frekvencijskoj domeni

Iz slika je vidiljivo da se svaki zvučni signal sastoji od tisuće različitih frekvencija. S obzirom da je brzina uzorkovanja signala jednaka 48000 Hz, prema Nyquistovom teoremu raspon frekvencija nalazi se u domeni od 0 do 24000 Hz. Definicija Nyquistova teorema uzorkovanja glasi: „ukoliko kontinuirani signal sadrži frekvencije od 0 do maksimalno f_0 Hz, tada se on u potpunosti može rekonstruirati iz slijeda uniformno udaljenih diskretnih uzoraka koji se pojavljuju s frekvencijom uzorkovanja f_s većom od $2f_0$ Hz.“ Kao i svi veliki teoremi, teorem uzimanja uzoraka krajnje je jednostavan:

$$f_s \geq 2f_0 \qquad [6.5]$$

S obizorm da su zvučni signali u ovom slučaju izgovorene ljudske riječi, noramlno je za očekivati da su jake frekvencije u području između 0 i 1 kHz koje u ljudskome govoru dominiraju.

6.2.5. Spektogram

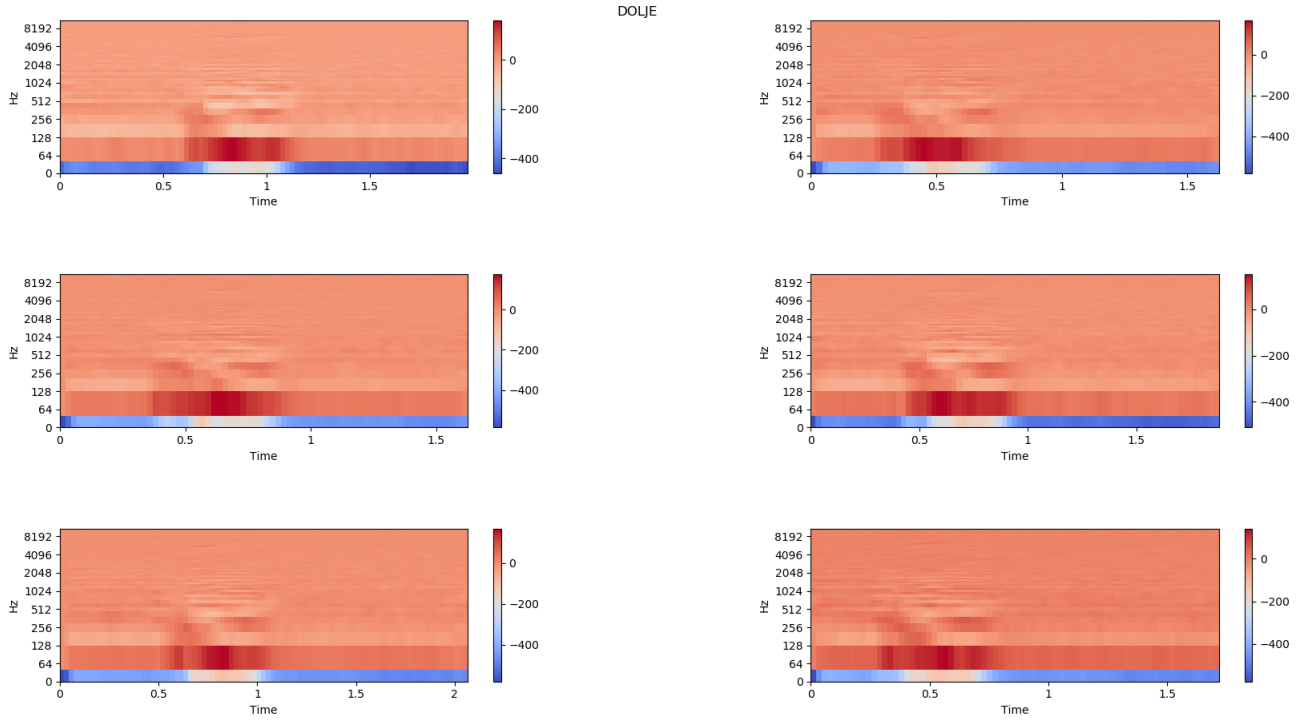
Diskretnom Fourierovom transformacijom signali su prikazani u frekvencijskom području, međutim, informacije o vremenu su izgubljene. Vremenska domena važna je zbog prikaza poretka slogova tj. slova u samoj riječi. Stoga, signale je potrebno prikazati ne samo u frekvencijskoj već i u vremenskoj domeni.

Vizualna reprezentacija frekvencije signala u ovisnosti o vremenu naziva se spektogram. Na apscisnoj osi spektograma prikazana je promjena vremena, a na osi ordinata promjena frekvencije dok su bojom prikazane veličine (amplitude) promatranih frekvenciju u određenom vremenu.

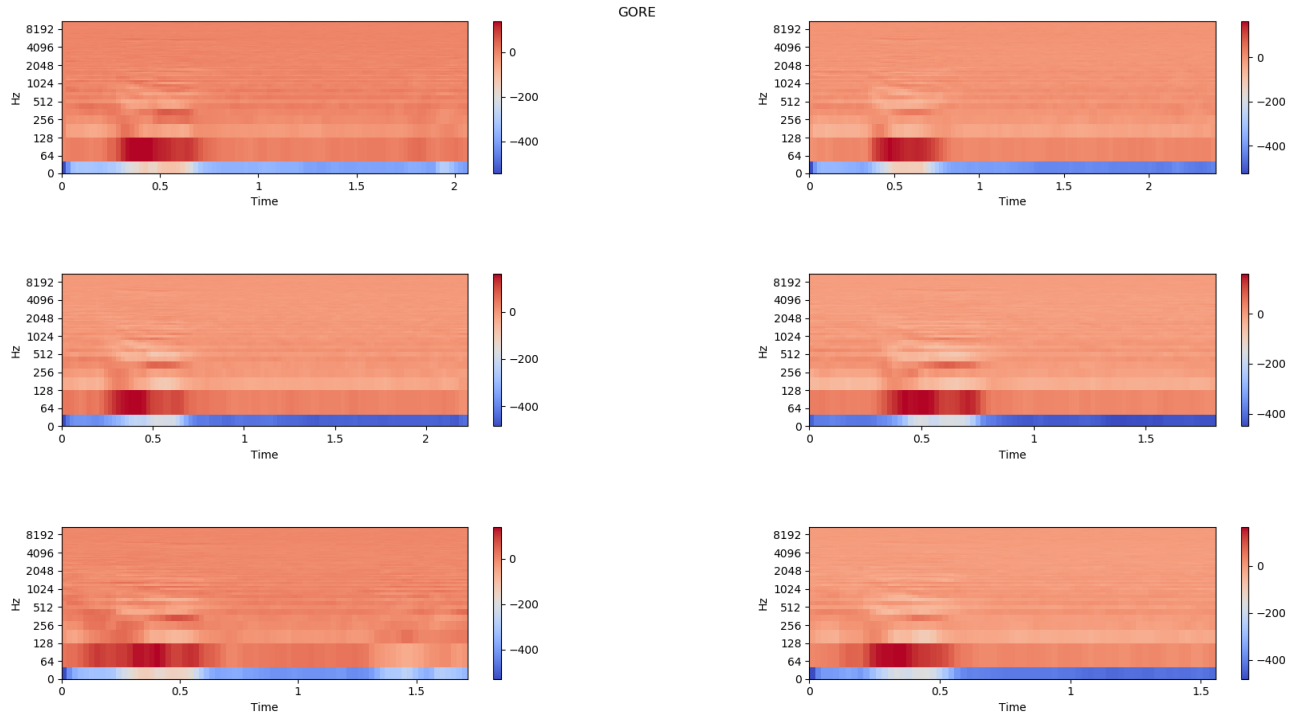
6.2.6. Mel frekvencijski prikaz

Spektogramska reprezentacija zvučnih signala koristi linearno mjerilo frekvencije. Međutim, u analizi zvučnih signala često se spominje pojam mel-frekvencije [6]. Mel-frekvencija koristi kvazi-logaritamsko mjerilo radi što točnijeg opisa ljudskog glasovnog aprata. Algoritam za izračunavanje mel-frekvencijskog kepstralnog koeficijenta (MFCC) je sljedeći:

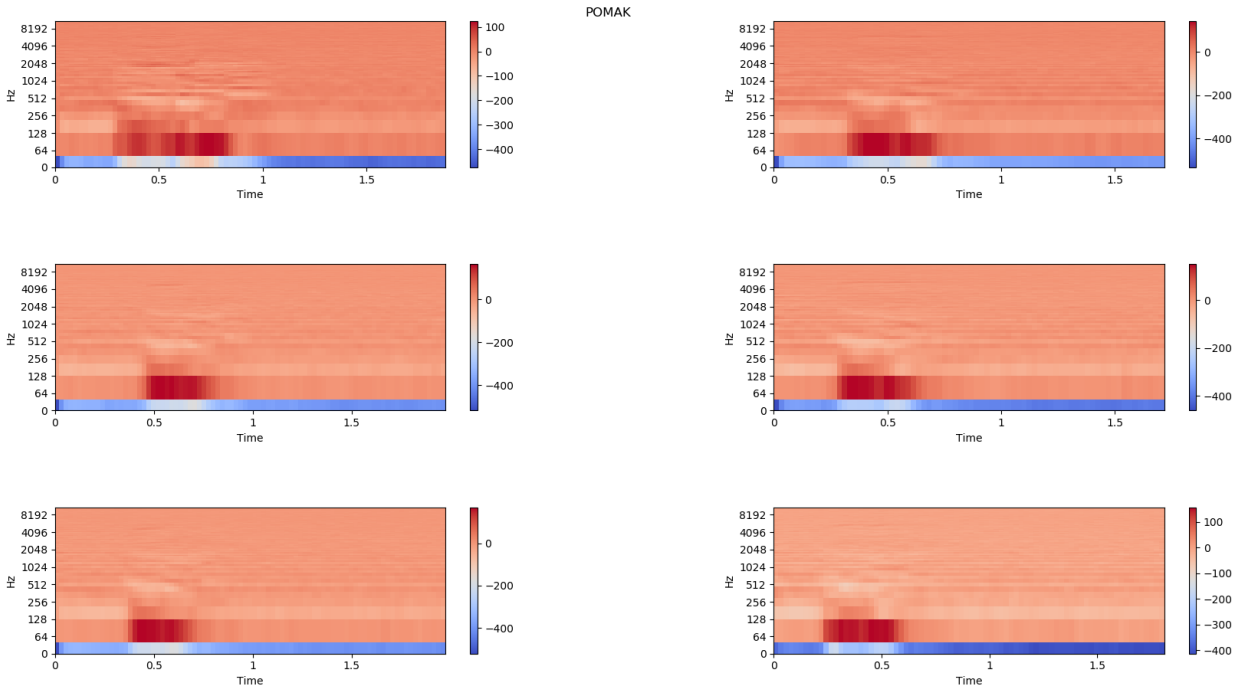
1) Zvučne signale potrebno je raspodijeliti u intervale (u trajanju od 20 do 30 ms s obzirom da čovjek ne može izgovoriti više od jednog fonema unutar tog vremenskog perioda)
2) Diskretnom Fourierovom transformacijom izračunava se spektar snage ulaznog signala
3) Frekvencije se mapiraju iz linearnog mjerila u logaritamsku mel skalu
4) Logaritmira se spektar snage u određenim točkama
5) Izračunava se diskretna kosinusna transformacija logaritamskog spektra snage



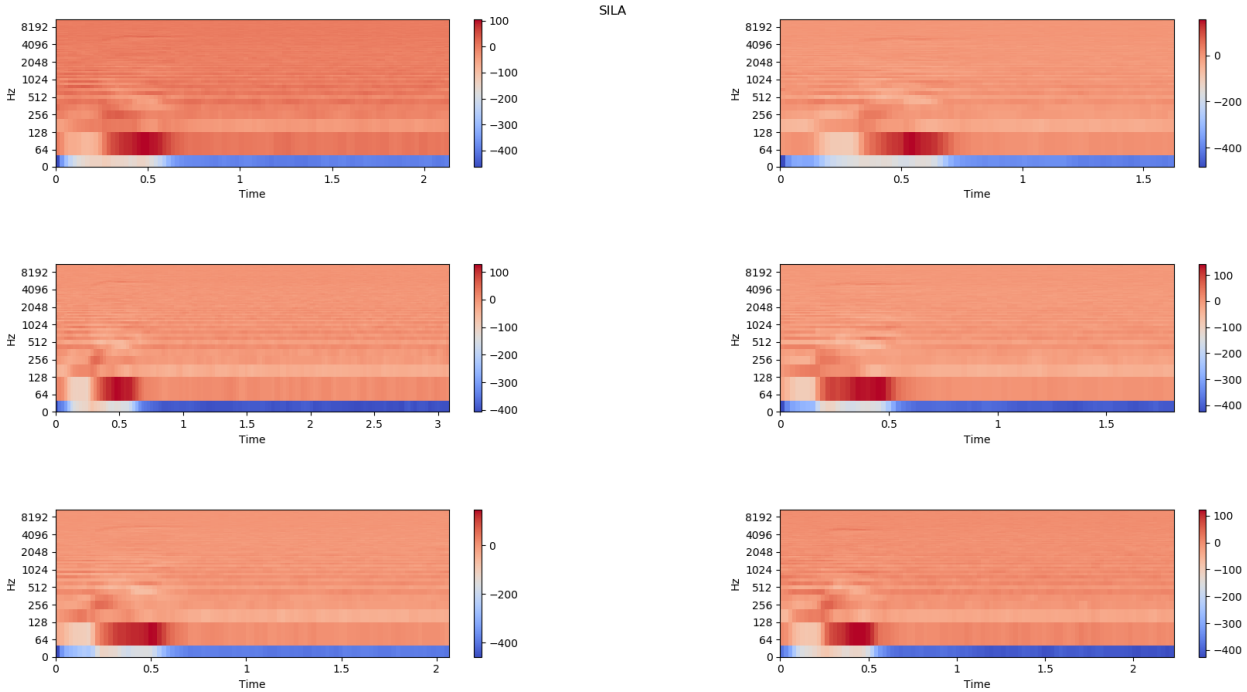
Slika 41. Mel frekvencijski prikaz uzoraka riječi „dolje“



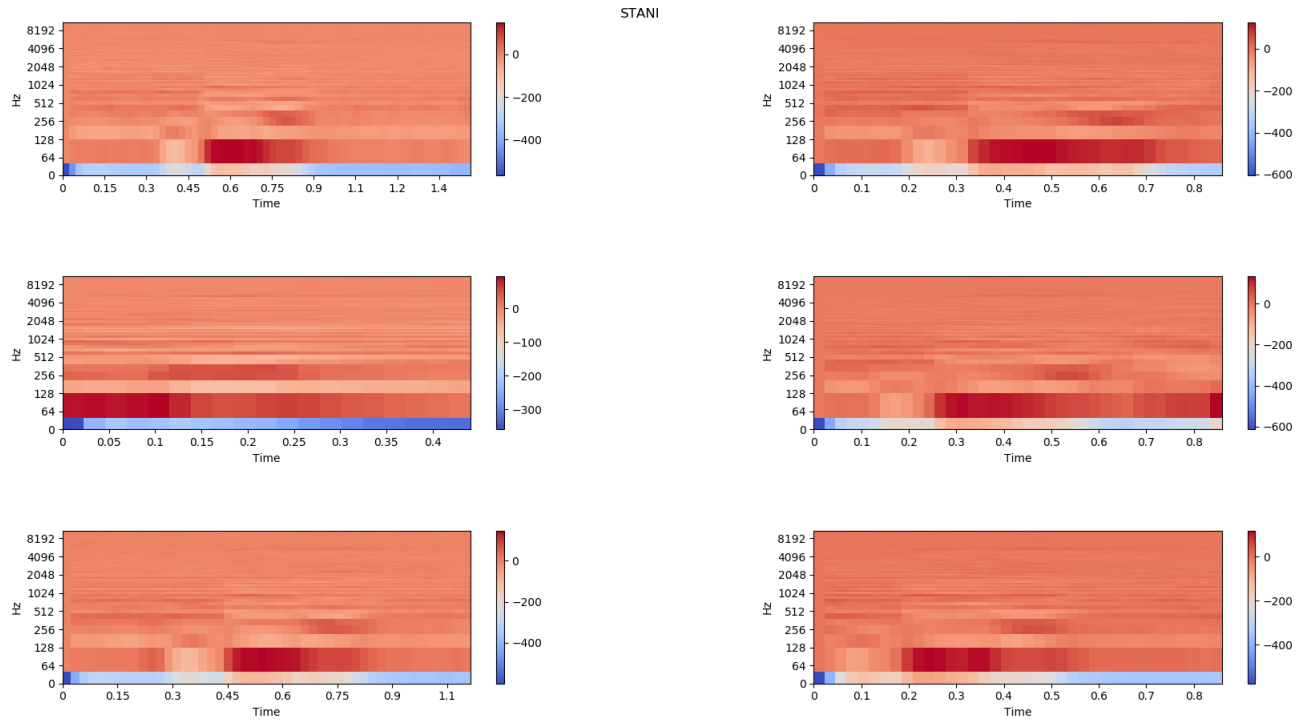
Slika 42. Mel frekvencijski prikaz uzoraka riječi „gore“



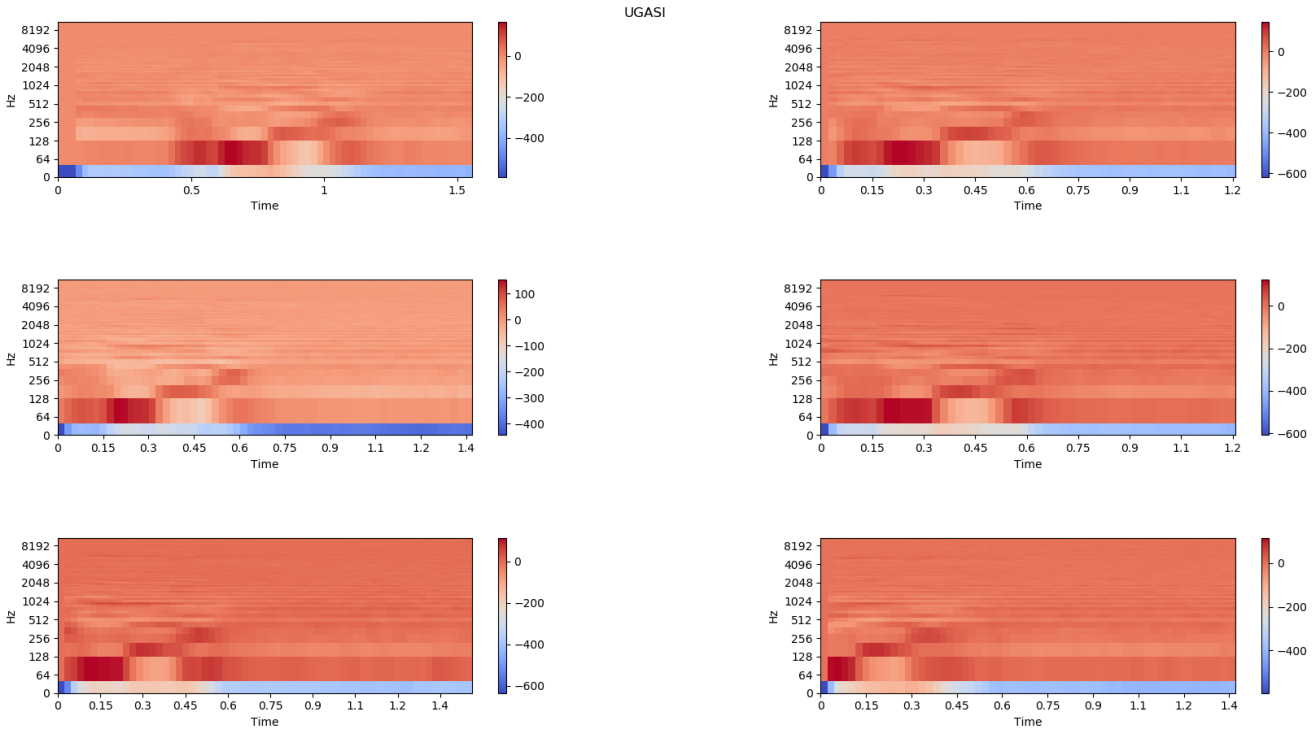
Slika 43. Mel frekvencijski prikaz uzoraka riječi „pomak“



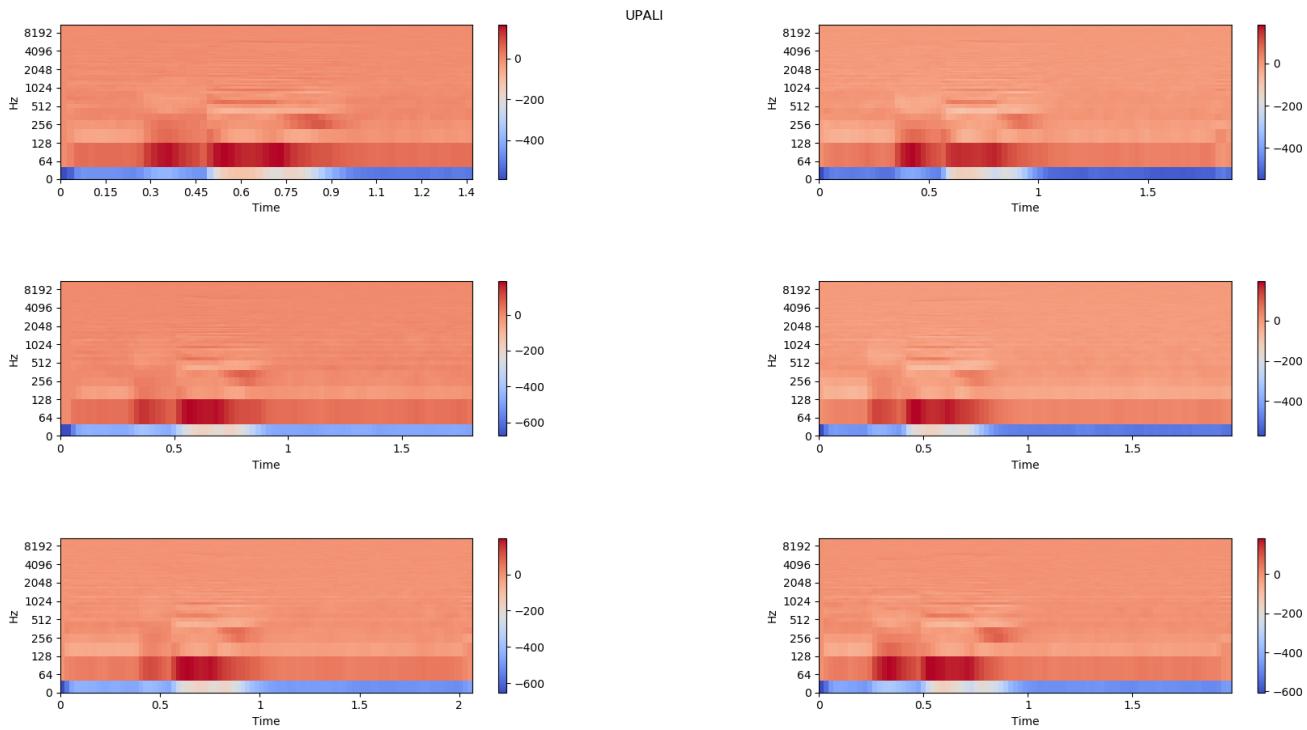
Slika 44. Mel frekvencijski prikaz uzoraka riječi „sila“



Slika 45. Mel frekvencijski prikaz uzoraka riječi „stani“



Slika 46. Mel frekvencijski prikaz uzoraka riječi „ugasi“



Slika 47. Mel frekvencijski prikaz uzoraka riječi „upali“

Time je problem klasifikacije zvučnih datoteka sveden na klasičan problem klasifikacije slika gdje su grafički prikazi zvučnih datoteka u mel frekvencijskom području matrično zapisani te u tom obliku čine ulaz neuronske mreže.

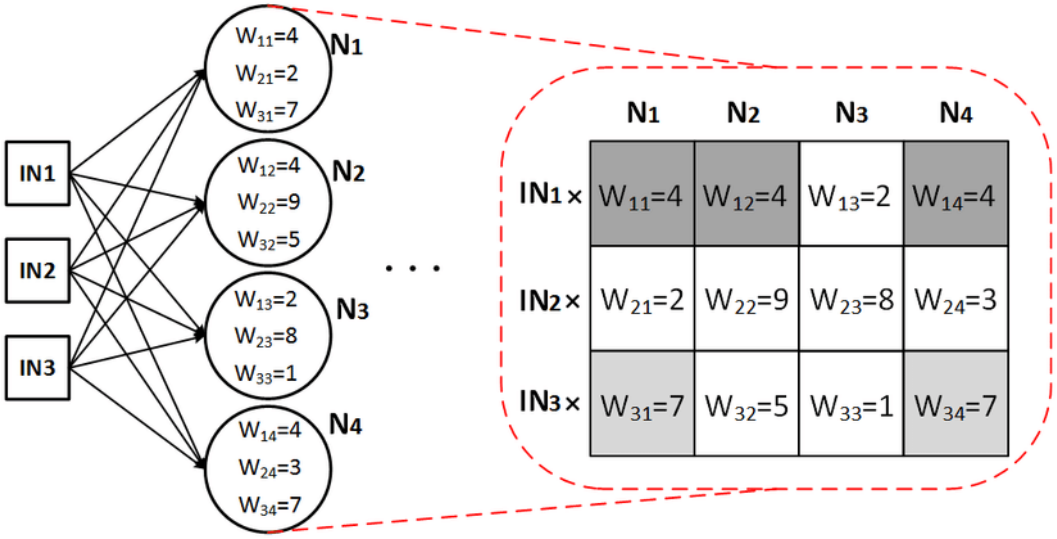
7. UMJETNE NEURONSKE MREŽE

Umjetna neuronska mreža skup je umjetnih neurona međusobno povezanih i interaktivnih kroz operacije obrade signala. Umjetna neuronska mreža sadrži jedan ili više ulaza, ali uvijek jedan aktivacijski izlaz. Između ulaza i izlaza nalaze se jedan (jednoslojna neuronska mreža) ili više (višeslojne mreže) skrivenih slojeva. Neuroni i slojevi međusobno su spojeni vezama koji se aktiviraju zadovoljavanjem određenog uvjeta aktivacijskom funkcijom. Karakteristično za neuronske mreže je implicitno programiranje, tj. mreži se ne definira način na koji treba obrađivati podatke već mrežu učimo na temelju samih podataka. Također, podaci koji se mreži dovode ne moraju biti savršeno precizni ni točni: mreža tijekom učenja razvija sposobnost generalizacije pa će dobro raditi i u situacijama kada su ulazni podatci zagađeni šumom.

Umjetne neuronske mreže trenutačno su dominantne u području klasifikacije te funkcijske regresije. Klasifikacija je postupak raspodjele ulaznih uzoraka na klase (razrede) što je i cilj implementacije neuronske mreže u ovome radu: glasovno upravljanje pneumatskom prešom klasifikacijom određenih riječi [7].

7.1. Težine i pragovi

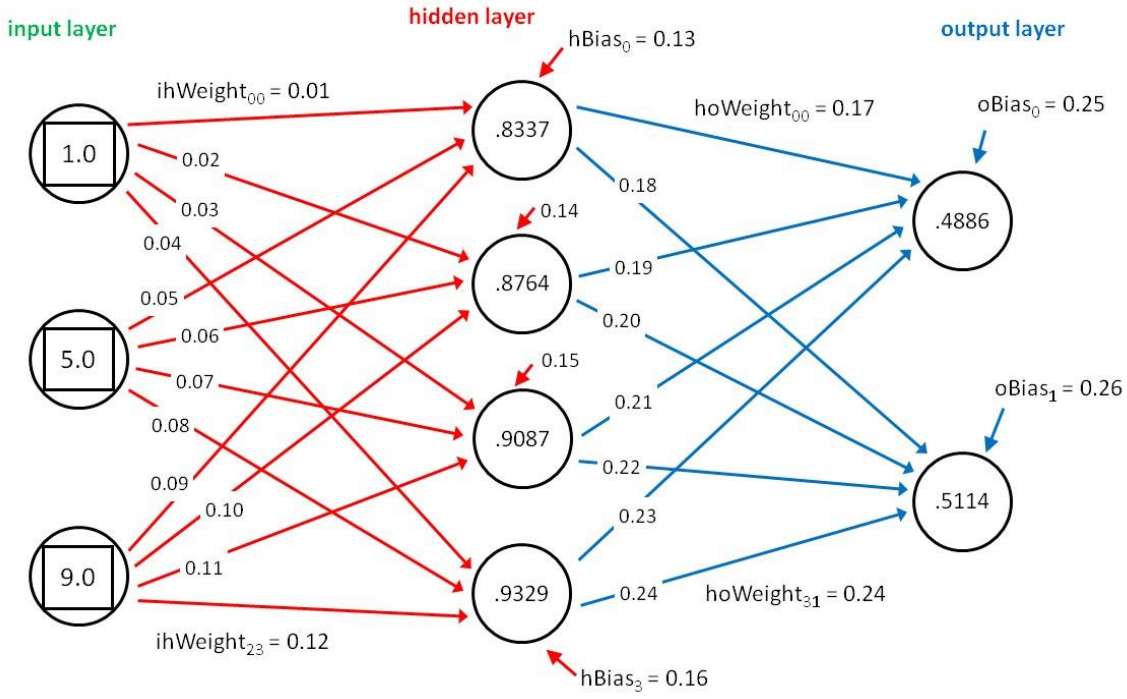
Težina (engl. weight) je veza između neurona koja posjeduje određenu brojčanu vrijednost. Što je vrijednost težine veća, više značaja pridodajemo neuronu sa ulazne strane težine. Važno je napomenuti da su prilikom računanja težine prikazane u matričnom obliku kao što je prikazano na slici 48.



Slika 48. Težine u neuronskoj mreži

Na prikazanoj slici ulazni sloj neuronske mreže sadrži 3 neurona, dok sljedeći sloj (skriveni sloj) sadrži 4 neurona. Na temelju tih spoznaja možemo definirati matricu sa 3 reda i 4 stupca sa upisanim vrijednostima težina u svako polje matrice. Povećanjem broja slojeva neuronske mreže povećava se broj definiranih matrica. Općenito vrijedi da ako sloj L sadrži N neurona, a sljedeći sloj (L+1) sardrži M neurona, matrica težine biti će dimenzije NxM.

Prag (engl. bias), baš kao i težina posjeduje određenu vrijedonst. Svaki neuron koji se ne nalazi u ulaznome sloju sadrži prag. Prag je dodatni parametar unutar neuronske mreže koji u obzir uzima dodatne nepredviđene ili neprimjetne faktore.

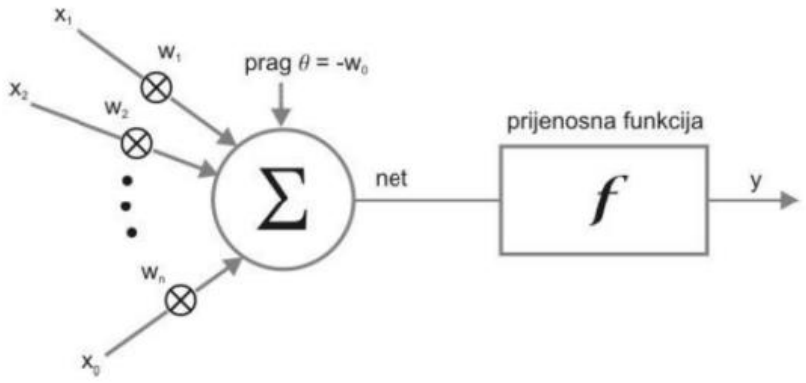


Slika 49. Pragovi u neuronskoj mreži

Na slici 49 prikazani su pragovi u skrivenom i izlaznom sloju, gdje je uz svaki neuron prikazana crvena odnosno plava strelica koju izvorni neuroni ne posjeduju. Pomake, isto kao i težine prikazujemo u obliku matrica (odnosno vektora). Vektor pomaka skrivenog sloja na gornjoj slici glasi: b = [0.13, 0.14, 0.15, 0.16].

7.2. Aktivacijske funkcije

Svaki neuron posjeduje aktivacijsku funkciju. Aktivacijska funkcija može se definirati kao način na koji neuron donosi odluku. Aktivacijska funkcija uzima sumu umnoška ulaza neurona s pripadnim težinama i pragovima te ih preslikava na izlaz neurona koji modelira signal na aksonu (y).



Slika 50. Umjetni neuron

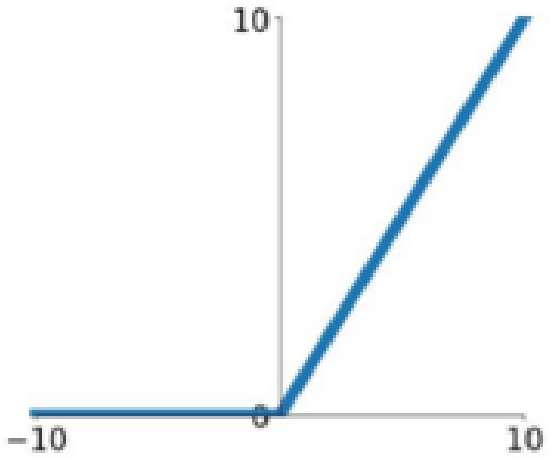
Ulazi neurona označeni su sa $x_1, x_2, \ldots\ x_n$, njihove pripadne težine sa $w_1, w_2, \ldots\ w_n$, a prag sa θ. Izlaz neurona može se zapisati kao:

$$y = f\left(\theta + \sum_{i=1}^{n} x_i w_i\right) \qquad [7.1]$$

Postoji više izbora aktivacijskih funkcija $f(z)$, gdje je z skup svih ulaza.U nastavku prikazane su najpopularnije:

Linearna rektifikacijska funkcija (ReLU) – osigurava da izlaz neurona ne zahvaća vrijednosti manje od 0. Ukoliko je z veće od 0, izlaz ostaje z, a ako je z negativan izlaz je jednak 0.

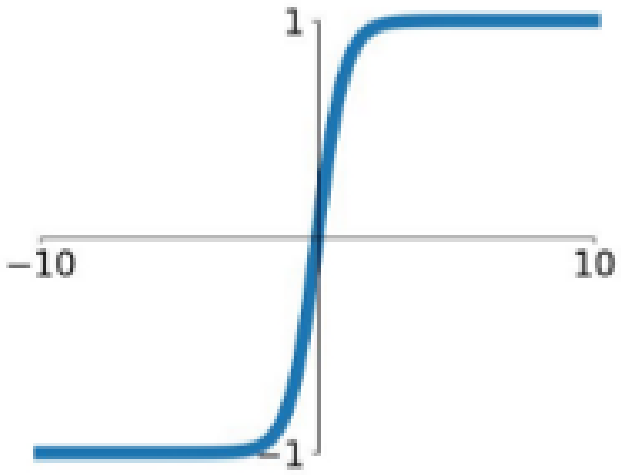
$$f(z) = \max(0, z) \qquad [7.2]$$



Slika 51. Linearna rektifikacijska funkcija

Tanges hiperbolični – neparna, monotono rastuća funkcija. Izlaz funkcije jednak je hiperboličnom tangensu skupa svih ulaza (z).

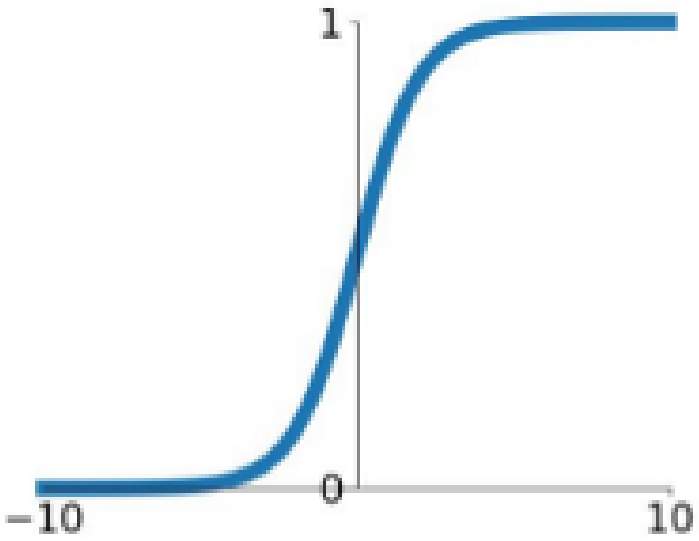
$$f(z) = \tanh(z) \qquad [7.3]$$



Slika 52. Funkcija tanges hiperbolični

Sigmoidalna funkcija – derivabilna je na cijeloj domeni i ograničena što su dobra svojstva za algoritam unazadne propagacije i učenja mreže.

$$f(z) = \frac{1}{1 - e^{-kz}} \qquad [7.4]$$



Slika 53. Sigmoidalna funkcija

7.3. Matematička pozadina neuronske mreže

S obzirom da neuronske mreže imaju i po milijune parametara (težina i pragova) koje je potrebno optimirati kako bi mreža radila što je moguće bolju klasifikaciju, nužno je znati kako te parametre odrediti. U ovom potpoglavlju prikazane su osnovne matematičke operacije u neuronskim mrežama po koracima [8].

Korak 1: Za svaki neuron, pomnoži ulaznu vrijednost x_i sa težinom w_i te zbroji sve pomnožene vrijednosti.

$$x.w = (x_1 w_1) + (x_2 w_2) + \cdots + (x_n w_n) \qquad [7.5]$$

Korak 2: dodaj prag u sumu pomnoženih vrijednosti, a ukupna suma jednaka je skupu svih ulaza z:

$$z = x.w + b \qquad [7.6]$$

Korak 3: proslijedi vrijednost z aktivacijskoj funkciji (npr. Sigmoidalnoj):

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}} \qquad [7.7]$$

7.3.1. Backpropagation algoritam

Za što bolju funkcionalnost neuronske mreže potrebno je minimizirati pogrešku mreže što se svodi na pretraživanje n-dimenzionalnog prostora, gdje je n ukupan broj težina u mreži. Ideja algoritma backpropagation jest u svakom sloju neuronske mreže odrediti greške i gradijente te na temelju gradijenata ažurirati vrijednosti težina čime se automatski smanjujuje greška neuronske mreže.

Korak 1: Funkcijom pogreške (engl. Cost function) određuje se greška izlaznog sloja. Funkcija pogreške računa se za cijeli skup podataka predviđen za treniranje. Prosjek skupa podataka definiran je funkcijom srednje kvadratne pogreške (engl. Mean Squared Error):

$$C = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad [7.8]$$

gdje su:

y_i - očekivani izlaz i-tog neurona,
$\hat{y}_i$ - trenutačni izlaz i-tog neurona.

Korak 2: Kako bi našli najbolje moguće težine i pragove, potrebno je poznavati kako se funkcija pogreške mijenja u odnosu na promjenu težina i pragova. Drugim riječima, potrebno je pronaći gradijent funkcije troška s obzirom na težine i pragove.

Budući da funkcija troška nije izravno povezana sa težinom, odnos je definiran lančanim pravilom:

$$\frac{\partial \mathrm{C}}{\partial w_i} = \frac{\partial \mathrm{C}}{\partial \hat{\mathrm{y}}} \times \frac{\partial \hat{\mathrm{y}}}{\partial \mathrm{z}} \times \frac{\partial \mathrm{z}}{\partial w_i} \qquad [7.9]$$

Gdje su:

$$\frac{\partial \mathrm{C}}{\partial \hat{\mathrm{y}}} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \qquad [7.10]$$

$$\frac{\partial \hat{\mathrm{y}}}{\partial z} = \sigma(z)[1 - \sigma(z)] \qquad [7.11]$$

$$\frac{\partial z}{\partial w_i} = x_i \qquad [7.12]$$

Uvrštavanjem izraza [7.9], [7.10] i [7.11] u izraz [7.12], slijedi:

$$\frac{\partial \mathrm{C}}{\partial w_i} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{\mathrm{y}}_i) \times \sigma(z)[1 - \sigma(z)] \times x_i, \qquad [7.13]$$

a s obzirom da je teoretska vrijdenost praga jednaka 1 slijedi:

$$\frac{\partial \mathrm{C}}{\partial \mathrm{b}} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{\mathrm{y}}_i) \times \sigma(z)[1 - \sigma(z)]. \qquad [7.14]$$

7.3.2. Optimizacija

Optimizacija je postupak izbora najbolje težine i praga samog perceptrona koristeći gradijentni spust kao optimizacijski algoritam. Gradijentni spust mijenja vrijednosti težina i pragova proporcionalno negativnoj vrijednosti gradijenta funkcije troška. Faktor učenja α (engl. Learning rate) je hiperparametar koji se koristi kao kontrola koliko se težine i pragovi mijenjaju. Backpropagation i gradijentni spust ponavljaju se do konvergencije rješenja. Ažuriranje težina i pragova definirani su sljedećim izrazima:

$$w_i = w_i - \left(\alpha \times \frac{\partial \mathrm{C}}{\partial w_i}\right) \qquad [7.15]$$

$$b = b - \left(\alpha \times \frac{\partial \mathrm{C}}{\partial \mathrm{b}}\right) \qquad [7.16]$$

7.4. Konvolucijske neuronske mreže

Konvolucijska neuronska mreža vrsta je duboke neuronske mreže najčešće korištena pri klasifikaciji slika. Naziv „konvolucijska neuronska mreža“ ukazuje da mreža primjenjuje matematičku operaciju koja se zove konvolucija. Konvolucija je specifčna vrsta linearne operacije koja se koristi na mjestu klasičnog matričnog množenja u bar jednom od slojeva mreže. Neuroni u konvolucijskim neuronskim mrežama su dvodimenzionalni i nazivamo ih mapama značajki (engl. Feature maps). Ulaz konvolucijske neuronske mreže također je dvodimenzionalan, a umjesto težina koriste se jezgre (engl. Kernels).

Građa konvolucijske neuronske mreže sastoji se od tri različite vrste slojeva: konvolucijski slojevi (engl. Convolutional layers), slojevi sažimanja (engl. Max-Pooling layers) i potpuno povezani slojevi (engl. Fully-Connected layers) [9].

7.4.1. Konvolucijski sloj

Konvolucijski slojevi provode 2D konvoluciju s jezgrama uzimajući mape na ulazu sloja. Konvolucija se provodi prolazom kroz ulaznu mapu sa prozorom uz uvjet da je jednake veličine kao i jezgra, te se vrijednosti ulazne mape unutar prozora množe sa odgovarajućim vrijednostima jezgre.

7.4.2. Slojevi sažimanja

Funkcija slojeva sažimanja jest smanjenje dimenzija mapi značajki te uklanjanje varijance. Slojevi sažimanja ne posjeduju parametre koji se mogu učiti. Način sažimanja mape provodi se tako da se okvir predstavi s jednom vrijednošću te se najčešće pomiče na način da se svaka vrijednost iz mape značajki koristi u samo jednom sažimanju.

7.4.3. Potpuno povezani slojevi

Potpuno povezani slojevi promatraju se kao specijalni tipovi konvolucijsih slojeva gdje su sve mape značajki i sve jezgre veličine 1x1. Svaka mapa značajki povezana je sa svim mapama značajki iz prethodnog sloja. Izlazna jedinica je vektor rezultata. Za binarnu klasfikaciju, izlazni sloj generira samo jednu vrijednost.

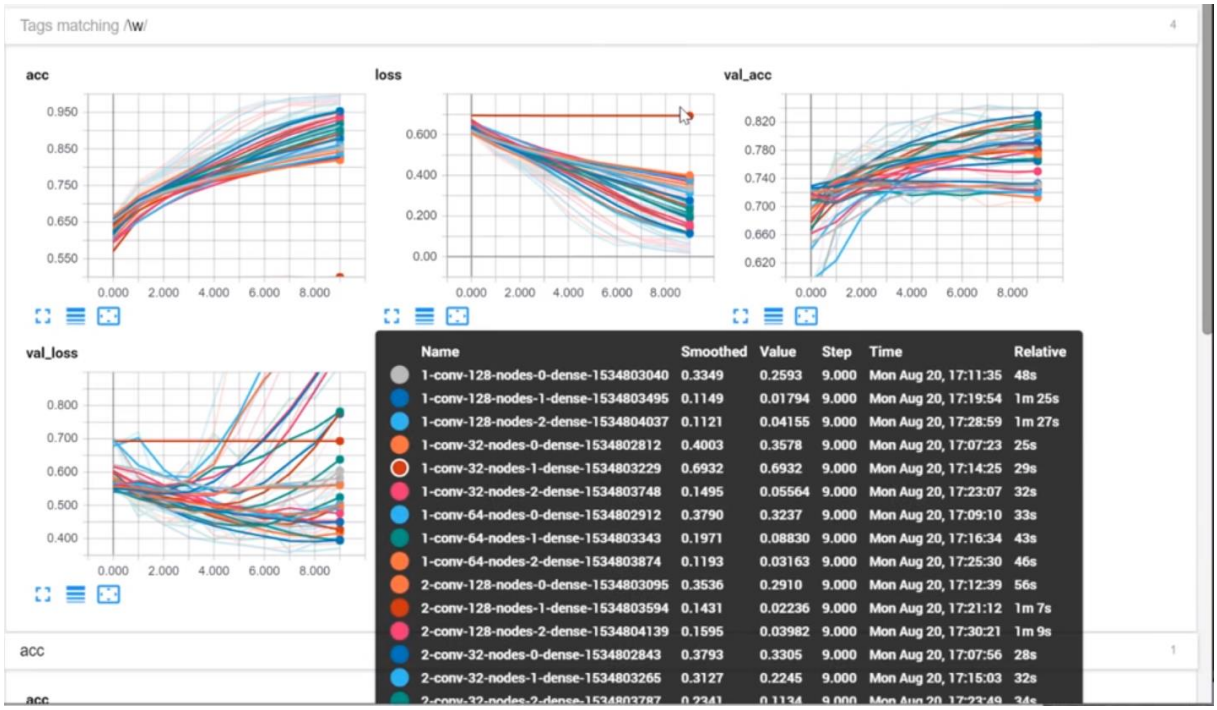
7.5. Treniranje neuronske mreže

Proces treniranja neuronske mreže ponovljen je za više različitih struktura gdje su se mjenjali sljedeći parametri:

- vrsta slojeva,
- broj slojeva,
- broj čvorova,
- aktivacijske funkcije.

- iznosi dropout-a

Različite strukture međusobno su uspoređene te je odabrana neuronska mreža sa najmanjim gubicima (najvećom točnosti).



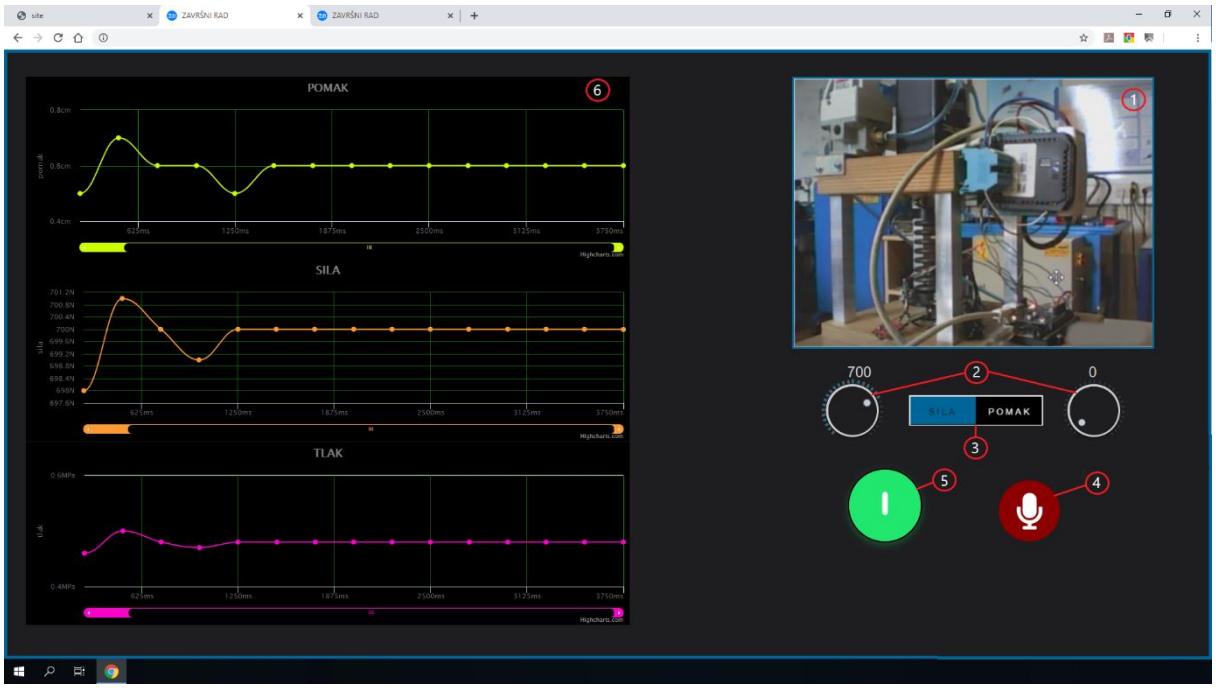
Slika 54. Usporedba neuronskih mreža

8. IZRADA WEB PORTALA

Kako bi korisnik mogao upravljati pneumatskom prešom putem WEB-a nužna je izrada WEB portala gdje će korisnik definirati željene parametre i tako započeti komunikaciju sa mikrokontrolerom. Također, potrebno je onemogućiti istovremeno upravljanje pneumatskom prešom od strane više korisnika odnosno samo jedan korisnik u jednom terminu ima pravo upravljati jednom pneumatskom prešom. Ukoliko se u laboratoriju nalazi više pneumatskih preša, potrebno je osigurati da n korisnika može upravljati n pneumatskim prešama u jednome terminu, pri čemu svaki korisnik upravlja jednom pneumatskom prešom. Kao odgovor na navedni problem osmišljena je WEB aplikacija kalendara. Za izradu WEB portala na back-endu je korišten Python programski jezik sa web2py framework-om te SQLite baza podataka dok je na font-endu uz tri standardne tehnologije (HTML, CSS, JavaScript) korištena Bootstrap biblioteka (HTML + CSS) te jQuery biblioteka (JavaScript).

8.1. Upravljačko sučelje

Upravljačko sučelje prikazano je na slici 55.



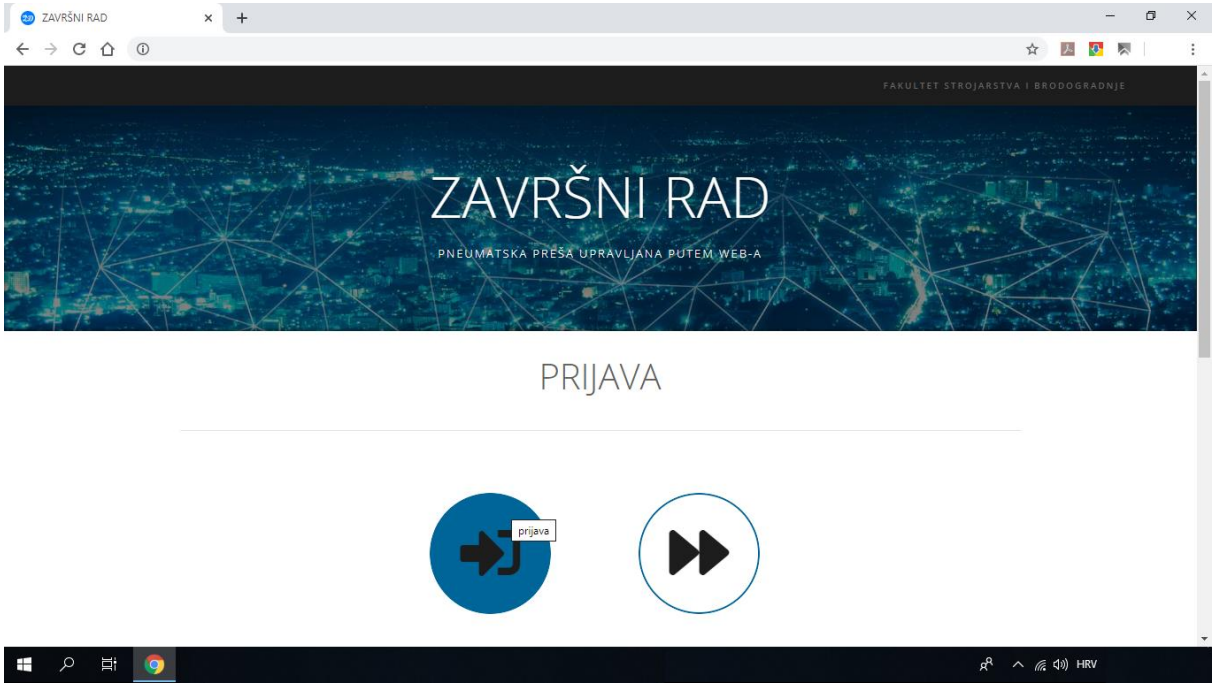
Slika 55. Upravljačko sučelje

Korisnik u realnome vremenu putem video prijenosa može pratiti pneumatsku prešu putem video prozora (1). Vrsta regulacije zadaje se klikom na željeno dugme (3) ovisno želi li se

provesti regulacija sile ili pomaka. S obzirom na vrstu regulacije zadaje se referentna vrijednost (2). Maksimalna referentna vrijednost pomaka iznosi 2,1 cm, dok maksimalna referentna vrijednost sile iznosi 1000 N. Klikom na dugme (5) diktira se gašenje i paljenje pneumatske preše. Također, svim ovim opcijama može se glasovno upravljati klikom na dugme (4) te izgovaranjem jedne od sljedećih riječi: „upali“, „ugasi“, „sila“, „pomak“, „gore“, „dolje“, „stani“.

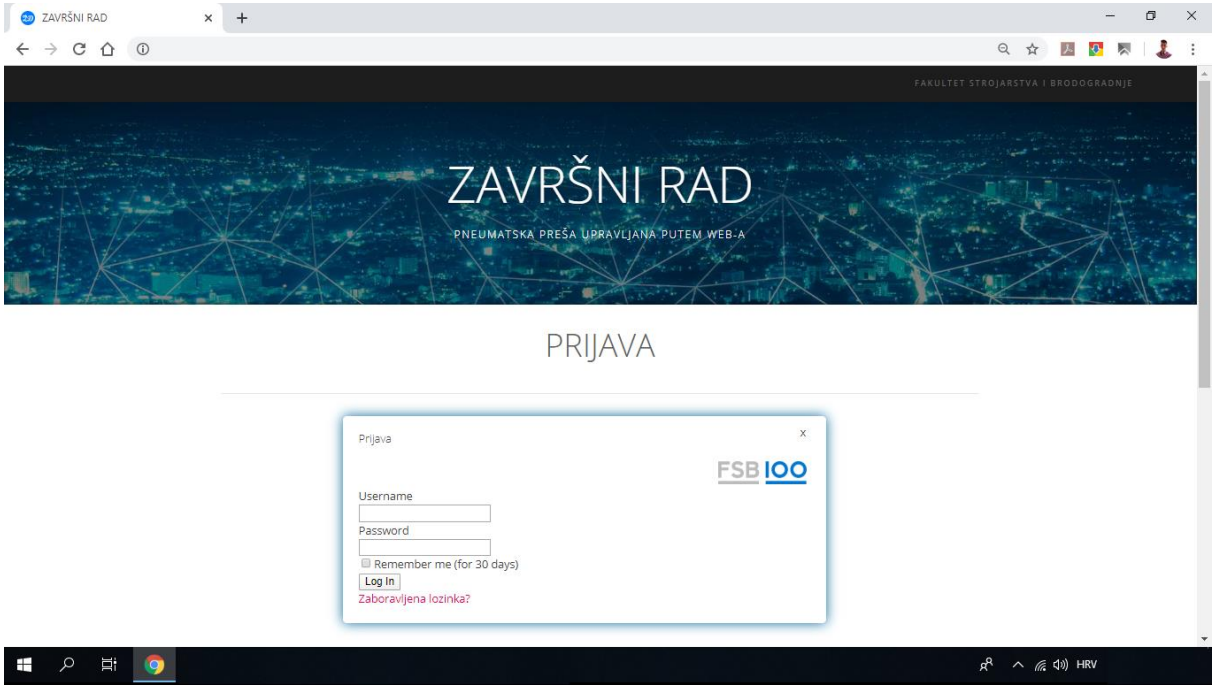
8.2. Sustav prijave

Kako bi korisnik mogao pristupiti korisničkom sučelju, mora izvršiti prijavu klikom na dugme „prijava“ koji se nalazi na naslovnoj stranici.



Slika 56. Naslovna stranica

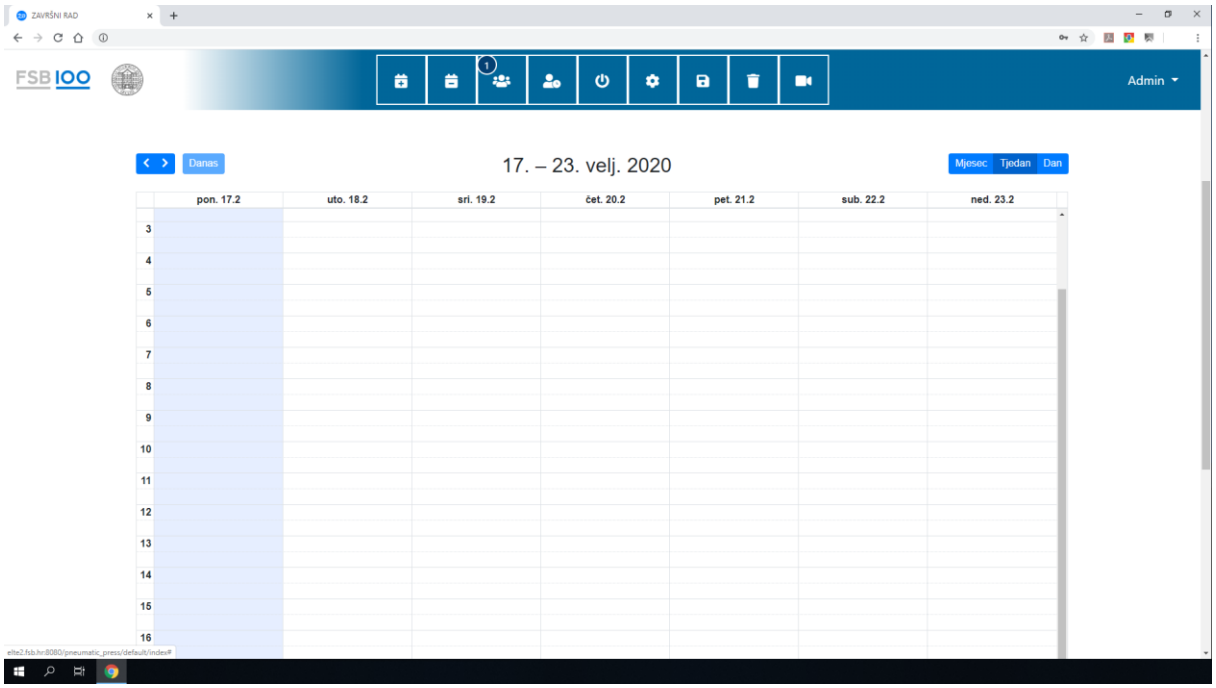
Klikom na dugme prikazuje se prozor u koji se korisnik prijavljuje. Moguće su dvije vrste prijave: kao administrator i kao korisnik čime se otvaraju 2 različita sučelja.



Slika 57. Sustav prijave

8.3. Administratorsko sučelje

WEB aplikacija kalendara zamišljena je na način da administrator (profesor) zadaje termine, a studentima se nudi mogućnost izbora jednog od zadanih termina. Nakon prijave u WEB aplikaciju otvara se administratorsko sučelje u kojem se administratoru nudi mogućnost generiranja termina upravljanja pneumatske preše kao i pregled trenutne situacije sustava te pregled studenata čiji termin je prošao. Cilj je bilo osmisliti što jednostavniji princip generiranja termina što je omogućeno kroz niz opcija koje se nalaze u gornjoj naslovnoj traci. Napomena je da su za neke opcije potrebne kombinacije tipki miša i znakova na tipkovnici što će biti napomenuto u nastavku, u kojem će svaka opcija biti posebno opisana.

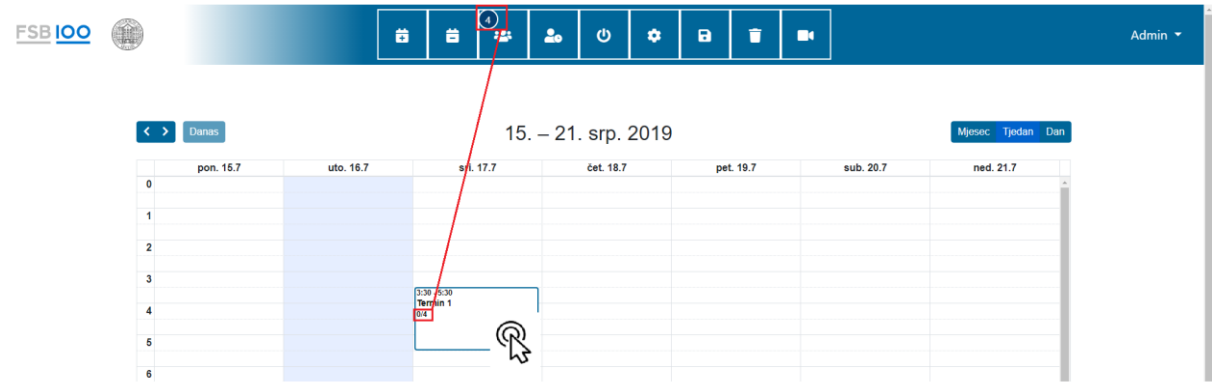


Slika 58. Naslovna stranica administratorskog sučelja

8.3.1. Kreiranje termina

8.3.1.1. Kreiranje pojedinog termina

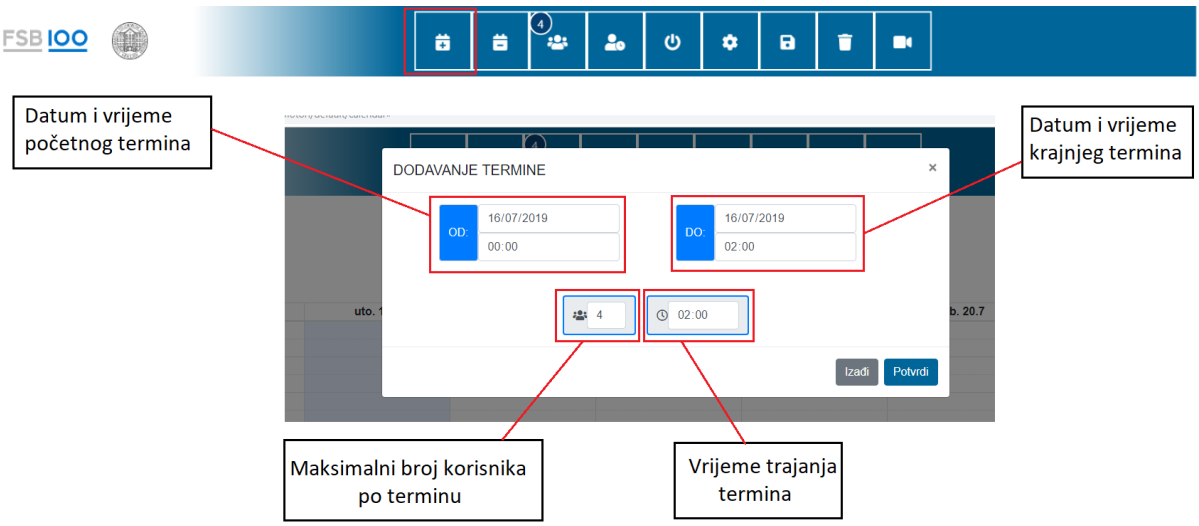
Za kreiranje pojedinog termina potrebno je izvršiti dvoklik na kalendar. Početno vrijeme termina ovisit će o mjestu samog klika, a maksimalni broj korisnika ovisti će o postavljenoj vrijednosti koja se nalazi u izborniku (u početku ona je jednaka broju aktivnih pneumatskih preša).



Slika 59. Kreiranje pojedinog termina

8.3.1.2. Kreiranje seta termina

Za kreiranje više termina u rasponu od početnog do konačnog sa mogućnosti definiranja broja korisnika i vremena trajanja svih termina potrebno je kliknuti na prvu ikonu u izborniku čime se otvara prozor prikazan na slici 60.

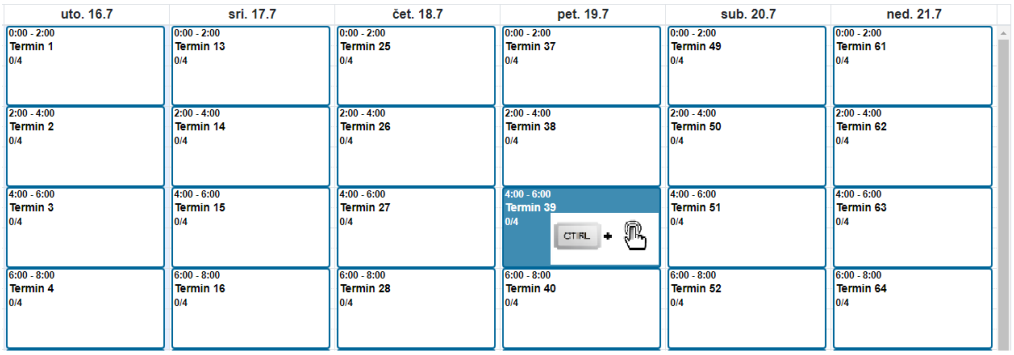


Slika 60. Kreiranje seta termina

8.3.2. Brisanje termina

8.3.2.1. Brisanje pojedinog termina

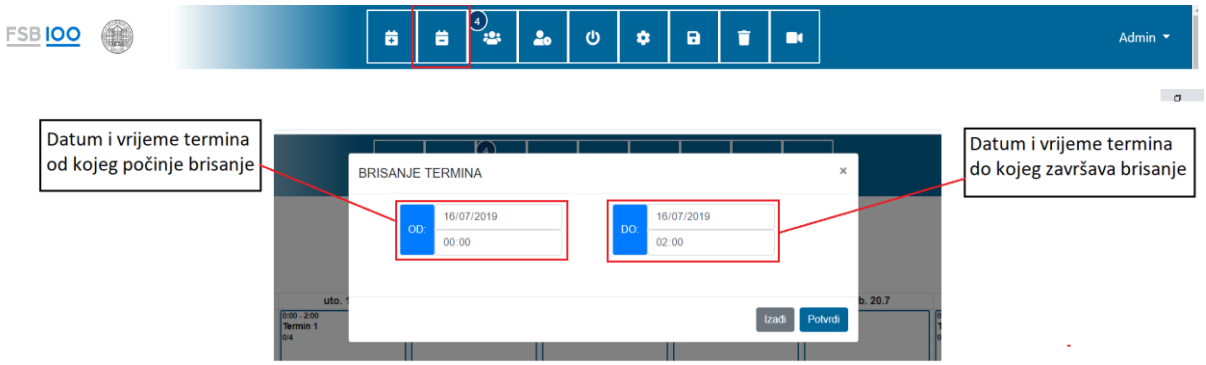
Brisanje jednog termina moguće je pritiskom kombinacije tipki ctrl+click (tipkovnica – miš) na termin koji želimo izbrisati.



Slika 61. Brisanje pojedinog termina

8.3.2.2. Brisanje seta termina

Brisanje svih termina koji se nalaze između početnog i konačnog moguće je klikom na drugu po redu ikonu u izborniku čime se otvara prozor prikazan na slici 62. Za aktivaciju opcije potrebno je definirati datume i vremena početnog i konačnog termina.



Slika 62. Brisanje seta termina

8.3.3. Spremanje promjena

Nakon završene strukture kalendara potrebno je stisnuti dugme „Spremi“ koje se nalazi u izborniku kako bi promjene bile pohranjene. Nakon pritiska na dugme pokazivač miša (engl. cursor) mjenja oblik te se nakon pohranjenih promjena vraća u prvobitni što je znak da su promjene uspješno pohranjene.

Slika 63. Spremanje promjena

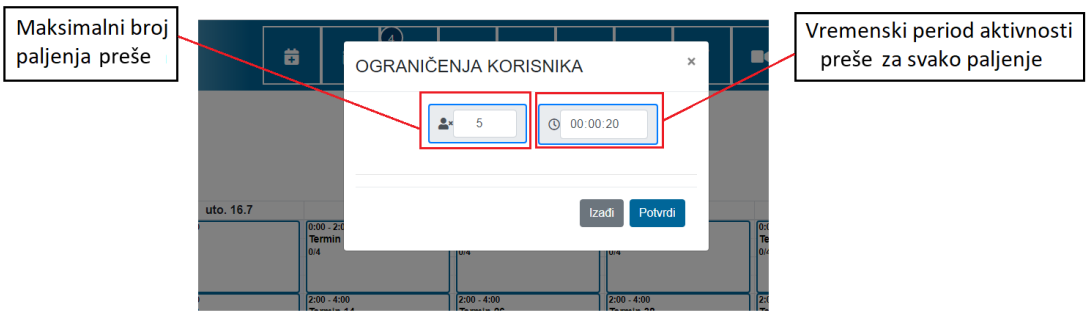
8.3.4. Broj korisnika

Definiranje broja korisnika prilikom kreiranja svakog novog termina moguće je regulirati klikom na treću po redu ikonu u izborniku gdje se upisuje željena brojčana vrijednost. Početna vrijednost broja korisnika jednaka je broju aktivnih pneumatskih preša.

Slika 64. Definiranje broja korisnika

8.3.5. Ograničenja termina

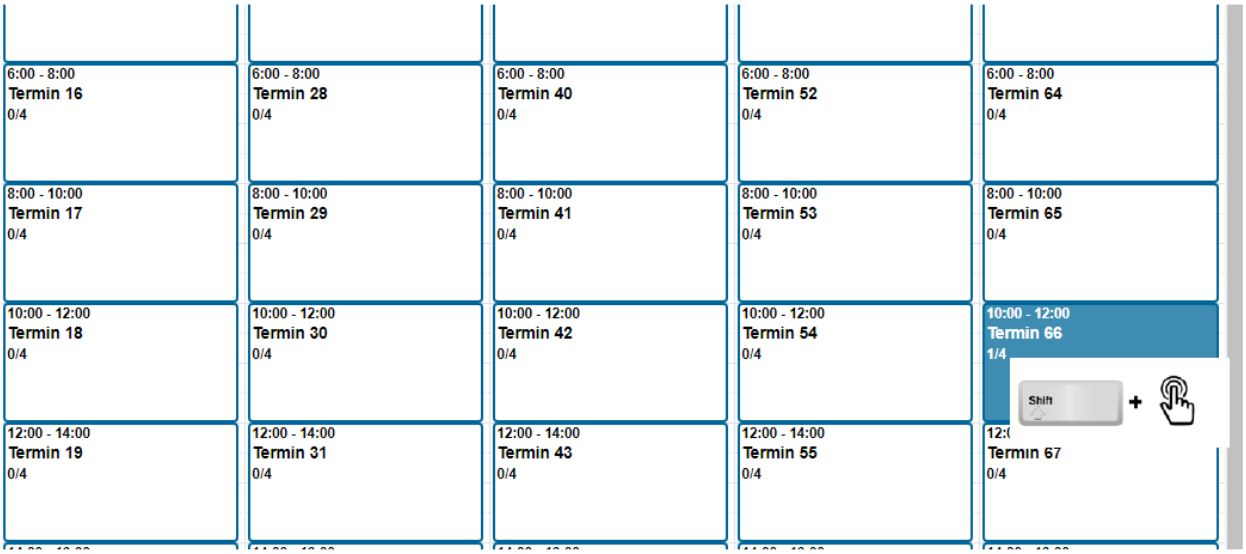
Kako bi se onemogućila dugotrajna aktivnost sustava, neophodno je bilo implementirati određena ograničenja. Pritiskom na četvrtu po redu ikonu u izborniku moguće je definirati željeni broj paljenja pneumatske preše kao i vremenski period u kojem će preša biti u aktivnom stanju. Primjer prikazan na slici 65 ukazuje da je 5 puta moguće držati prešu upaljenom, a svako aktivno stanje trajati će 20 sekundi nakon čega će se preša automatski ugasiti.



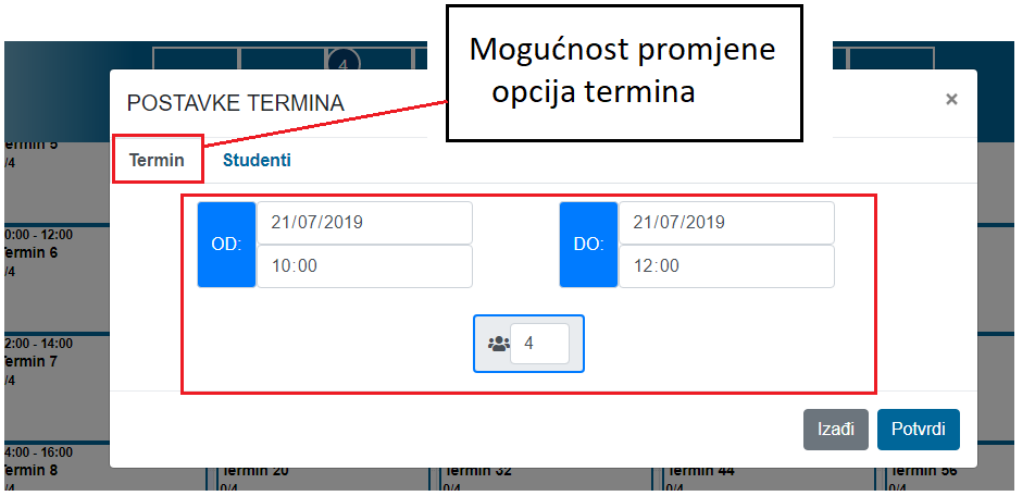
Slika 65. Ograničenja termina

8.3.6. Opcije termina

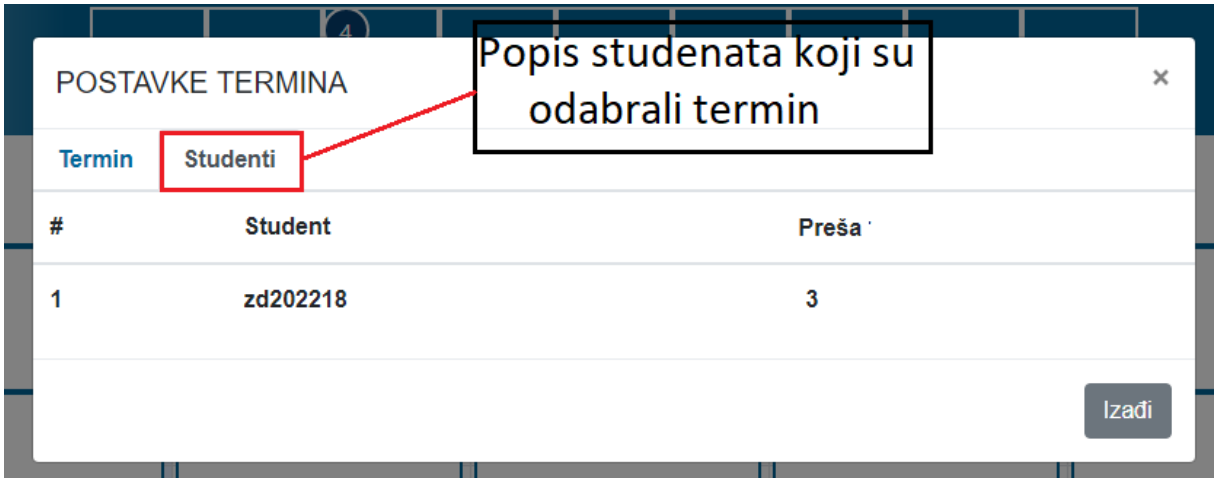
Pritiskom kombinacije tipki shift+click (tipkovnica – miš) na jedan od termina prikazuju se pojedinosti vezane za termin. Pritom, moguće je izvršiti sljedeće promjene: promjena datuma i vremena trajanja termina, maksimalan broj korisnika po terminu. Također, vidljiv je prikaz svih korisnika koji su taj termin prijavili.



Slika 66. Opcije termina



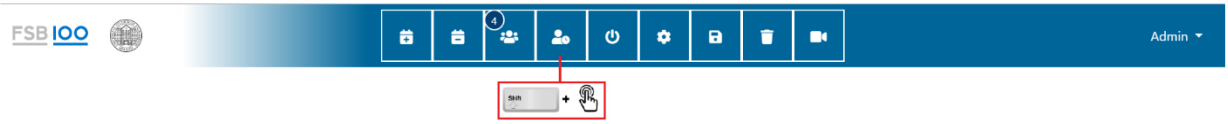
Slika 67. Definiranje opcija termina



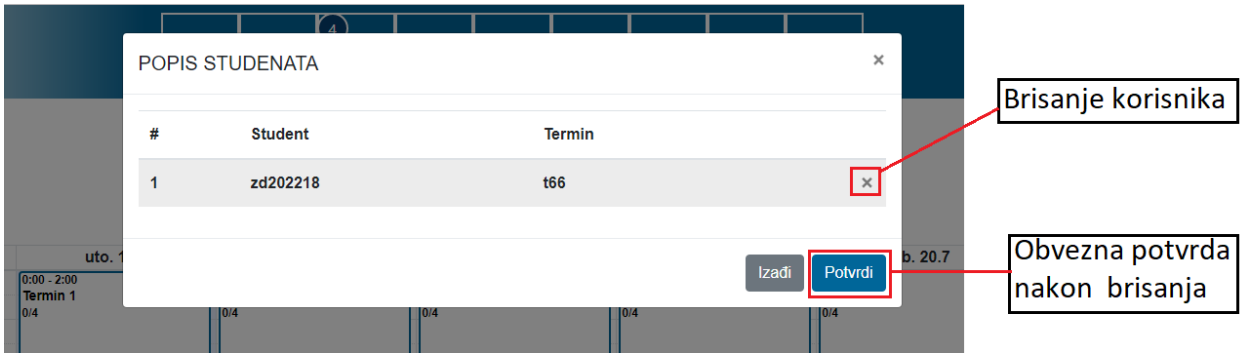
Slika 68. Popis korisnika koji su odabrali termin

8.3.7. Popis korisnika

Pregled svih korisnika istovremeno sa mogućnošću brisanja odabranog korisnikovog termina za potencijalni pronalazak novog termina moguće je pritiskom kombinacije tipki shift + click (tipkovnica – miš) na ikonu broja korisnika koja se nalazi u izborniku (treća po redu). Napomena: nakon potvrđivanja brisanja korisnika te pritiska na dugme „Potvrdi“ korisniku se automatski šalje mail da može pronaći novi termin. Moguće je brisanje i više korisnika, ali se brisanje obvezno mora potvrditi.



Slika 69. Popis korisnika



Slika 70. Brisanje korisnika

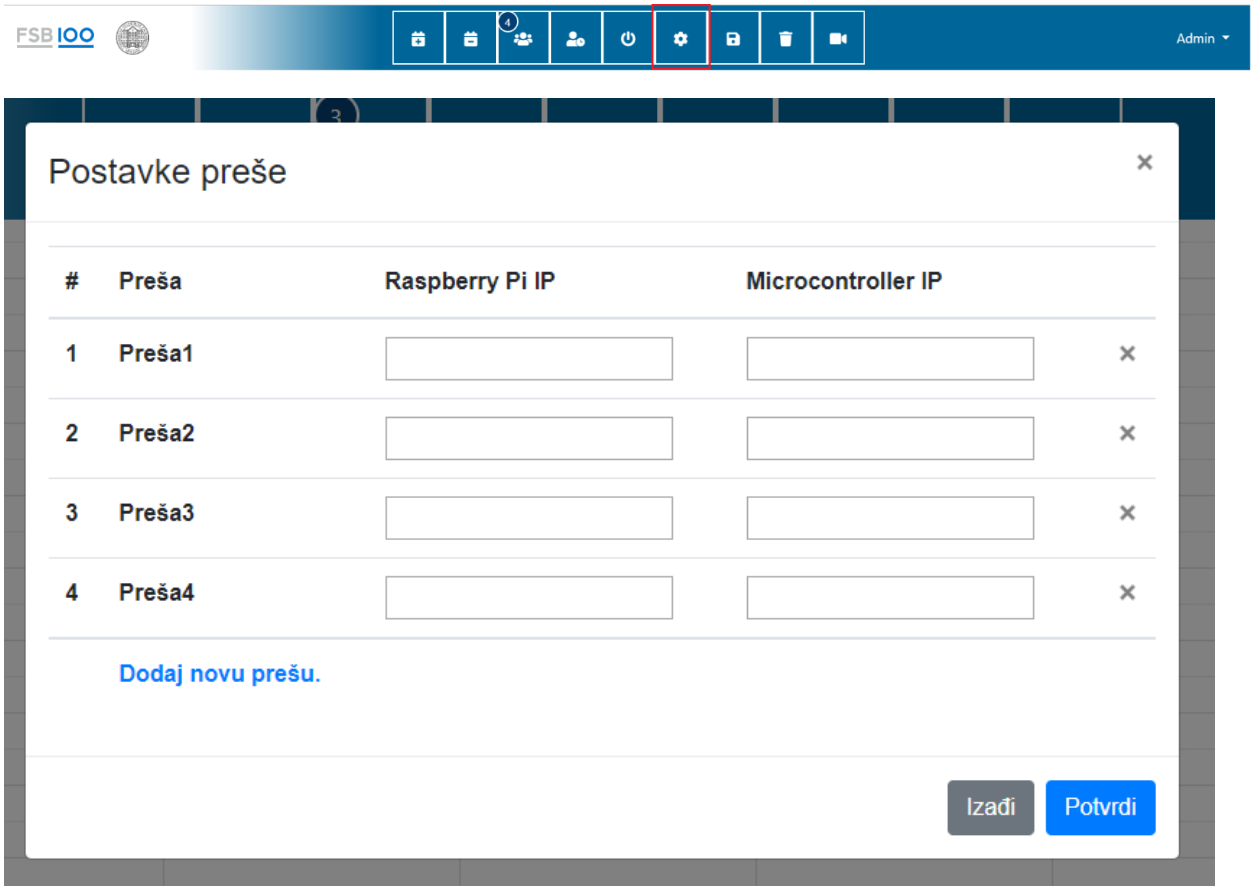
8.3.8. Status preše

Moguće je definirati koje preše (ukoliko ih je više) su trenutno aktivne klikom na ikonu u izborniku (peta po redu). Ukoliko se jedna od aktivnih preša zbog kvara naknadno ugasi, svi korisnici koji su prijavili jedan od trenutnih ili budućih termina biti će preusmjereni na neku od slobodnih preša u terminu koji su prijavili. Ukoliko ne postoji slobodna preša u tom terminu korisnik se automatski odjavljuje sa prijavljenog termina te dobiva mail da pronađe novi termin.

Slika 71. Status preše

8.3.9. Postavke preše

Za definiranje parova IP adresa (raspberry PI + mikrokontroler) za svaku prešu, kao i za dodavanje nove preše te brisanje postojeće implementirana je opcija koja se nalazi na šestom mjestu u izborniku.



Slika 72. Postavke preše

8.3.10. Brisanje baze podataka

Za sveukupni reset aplikacije potrebno je pritisnuti predzadnju ikonu u izborniku čime se brišu svi podaci iz baza podataka.

Slika 73. Brisanje baze podataka

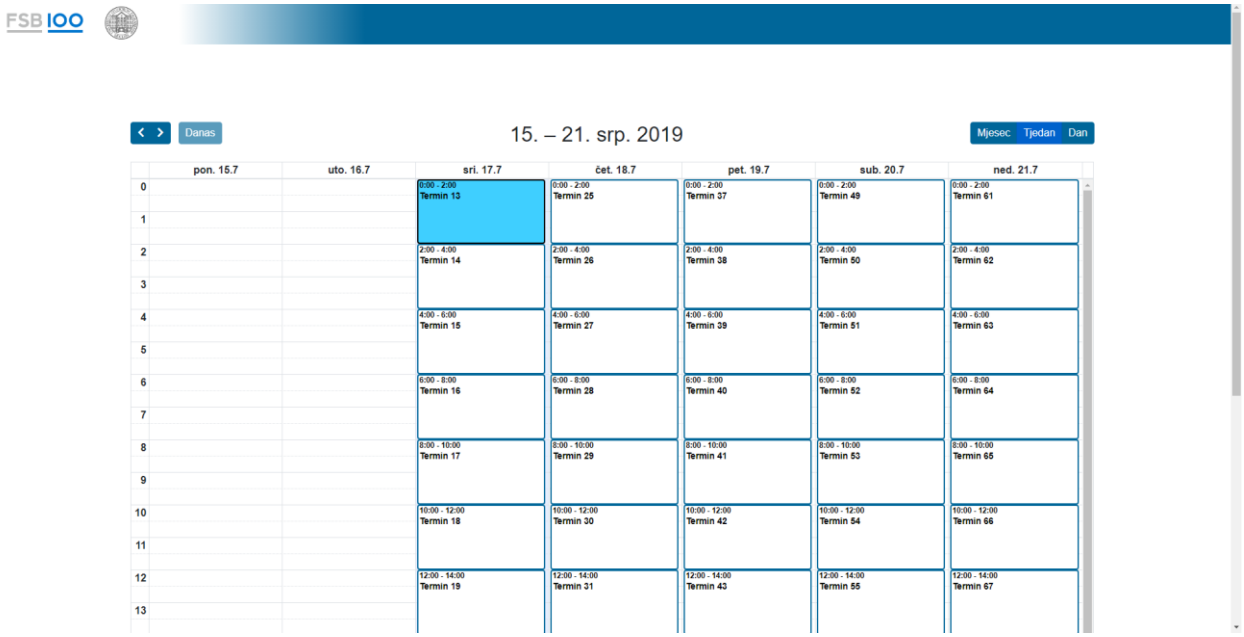
8.3.11. Video nadzor

Administratoru se nudi mogućnost nadziranja svih aktivnih preša pritiskom na posljednju ikonu u izborniku kako bi u slučaju nenadane situacije mogao reagirati.

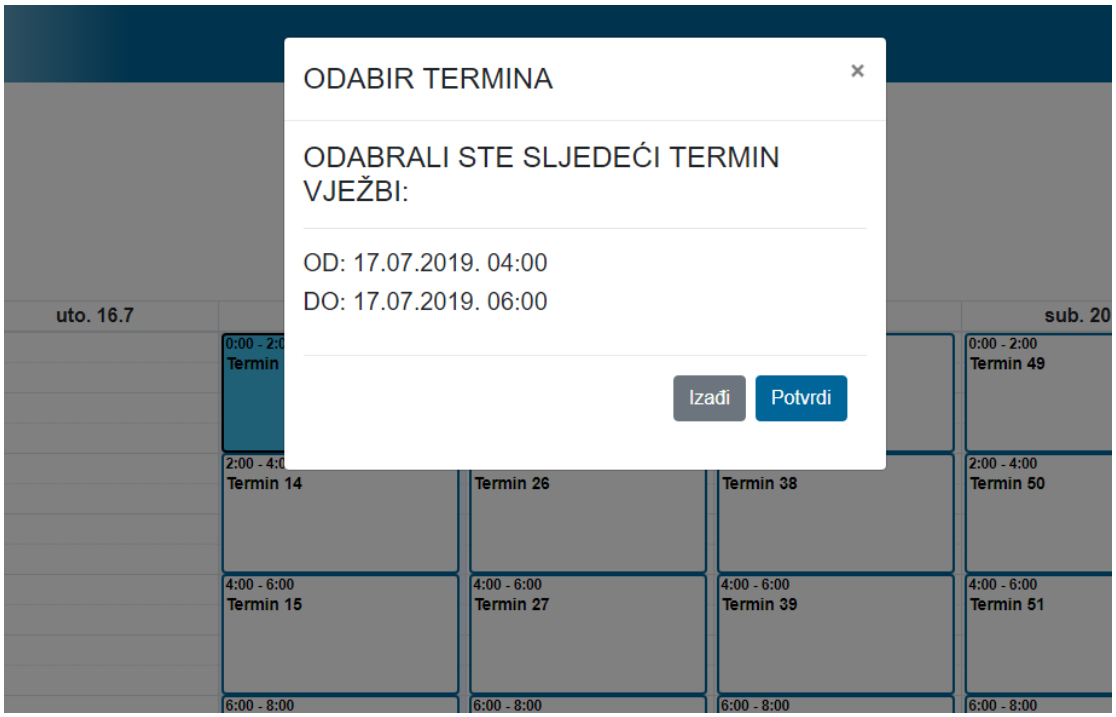
Slika 74. Video nadzor

8.4. Korisničko sučelje

Nakon prijave, korisniku se prikazuje kalendar. Prikazani kalendar sastoji se od termina koje je administrator prethodno definirao a koji nisu popunjeni ili istekli. Klikom na jedan od termina korisniku se otvara prozor upita o odabiru termina. Za potvrdu izbora termina potrebno je kliknuti dugme “Potvrdi”.

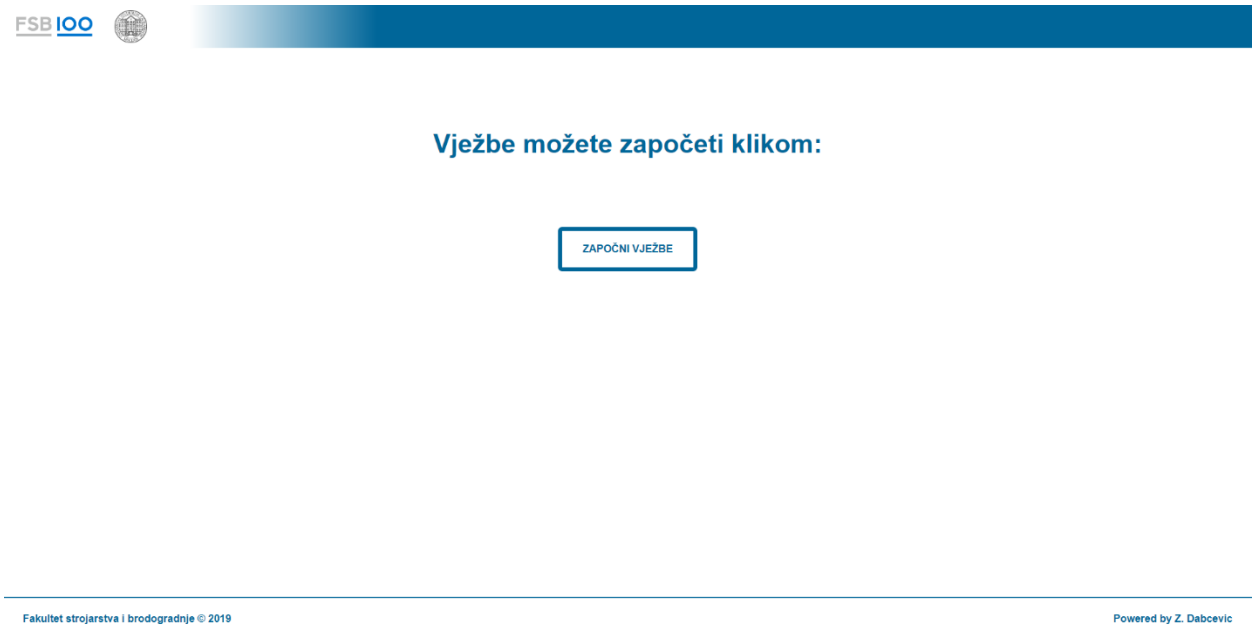


Slika 75. Korisničko sučelje

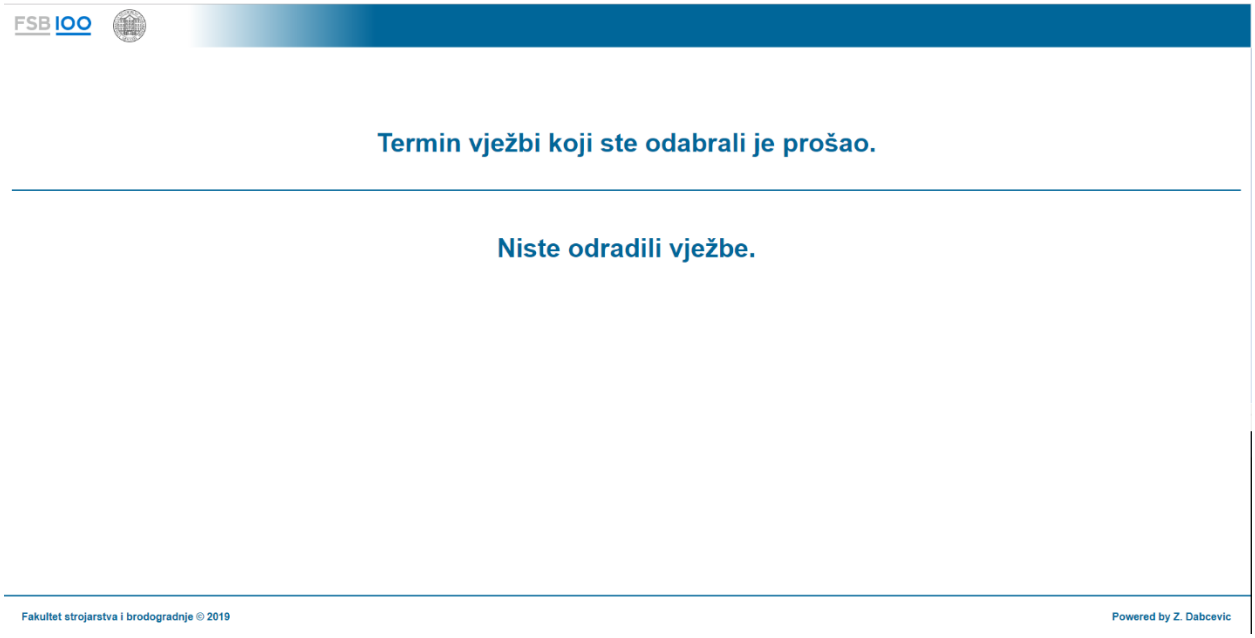


Slika 76. Odabir termina

Korisniku ima pravo pristupa upravljačkom sučelju jedino u rezerviranom terminu. Ukoliko je rezervirani termin prošao, korisniku se onemogućuje pristup upavljačkom sučelju.



Slika 77. Mogućnost pristupa upravljačkome sučelju



Slika 78. Onemogućen pristup upravljačkome sučelju

9. ZAKLJUČAK

Kao i svi ostali aspekti tehnologije, tako i internet stvari ima svoje prednosti i nedostatke. Veliki nedostatak interneta stvari svakako jest pitanje sigurnosti. Budući da su uređaji povezani putem interneta, postoji veliki rizik od „curenja“ informacija koje bi potencijalno mogle biti od velike važnosti.

S druge strane, najveća prednosti svakako je brzo i jednostavno prikupljanje informacija bez obzira na geografsku lokaciju pomoću kvalitetnije komunikacije preko mreže međusobno povezanih uređaja što komunikaciju čini transparetnijom i učinkovitijom. Internet stvari znatno olakšava komunikaciju između uređaja i čini ju bržom, što za posljedicu ima znatnu uštedu vremena i novca. Pored toga, automatiziranje zadataka koje uređaji moraju izvršiti, pomaže u poslovanju povećanja kvalitete usluga i smanjenju razine ljudske intervencije.

LITERATURA

[1] Rajčić, N.: Konstrukcija i upravljanje pneumatske preše, završni rad, FSB, 2017.

[2] Dabčević, Z.: Dijagnosticiranje i rješavanje TCP/IP problema u operacijskom sustavu Windows 7, Tehnička škola Ruđer Bošković Zagreb, 2014.

[3] Addison, W.: UNIX Network Programming Volume 1, Third Edition: The Sockets Networking API, Pearson Education, 2004.

[4] Wikipedia, Fourier transform, 08. 02. 2020.

[5] Wikipedia, PID controller, 10. 02. 2020.

[6] Practical cryptography, Mel Frequency Cepstral Coefficient (MFCC), 12. 02. 2020.

[7] Čupić, M.: Umjetne neuronske mreže, Marko Čupić, 2018.

[8] Dasaradh, S. K., An Introduction To Mathematics Behind Neural Networks, 12. 02. 2020.

[9] Sumit Saha, A Comprehensive Guide to Convolutional Neual Networks – the ELI5 way, 14. 02. 2020.

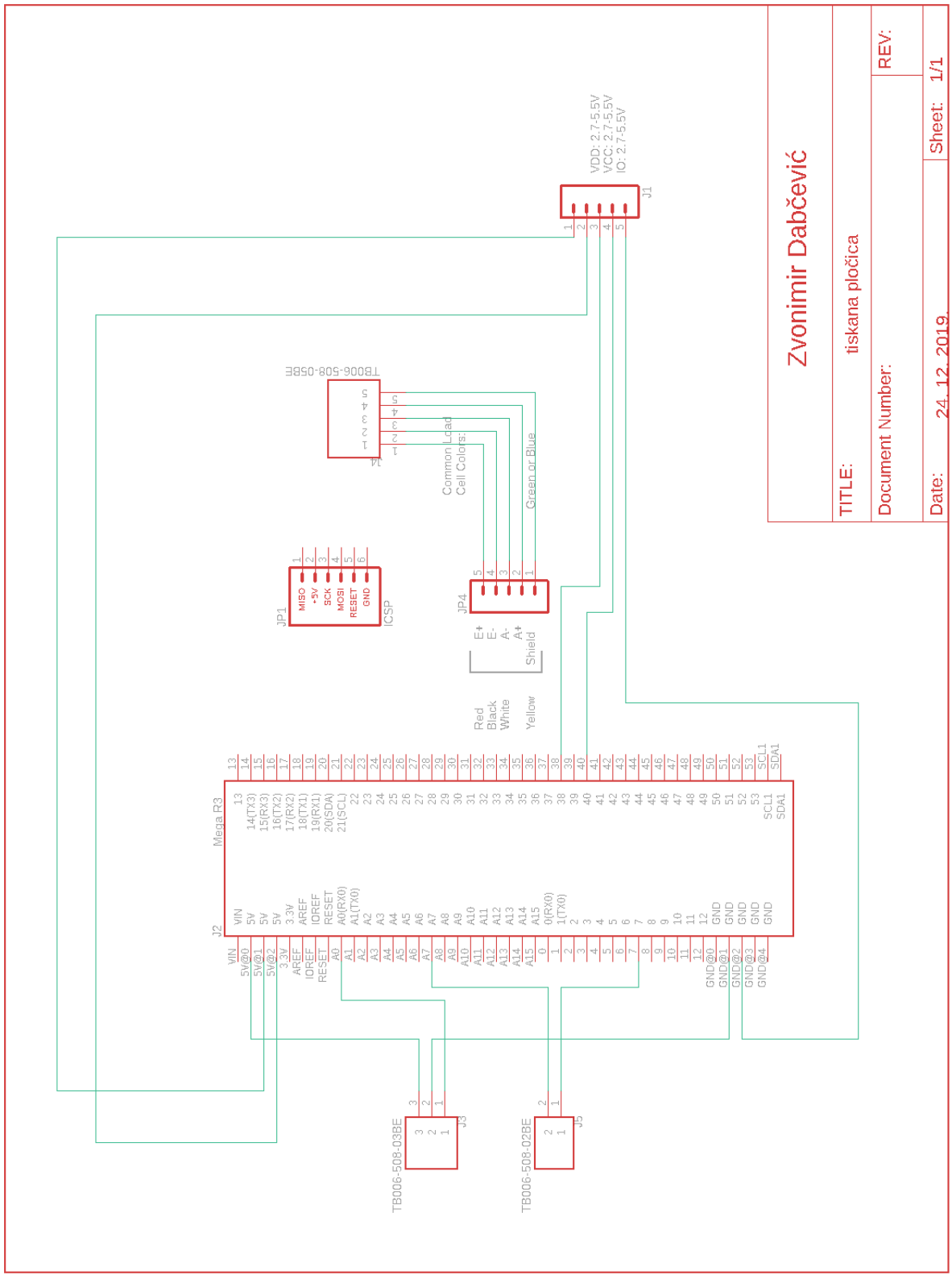
PRILOZI

I. CD-R disc

II. Shema tiskane pločice

III. Programski kôdovi

PRILOG II. Shema tiskane pločice



PRILOG III. Programski kôdovi

Arduino (C++) kôd

```
#include <Ethernet.h> //Load Ethernet Library
#include <EthernetUdp.h> //Load UDP library
#include <SPI.h> //Load the SPI library
#include<TimerOne.h>
#include "HX711.h"

double a0_read;
float out_volt, final_distance, shift;
int F;
long f_sen;
char reg_by, prev_reg_by;
double a7_read, out7_volt, pressure;

const int LOADCELL_DOUT_PIN = 38;
const int LOADCELL_SCK_PIN = 40;

HX711 scale;

double kp;
double ki;
double kd;

unsigned long currentTime, previousTime;
double elapsedTime;
double error;
double lastError;
double input, output, sensor_send;
float setPoint;
double cumError, rateError;

byte mac[] { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEE}; //Assign a mac address
IPAddress ip{192, 168, 0, 109};
unsigned int localPort = 5000; //Assign a Port to talk over
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
String datReq; //String for our data
int packetSize; //Size of Packet
int flag, flag2;
EthernetUDP Udp; //Define UDP Object

void setup() {
  Serial.begin(9600); //Turn on Serial Port
  Ethernet.begin(mac, ip); //Initialize Ethernet
  Udp.begin(localPort); //Initialize Udp
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  delay(1500); //delay
  Timer1.initialize(100000);
  Timer1.attachInterrupt(PID_loop);
}

void loop() {
  packetSize = Udp.parsePacket(); //Read the packetSize
  if (packetSize > 0) {
    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE); //Reading the data requset on
the UDP
    String datReq(packetBuffer); //Convert packetBuffer array to string datReq
    reg_by = datReq.charAt(0);
    datReq.remove(0, 1);
    setPoint = datReq.toFloat();
    if (prev_reg_by != reg_by) {
      if (reg_by == 's') {
        kp = 55;
        ki = 20;
```

```
        kd = 5;
        error = 0;
        cumError = 0;
        rateError = 0;
      }
      else if (reg_by == 'f') {
        kp = 0.1;
        ki = 0.3;
        kd = 0.01;
      }
    }
    prev_reg_by = reg_by;
    memset(packetBuffer, 0, UDP_TX_PACKET_MAX_SIZE);
  }
  if (setPoint !=0) {
      if (reg_by == 's') {
        f_sen = scale.read(); //force
        F = ((25113.6*f_sen)/(40265313.6))-138.5;
      }
      else if (reg_by == 'f') {
        a0_read = analogRead(A0); //shift
        out_volt = a0_read * (5.0 / 1023.0);
        final_distance = (-15*out_volt) + 27.75;
        shift = 15.07 - final_distance;
        shift = round(shift * 10.0)/10.0;
      }
    a7_read = analogRead(A7);
    out7_volt = a7_read * (5.0 / 1023.0);
    pressure = 0.2032*out7_volt - 0.1527;
    flag = 1;
    Udp.beginPacket(Udp.remoteIP(), 56977); //Initialize Packet send
    Udp.print(shift); //Sent string back to client
    Udp.print(",");
    Udp.print(F);
    Udp.print(",");
    Udp.print(pressure);
    Udp.endPacket(); //Packetz_trans has been sent
  }
  else {
    flag = 0;
    analogWrite(7, 0);
  }
  delay(100);
}

void PID_loop(){
  if (flag == 1) {
    if (reg_by == 's') {
      a0_read = analogRead(A0); //shift
      out_volt = a0_read * (5.0 / 1023.0);
      final_distance = (-15*out_volt) + 27.75;
      shift = 15.07 - final_distance;
      shift = round(shift * 10.0) / 10.0;
      if (shift >= 0 && shift <= 2.1) {
        output = computePID(shift);
        analogWrite(7, output);
      }
    }
    else if (reg_by == 'f') {
      f_sen = scale.read(); //force
      F = ((25113.6*f_sen)/(40265313.6))-138.5;
      if (F >= -100 && F <= 1500) {
        output = computePID(F);
        analogWrite(7, output);
      }
    }
```

```
  }
}

double computePID(float inp) {
  elapsedTime = 0.1; //time_difference
  error = setPoint - inp; //error
  cumError += error * elapsedTime; //integral
  rateError = (error - lastError)/elapsedTime; //derivacija
  double out = kp*error + ki*cumError + kd*rateError;
  if (out > 255) { out = 255;}
  else if (out < 0) { out = 0;}

  lastError = error;

  return out;
}
```

Python (neuronska mreža) kôd sa najboljom strukturu:

```
def extract_features(file_name):

    try:
        audio, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=268)
        mfccsscaled = np.mean(mfccs.T,axis=0)
        mfccsscaled = librosa.util.normalize(mfccsscaled)

    except Exception as e:
        print("Error encountered while parsing file: ", file)
        return None
    return mfccsscaled

import pandas as pd
import os
import librosa
import numpy as np

DATADIR = path
CATEGORIES = ["dolje", "gore", "pomak", "sila", "stani", "ugasi", "upali"]

features = []

def create_training_data():
    for category in CATEGORIES:
        path=os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for wav in os.listdir(path):
            try:
                file_name = os.path.join(path,wav)
                data = extract_features(file_name)
                features.append([data, class_num])
            except:
                print('except')

create_training_data()

featuresdf = pd.DataFrame(features, columns=['feature','class_label'])

from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical

X = np.array(featuresdf.feature.tolist())
y = np.array(featuresdf.class_label.tolist())
le = LabelEncoder()
yy = to_categorical(le.fit_transform(y))
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, yy, test_size=0.2,
random_state = 42, shuffle=True)

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, Conv2D, MaxPooling2D,
GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.utils import np_utils
from sklearn import metrics

num_rows = 4
num_columns = 32
num_channels = 1

x_train = x_train.reshape(x_train.shape[0], num_rows, num_columns, num_channels)
x_test = x_test.reshape(x_test.shape[0], num_rows, num_columns, num_channels)
print(x_train.shape)
print(y_train.shape)
num_labels = yy.shape[1]
filter_size = 2

# Construct model
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=2, input_shape=(num_rows, num_columns,
num_channels), activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Dense(32))
model.add(Activation("relu"))

model.add(Dense(64))
model.add(Activation("relu"))

model.add(Dense(128))
model.add(Activation("relu"))

model.add(Dense(256))
model.add(Activation("relu"))

model.add(Dense(512))
model.add(Activation("relu"))

model.add(Dense(512))
model.add(Activation("relu"))

model.add(Dense(num_labels, activation='softmax'))

model.compile(loss='categorical_crossentropy', metrics=['accuracy'],
optimizer='adam')

score = model.evaluate(x_test, y_test, verbose=1)
accuracy = 100*score[1]

from keras.callbacks import ModelCheckpoint

num_epochs = 100
num_batch_size = 20

checkpointer = ModelCheckpoint(filepath=path,
                               verbose=1, save_best_only=True)
```

```
model.fit(x_train, y_train, batch_size=num_batch_size, epochs=num_epochs,
validation_data=(x_test, y_test), callbacks=[checkpointer], verbose=1)
```

WEB Stranica – Python kôd

default.py

```
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
# This is a sample controller
# this file is released under public domain and you can use without limitations
# ---------------------------------------------------------------------------

# ---- example index page ----
import os
import librosa
import numpy as np
from tensorflow import keras
import datetime
from datetime import timedelta
from socket import *
import json

@auth.requires_login()
def index():
    if db(db.selected_time.f_user_id==auth.user.id).count():
        redirect(URL('appointment_chosen'))
    else:
        return dict()

def get_wav():
    try:
        fd=(request.vars['audio_data']).file
        act_max = 0.75
        filepath = os.path.join(request.folder, 'models',
                                'weights.best.basic_cnn.hdf5')
        model = keras.models.load_model(filepath)
        audio, sample_rate = librosa.load(fd, res_type='kaiser_fast')
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=128)
        mfccsscaled = np.mean(mfccs.T,axis=0)
        mfccsscaled = librosa.util.normalize(mfccsscaled)
        CATEGORIES = ["dolje", "gore", "pomak", "sila", "stani", "ugasi", "upali"]
        prediction = np.array(mfccsscaled)
        prediction = prediction.reshape(1, 4, 32, 1)
        prediction_res = model.predict([prediction])
        prediction_res = prediction_res[-1]
        maxElement = np.amax(prediction_res)
        max_index, = np.where(prediction_res == maxElement)
        maxElement = maxElement.item()
        return_string = CATEGORIES[(max_index)[-1]]
        if (float(maxElement) > act_max):
            return json.dumps({"value":return_string})
        else:
            return json.dumps({"value":"empty"})
    except:
        return json.dumps({"value":"empty"})


def final_end():
    for row in db(db.selected_time.f_user_id==auth.user.id).select():
        selected_id = row.appointment_id
    if(db(db.dates.event_appointment_id==selected_id).count()):
        for row in db(db.dates.event_appointment_id==selected_id).select():
            start_date = row.event_start_date+datetime.timedelta(minutes=2)
```

```
            final_date = row.event_final_date
    else:
        db(db.selected_time.f_user_id==auth.user.id).delete()
        redirect(URL('index'))
    return dict(start_date=start_date, final_date=final_date)

def user_database():
    if db(db.selected_time.f_user_id==auth.user.id).count():
        db_id =
db(db.selected_time.f_user_id==auth.user.id).select().first().press_number
    else:
        db_id = 1
    return "press"+str(db_id)

@auth.requires(True, requires_login=session.req_boolean)
def results():
    if db(db.selected_time.f_user_id==auth.user.id).count():
        pass
    else:
        redirect(URL('index'))
    return dict()

def get_results():
    import csv
    import sys
    import os
    if db(db.selected_time.f_user_id==auth.user.id).count():
        if
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.db_export).fir
st().db_export:
            file_to_read =
(db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.db_export).fi
rst().db_export)
            f = open(os.path.join(request.folder, 'uploads', file_to_read), 'r')
            reader = csv.reader(f)
            lista_struja = []
            lista_napon = []
            lista_brzina = []
            row_num=0
            for row in reader:
                if(row_num != 0):
                    splitted = (row[0].split(';'))
                    lista_napon.append(splitted[1])
                    lista_struja.append(splitted[2])
                    lista_brzina.append(splitted[3])
                else:
                    row_num+=1
            f.close
            return json.dumps({'napon':lista_napon, 'struja':lista_struja,
'brzina':lista_brzina})
        else:
            pass
    return dict()

def initial_values():
    db_id = user_database()
    rows=db((db[db_id]).id>0).select()
    try:
        response.cookies['counter'] = rows.last().id
    except AttributeError:
        response.cookies['counter'] = 0
    return json.dumps([[r.input1, r.input2, r.input3] for r in rows])

def socket_communication():
    data_send = (request.vars['socket_value'])
    if (data_send == None):
```

```
        data_send = ""
    transmitter(data_send)
    return json.dumps({"status":200})

def transmitter(message):
    try:
        client_socket
    except:
        client_socket = socket(AF_INET, SOCK_DGRAM)
        client_socket.bind((secret_ip1, secret_port1))
        address = (secret_ip2, secret_port2) #Define who you are talking to
        client_socket.settimeout(1) #only wait 1 second for a response
    try:
        client_socket.sendto(bytes(message, 'utf-8'), address) #send command to
arduino
        data, addr = client_socket.recvfrom(1024)
        data = data.decode("utf-8")
        data_strip = [x.strip() for x in data.split(',')]
        db.receive.insert(shift=data_strip[0], F_force=data_strip[1],
pressure=data_strip[2])
    except:
        pass
    return

def getdata():
    if (auth.is_logged_in()):
        db_id = user_database()
        rows = db((db[db_id]).id>int(request.cookies['counter'].value)).select()
        if not rows:
            i=json.dumps([])
        else:
            response.cookies['counter'] = rows.last().id
            i= json.dumps([[r.input1, r.input2, r.input3] for r in rows])
        return i
    else:
        rows = db((db["receive"]).id>int(session.counter)).select()
        if not rows:
            i=json.dumps([2, 3, 1])
        else:
            session.counter = rows.last().id
            i = json.dumps([[r.shift, r.F_force, r.pressure] for r in rows])
        return i

@auth.requires_login()
def appointment_chosen():
    if (db(db.selected_time.f_user_id==auth.user.id).count()):
        mydict = final_end()
        start_date = (mydict['start_date'])
        final_date = (mydict['final_date'])
        db_id = user_database()
        if
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.db_export).fir
st().db_export:
            file_to_upload =
(db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.db_export).fi
rst().db_export)
        else:
            file_to_upload = 0
        if
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.generated_app)
.first().generated_app:
            check_if_started =
(db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.generated_app
).first().generated_app)
        else:
            check_if_started = 0
```

```
    else:
        redirect(URL('index'))
        start_date = 0
        final_date = 0
        file_to_upload = 0
        check_if_started = 0
        db_empty = 0
    current_datetime = datetime.datetime.now()
    checkout_datetime = start_date - current_datetime
    checkout_time = datetime.timedelta(0, 28800)
    return dict(start_date = start_date, final_date=final_date,
current_datetime=current_datetime, checkout_datetime=checkout_datetime,
checkout_time=checkout_time, file_to_upload=file_to_upload,
check_if_started=check_if_started)

def date_converter(date_input):
    date_output = datetime.datetime.strptime(date_input, '%Y-%m-%d %H:%M')
    return date_output

def save():
    date_vars=(request.vars)
    db.dates.truncate()
    if date_vars["event_appointment_id[]"]:
        if isinstance(date_vars["event_appointment_id[]"], list):
            for i in range (0,len(date_vars["event_appointment_id[]"])):

db.dates.insert(event_start_date=date_converter(date_vars["event_start_date[]"][i])
,

event_final_date=date_converter(date_vars["event_final_date[]"][i]),

event_appointment_id=date_vars["event_appointment_id[]"][i])
                if
(db(db.id_counter.appointment_id==date_vars["event_appointment_id[]"][i]).count()==
False):

db.id_counter.insert(appointment_id=date_vars["event_appointment_id[]"][i],
                                        max_count=date_vars["max_users2[]"][i])
                else:
                    current_max =
db(db.id_counter.appointment_id==date_vars["event_appointment_id[]"][i]).select(db.
id_counter.max_count).first().max_count
                    if (date_vars["max_users2[]"][i] != current_max):

db(db.id_counter.appointment_id==date_vars["event_appointment_id[]"][i]).update(max
_count=date_vars["max_users2[]"][i])
        elif isinstance(date_vars["event_appointment_id[]"], str):

db.dates.insert(event_start_date=date_converter(date_vars["event_start_date[]"]),

event_final_date=date_converter(date_vars["event_final_date[]"]),

event_appointment_id=date_vars["event_appointment_id[]"])
            if
(db(db.id_counter.appointment_id==date_vars["event_appointment_id[]"]).count()==Fal
se):

db.id_counter.insert(appointment_id=date_vars["event_appointment_id[]"],
                                        max_count=date_vars["max_users2[]"])
            else:
                    current_max =
db(db.id_counter.appointment_id==date_vars["event_appointment_id[]"]).select(db.id_
counter.max_count).first().max_count
                    if (date_vars["max_users2[]"] != current_max):
```

```
db(db.id_counter.appointment_id==date_vars["event_appointment_id[]"]).update(max_co
unt=date_vars["max_users2[]"])
        else:
            pass
    else:
        pass
    if date_vars["deleted_events[]"]:
        rows=db(db.selected_time.id>0).select()
        if isinstance(date_vars["deleted_events[]"], list):
            for j in range (0,len(date_vars["deleted_events[]"])):
                for row in rows:
                    if (row.appointment_id == date_vars["deleted_events[]"][j]):

db(db.selected_time.appointment_id==date_vars["deleted_events[]"][j]).delete()

db(db.id_counter.appointment_id==date_vars["deleted_events[]"][j]).delete()
        elif isinstance(date_vars["deleted_events[]"], str):
            for row in rows:
                    if (row.appointment_id == date_vars["deleted_events[]"]):

db(db.selected_time.appointment_id==date_vars["deleted_events[]"]).delete()

db(db.id_counter.appointment_id==date_vars["deleted_events[]"]).delete()
        else:
            pass
    else:
        pass
    return json.dumps({"data":"Termini su generirani!", "status":200})

def press_update():
    press_update_vars=(request.vars)
    current_datetime = datetime.datetime.now()
    if "check_others" in press_update_vars:
        del press_update_vars["check_others"]
        new_max_value = len(press_update_vars)
        rows = db(db.id_counter.id>0).select()
        for row in rows:
            date_to_compare =
db(db.dates.event_appointment_id==row.appointment_id).select(db.dates.event_final_d
ate).first().event_final_date
            if (current_datetime < date_to_compare):
                max_count =
db(db.id_counter.appointment_id==row.appointment_id).select(db.id_counter.max_count
).first().max_count
                if (max_count != new_max_value):
                    db(db.id_counter.appointment_id ==
row.appointment_id).update(max_count=new_max_value)
    if not press_update_vars:
        rows=db(db.press.id>0).select()
        for row in rows:
            db(db.press.name == row.name).update(dc_active=False)
        rows2 = db(db.selected_time.id>0).select()
        for row in rows2:
            date_to_compare =
db(db.dates.event_appointment_id==row.appointment_id).select(db.dates.event_final_d
ate).first().event_final_date
            if (current_datetime < date_to_compare):
                rows2=db(db.selected_time.f_user_id==row.f_user_id).delete()
                db(db.id_counter.appointment_id ==
row.appointment_id).update(id_count=db.id_counter.id_count - 1)
                print("Javi korisniku da prona'e novi termin")

    else:
        if (db(db.press).count()):
            rows=db(db.press.id>0).select()
```

```
            for key, value in press_update_vars.items():
                if (value == "on"):
                    value = True
                for row in rows:
                    if ((row.name == key) and (row.dc_active != value)):
                        db(db.press.name == row.name).update(dc_active=True)
                    elif ((row.name not in press_update_vars) and (row.dc_active !=
False)):
                        db(db.press.name == row.name).update(dc_active=False)
        if (db(db.selected_time).count()):
            rows = db(db.selected_time.id>0).select()
            for row in rows:
                date_to_compare =
db(db.dates.event_appointment_id==row.appointment_id).select(db.dates.event_final_d
ate).first().event_final_date
                if (current_datetime < date_to_compare):
                    row.press_number = "Preša"+str(row.press_number)
                    if (row.press_number not in press_update_vars):
                        key_value_counter = 0
                        list_to_compare = []
                        for row2 in
db(db.selected_time.appointment_id==row.appointment_id).select():
                            list_to_compare.append(int(row2.press_number))
                        for key, value in press_update_vars.items():
                            key_value_counter+=1
                            press_num_int = int(''.join(filter(str.isdigit, key)))
                            if (press_num_int not in list_to_compare):

db(db.selected_time.f_user_id==row.f_user_id).update(press_number=press_num_int)
                                break
                            elif (key_value_counter >= len(press_update_vars)):

db(db.selected_time.f_user_id==row.f_user_id).delete()
                                db(db.id_counter.appointment_id ==
row.appointment_id).update(id_count=db.id_counter.id_count - 1)
                                print("Javi korisniku da prona'e novi termin")
    return json.dumps({"Error":"Error!", "status":200})

def restrictions_update():
    db.restrictions.truncate()
    restrictions = (request.vars)
    db.restrictions.insert(press_start = restrictions["press_start_num"],
                           press_time = restrictions["press_dur"])
    return json.dumps({"Error":"Error!", "status":200})

def press_conf_update():
    press_conf_update_vars=(request.vars)
    db.press.truncate()
    if press_conf_update_vars["name[]"]:
        if isinstance(press_conf_update_vars["name[]"], list):
            for i in range (0,len(press_conf_update_vars["name[]"])):
                if (press_conf_update_vars["dc_active[]"][i] == "true"):
                    press_conf_update_vars["dc_active[]"][i] = True
                else:
                    press_conf_update_vars["dc_active[]"][i] = False
                db.press.insert(name=press_conf_update_vars["name[]"][i],
                                mc_ip=press_conf_update_vars["mc_ip[]"][i],
                                rpi_ip=press_conf_update_vars["rpi_ip[]"][i],

dc_active=press_conf_update_vars["dc_active[]"][i])
        elif isinstance(press_conf_update_vars["name[]"], str):
            if (press_conf_update_vars["dc_active[]"] == "true"):
                press_conf_update_vars["dc_active[]"] = True
            else:
                press_conf_update_vars["dc_active[]"] = False
            db.press.insert(name=press_conf_update_vars["name[]"],
```

```
                                    mc_ip=press_conf_update_vars["mc_ip[]"],
                                    rpi_ip=press_conf_update_vars["rpi_ip[]"],
                                    dc_active=press_conf_update_vars["dc_active[]"])
        else:
            pass
    else:
        pass
    return json.dumps({"status":200})

def delete_students():
    delete_students=(request.vars)
    if delete_students["deleted_students[]"]:
        if isinstance(delete_students["deleted_students[]"], list):
            for i in range (0,len(delete_students["deleted_students[]"])):
                stud_del_id =
str(db(db.auth_user.username==delete_students["deleted_students[]"][i]).select(db.a
uth_user.id).as_list()[0]["id"])
                app_id =
db(db.selected_time.f_user_id==stud_del_id).select(db.selected_time.appointment_id)
.first().appointment_id
                db(db.selected_time.f_user_id==stud_del_id).delete()
                db(db.id_counter.appointment_id ==
app_id).update(id_count=db.id_counter.id_count - 1)
        elif isinstance(delete_students["deleted_students[]"], str):
            stud_del_id =
str(db(db.auth_user.username==delete_students["deleted_students[]"]).select(db.auth
_user.id).as_list()[0]["id"])
            app_id =
db(db.selected_time.f_user_id==stud_del_id).select(db.selected_time.appointment_id)
.first().appointment_id
            db(db.selected_time.f_user_id==stud_del_id).delete()
            db(db.id_counter.appointment_id ==
app_id).update(id_count=db.id_counter.id_count - 1)
            #mail.send('zd202218@stud.fsb.hr',
            #    'Vježbe iz elektrotehnike',
            #    '<html>Poštovani kolega, <br><br>možete pronaći novi termin
vježbi. <br><br> LP</html>')
    return json.dumps({"status":200})

def save_date():
    appointment_id = (request.vars.appointment_id)
    current_datetime = datetime.datetime.now()
    date_to_compare =
db(db.dates.event_appointment_id==appointment_id).select(db.dates.event_final_date)
.first().event_final_date
    press_number_list = []
    if db(db.press).count():
        rows = db(db.press.id>0).select()
        for row in rows:
            if (row.dc_active == False):
                press_number_list.append(int(''.join(filter(str.isdigit,
row.name))))
    else:
        return json.dumps({"data":"Nema slobodnih preša u ovom terminu.",
"status":300})
    if db(db.selected_time).count():
        for row in db(db.selected_time.appointment_id == appointment_id).select():
            press_number_list.append(int(row.press_number))
        new_press_number = db(db.selected_time.id
>0).select(orderby=~db.selected_time.id, limitby=(0,1)).first().press_number
    else:
       new_press_number = 1
    rows = db(db.press.id>0).select(orderby='<random>')
    for row in rows:
        row.name = int(''.join(filter(str.isdigit, row.name)))
        if (row.name not in press_number_list):
```

```
            new_press_number = row.name
            break
        elif (len(rows)!=len(press_number_list)):
            continue
        else:
            return json.dumps({"data":"Nema slobodnih preša u ovom terminu.",
"status":300})

    if (db(db.selected_time.f_user_id==auth.user.id).count()==False and
auth.user.id!= None):
        if db(db.id_counter.appointment_id == appointment_id).count():
            for row in db(db.id_counter.appointment_id == appointment_id).select():
                value_to_compare = row.id_count
                max_value = row.max_count
            if ((value_to_compare < max_value)  & (current_datetime <
date_to_compare)):
                db.selected_time.insert(appointment_id=appointment_id,
press_number=new_press_number)
                db(db.id_counter.appointment_id ==
appointment_id).update(id_count=db.id_counter.id_count + 1)
                return json.dumps({"data":"Vaš termin je zabilježen",
"status":200})
            elif (value_to_compare >= max_value):
                return json.dumps({"data":"Termin je popunjen! Molimo odaberite
drugi termin.", "status":300})
            else:
                return json.dumps({"data":"Vrijeme za prijavu ovog termina je
isteklo!", "status":300})
        else:
            if (current_datetime < date_to_compare):
                db.selected_time.insert(appointment_id=appointment_id,
press_number=new_press_number)
                db.id_counter.insert(appointment_id=appointment_id, id_count=1)
                return json.dumps({"data":"Vaš termin je zabilježen",
"status":200})
            else:
                return json.dumps({"data":"Vrijeme za prijavu ovog termina je
isteklo!", "status":300})
    else:
        return json.dumps({"data":"Već ste prijavili termin", "status":300})


def check_if_6():
    list_over_6 = []
    for row in db(db.id_counter.id_count >= db.id_counter.max_count).select():
        list_over_6.append(row.appointment_id)
    return response.json(list_over_6)

def students_org():
    if (db(db.selected_time).count()):
        rows=db(db.selected_time.id>0).select()
        for row in rows:
            row.f_user_id =
str(db(db.auth_user.id==row.f_user_id).select(db.auth_user.first_name).as_list()[0]
["first_name"])
        return json.dumps([[val.f_user_id, val.appointment_id, val.press_number]
for val in rows])
    else:
        return []


def press_initial():
    if (db(db.press).count()):
        press_rows=db(db.press.id>0).select()
        return json.dumps([[val2.name, val2.dc_active, val2.rpi_ip, val2.mc_ip] for
val2 in press_rows])
```

```
    else:
        return []

def reset():
    db.auth_user.truncate()
    db(db.selected_time.db_export).delete()
    db.selected_time.truncate()
    db.id_counter.truncate()
    db.dates.truncate()
    db.scheduler_task.truncate()
    db.scheduler_run.truncate()
    db.restrictions.truncate()
    if db(db.press).count():
        rows = db(db.press.id>0).select()
        for row in rows:
            name_db = "press"+''.join(filter(str.isdigit, row.name))
            db[name_db].truncate()
    return json.dumps({})

def values():
    rows=db(db.dates.id>0).select()
    if (db(db.dates).count()):
        return json.dumps([[val.event_start_date, val.event_final_date,
val.event_appointment_id] for val in rows], default = myconverter)
    else:
        return []


def values2():
    if (db(db.id_counter).count()):
        rows=db(db.id_counter.id>0).select()
        return json.dumps([[val.appointment_id, val.id_count, val.max_count] for
val in rows])
    else:
        return []

def myconverter(o):
    if isinstance(o, datetime.datetime):
        return o.__str__()

def get_restrictions():
    if (db(db.restrictions).count()):
        rows=db(db.restrictions.id>0).select()
        return json.dumps([[val.press_start, str(val.press_time)] for val in rows])
    else:
        return []

def user_starts():
    if db(db.selected_time.f_user_id==auth.user.id):
        press_start_num =
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.press_start_nu
m).first().press_start_num
        return json.dumps([press_start_num])
    else:
        return dict()

def add_one_start():
    if db(db.selected_time.f_user_id==auth.user.id):

db(db.selected_time.f_user_id==auth.user.id).update(press_start_num=db.selected_tim
e.press_start_num + 1)
        press_start_num =
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.press_start_nu
m).first().press_start_num
        return json.dumps([press_start_num])
    elif auth.user.id == None:
```

```
        redirect(URL('index'))

def delete_appointment_for_user():
    for row in db(db.selected_time.f_user_id==auth.user.id).select():
        appointment_id = row.appointment_id
    db(db.id_counter.appointment_id ==
appointment_id).update(id_count=db.id_counter.id_count - 1)
    db(db.selected_time.f_user_id==auth.user.id).delete()
    return json.dumps({"data":"Uspješno ste odjavili termin.", "status":200})

def check_to_start():
    if db(db.selected_time.f_user_id==auth.user.id).count():
        mydict = final_end()
        start_date = (mydict['start_date'])
        final_date = (mydict['final_date'])
    else:
        redirect(URL('index'))
        start_date = 0
        final_date = 0
    current_datetime = datetime.datetime.now()
    if (start_date < current_datetime) & (current_datetime < final_date):
        return json.dumps({"status":200})
    else:
        return json.dumps({"status":300})

def check_for_data():
    if db(db.selected_time.f_user_id==auth.user.id).count():
        if
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.db_export).fir
st().db_export:
            return json.dumps({"check":1})
        else:
            return json.dumps({})
    else:
        return json.dumps({})

def start_exercise():
    if (auth.is_logged_in()):
        if db(db.selected_time.f_user_id==auth.user.id).count():
            mydict = final_end()
            start_date = (mydict['start_date'])
            final_date = (mydict['final_date'])
            press_id =
db(db.selected_time.f_user_id==auth.user.id).select().first().press_number
            rpi_ip = db(db.press.id==press_id).select().first().rpi_ip
            press_start_num =
db(db.selected_time.f_user_id==auth.user.id).select(db.selected_time.press_start_nu
m).first().press_start_num
            press_max =
db(db.restrictions).select(db.restrictions.press_start).first().press_start
            for row in db(db.selected_time.f_user_id==auth.user.id).select():
                check_if_queued = row.generated_app
                user_id = row.id
            current_datetime = datetime.datetime.now()
            if (start_date < current_datetime) & (final_date > current_datetime):
                diff = (final_date - current_datetime).total_seconds() * 1000
                if (check_if_queued != 1):
                    db_id = user_database()
                    scheduler.queue_task('scheduler_appointment',
pvars=dict(user_id=user_id, db_id=db_id), prevent_drift = True, start_time =
final_date)

db(db.selected_time.f_user_id==auth.user.id).update(generated_app=1)
                else:
                    pass
        else:
```

```
            press_start_num = 0
            press_max = 0
            diff = 0
            rpi_ip = 0
            redirect(URL('index'))
    else:
        session.counter = 0
        db.receive.truncate()
        if (db(db.press).count()):
            rpi_ip = db(db.press).select().first().rpi_ip
        else:
            rpi_ip = 0
        diff = 0
        press_max=0
        press_start_num = 0
    return dict(diff=diff, rpi_ip=rpi_ip, press_start_num=press_start_num,
press_max=press_max)

def send_mc_ip():
    if (db(db.selected_time.f_user_id==auth.user.id).count()):
        press_id =
db(db.selected_time.f_user_id==auth.user.id).select().first().press_number
        mc_ip2 = db(db.dc_press.id==press_id).select().first().mc_ip
    else:
        mc_ip2 = 0
    return json.dumps(mc_ip2)

# ---- API (example) -----
@auth.requires_login()
def api_get_user_email():
    if not request.env.request_method == 'GET': raise HTTP(403)
    return response.json({'status':'success', 'email':auth.user.email})

# ---- Smart Grid (example) -----
@auth.requires_membership('admin') # can only be accessed by members of admin
groupd
def grid():
    response.view = 'generic.html' # use a generic view
    tablename = request.args(0)
    if not tablename in db.tables: raise HTTP(403)
    grid = SQLFORM.smartgrid(db[tablename], args=[tablename], deletable=False,
editable=False)
    return dict(grid=grid)

# ---- Embedded wiki (example) ----
def wiki():
    auth.wikimenu() # add the wiki to the menu
    return auth.wiki()

# ---- Action for login/register/etc (required for auth) -----
def user():
    """
    exposes:
    http://..../[app]/default/user/login
    http://..../[app]/default/user/logout
    http://..../[app]/default/user/register
    http://..../[app]/default/user/profile
    http://..../[app]/default/user/retrieve_password
    http://..../[app]/default/user/change_password
    http://..../[app]/default/user/bulk_register
    use @auth.requires_login()
        @auth.requires_membership('group name')
        @auth.requires_permission('read','table name',record_id)
    to decorate functions that need access control
    also notice there is http://..../[app]/appadmin/manage/auth to allow
administrator to manage users
```

```
    """
    return dict(form=auth())

# ---- action to server uploaded static content (required) ---
@cache.action()
def download():
    """
    allows downloading of uploaded files
    http://..../[app]/default/download/[filename]
    """
    return response.download(request, db)
```

db.py

```
if not db(db.auth_group).count():
    db.auth_group.insert(role='Admin',description='Admin')
    db.auth_group.insert(role='User',description='User')

if not db(db.auth_user).count():
    db.auth_user.insert(first_name = 'Admin', last_name = 'Admin',email =
'eltevjezbe', username = 'Admin',
                        password = db.auth_user.password.validate('ppress2019')[0])
    for i in range(45):
        db.auth_user.insert(first_name = 'User'+str(i), last_name =
'User'+str(i),email = 'user'+str(i)+'@a.com', username = 'User'+str(i),
                    password = db.auth_user.password.validate('password')[0])

if not db(db.auth_permission).count():
    db.auth_permission.insert(group_id='1', name='priority1')
    db.auth_permission.insert(group_id='2', name='priority2')

auth.add_membership('1', '1')
auth.add_membership('2', '2')

auth.settings.create_user_groups = None
auth.settings.everybody_group_id = 2

db.define_table('dates',
                Field('event_start_date', 'datetime'),
                Field('event_final_date', 'datetime'),
                Field('event_appointment_id', 'text'))


db.define_table('selected_time',
                Field('f_user_id', type='reference auth_user',
default=auth.user_id, # Will be None if no logged in user.
                readable=False, writable=False),
                Field('appointment_id', 'text'),
                Field('press_number', 'integer'),
                Field('generated_app', 'integer', default=0),
                Field('press_start_num', 'integer', default=0),
                Field('db_export', 'upload', autodelete=True))

db.define_table('restrictions',
                Field('press_start', 'integer'),
                Field('press_time', 'time'))

db.define_table('id_counter',
                Field('appointment_id', 'text'),
                Field('id_count', 'integer',default = 0),
                Field('max_count', 'integer'))

db.define_table('press',
                Field('name', 'text'),
                Field('mc_ip', 'text'),
```

```
                Field('rpi_ip', 'text'),
                Field('dc_active', 'boolean'))

if not db(db.restrictions).count():
    db.restrictions.insert(press_start = 5, press_time = "00:00:20")

if db(db.press).count():
    rows = db(db.press.id>0).select()
    for row in rows:
        name_db = "press"+''.join(filter(str.isdigit, row.name))
        db.define_table(name_db,
Field('press_shift','string'),Field('press_force','string'),Field('press_pressure',
'string'))

db.define_table('receive',
                Field('shift','string'),
                Field('F_force','string'),
                Field('pressure','string'))
```

WEB Stranica – JavaScript kôd

main.js

```
$(document).ready(function() {

        var dataPoints1=[];
        var dataPoints2=[];
        var dataPoints3=[];

        var chart1;
        var chart2;
        var chart3;

        var charts=[chart1, chart2, chart3];
        var data = [dataPoints1, dataPoints2, dataPoints3];

        var t = 0;

        var titles = ["POMAK", "SILA", "TLAK"];
        var units = ["cm", "N", "MPa"];
        var colors = ["#CCFF00", "#FF9933", "#FF00CC"];

        for (let i=1;i<4;i++) {
                charts[i-1] = Highcharts.chart('container'+i, {
                        chart: {
                                type: 'spline',
                                animation: Highcharts.svg, // don't animate in old IE
                                marginRight: 10,
                                backgroundColor: 'black',
                        },

                        time: {
                                useUTC: false
                        },

                        title: {
                                text: titles[i-1],
                                style: {
```

```
                                        color: '#808080',
                                        fontWeight: 'bold'
                                }
                        },
                        subtitle: {
                                text: ''
                        },
                        xAxis: {
                                type: 'time',
                                tickPixelInterval: 150,
                                gridLineColor: '#197F07',
                                gridLineWidth: 1,
                                scrollbar: {
                                        enabled: true,
                                        barBackgroundColor: 'black',
                                        barBorderRadius: 7,
                                        barBorderWidth: 0,
                                        buttonBackgroundColor: colors[i-1],
                                        buttonBorderWidth: 0,
                                        buttonArrowColor: 'white',
                                        buttonBorderRadius: 7,
                                        rifleColor: colors[i-1],
                                        trackBackgroundColor: colors[i-1],
                                        trackBorderWidth: 1,
                                        trackBorderColor: colors[i-1],
                                        trackBorderRadius: 7
                                },
                                labels: {
                                        formatter: function() {
                                        return this.value*250 + "ms";
                                        }
                                },
                        },
                        yAxis: {
                                title: {
                                        text: titles[i-1].toLowerCase()
                                },
                                labels: {
                                        format: '{value}'+units[i-1]
                                },
                                tickInterval: 0.2,
                                gridLineColor: '#197F07',
                                gridLineWidth: 1,
                                plotLines: [{
                                        value: 0.6,
                                        width: 1,
                                        color: '#fff'
                                }]
                        },
                        tooltip: {
                                headerFormat: 'vrijeme: {point.x}00 ms<br>',
                                pointFormat: titles[i-1].toLowerCase()+':
{point.y:.2f}'+ units[i-1],
                                shared: true
                        },
```

```
                    legend: {
                              enabled: false
                    },
                    exporting: {
                              enabled: false
                    },
                    plotOptions: {
                              series: {
                                        color: colors[i-1]
                              }
                    },
                    series: [{
                              name: titles[i-1].toLowerCase(),
                              data: data[i-1]
                    }]
          });
}

var charts_upd = function charts_update() {
          $.getJSON( "getdata", function( data ) {
                    for (var j=0; j<data.length; j++) {
                              if (data[j][0] != undefined) {

charts[0].series[0].addPoint(parseFloat(data[j][0]),false);

charts[1].series[0].addPoint(parseFloat(data[j][1]),false);

charts[2].series[0].addPoint(parseFloat(data[j][2]),false);
                              }
                    }
                    charts[0].redraw();
                    charts[1].redraw();
                    charts[2].redraw();
                    var lastPoint1 = charts[0].series[0].data.length - 1;
                    charts[0].xAxis[0].setExtremes(
                              lastPoint1 - (15 - 1), // min
                              lastPoint1); // max
                    var lastPoint2 = charts[1].series[0].data.length - 1;
                    charts[1].xAxis[0].setExtremes(
                              lastPoint2 - (15 - 1), // min
                              lastPoint2); // max
                    var lastPoint3 = charts[2].series[0].data.length - 1;
                    charts[2].xAxis[0].setExtremes(
                              lastPoint3 - (15 - 1), // min
                              lastPoint3); // max
                    var lastp1 = charts[0].series[0].data[lastPoint1].y
                    var lastp2 = charts[1].series[0].data[lastPoint2].y
                    var lastp3 = charts[2].series[0].data[lastPoint3].y
                    charts[0].setTitle(null, { text: lastp1+"cm"});
                    charts[1].setTitle(null, { text: lastp2+"N"});
                    charts[2].setTitle(null, { text: lastp3+"MPa"});
          })
};

$('#recButton').addClass("notRec");
```

```
var switchButton = document.querySelector(".switch-button");
var switchBtnRight = document.querySelector(".switch-button-case.right");
var switchBtnLeft = document.querySelector(".switch-button-case.left");
var activeSwitch = document.querySelector(".active");

function switchLeft() {
          switchBtnRight.classList.remove("active-case");
          switchBtnLeft.classList.add("active-case");
          activeSwitch.style.left = "0%";
}

function switchRight() {
          switchBtnRight.classList.add("active-case");
          switchBtnLeft.classList.remove("active-case");
          activeSwitch.style.left = "50%";
}

switchBtnLeft.addEventListener(
          "click",
          function() {
                    switchLeft();
          },
          false
);

switchBtnRight.addEventListener(
          "click",
          function() {
                    switchRight();
          },
          false
);

var shift_force_val;

function update_shift_force(dir) {
          if ($( ".switch-button > .right" ).hasClass( "active-case" )) {
                    shift_force_val = parseFloat($( "#shift" ).val());
                    if (shift_force_val >= 0.0 && shift_force_val <= 2.1) {
                              rot_value = (280.0/2.1)*shift_force_val;
                              if (dir == "up") {
                                        shift_force_val += 0.1;
                                        rot_value += 13.33;
                              }
                              else {
                                        shift_force_val -= 0.1;
                                        rot_value -= 13.33;
                              }
                              if (shift_force_val < 0.0) {
                                        shift_force_val = 0.0;
                              }
                              else if (shift_force_val > 2.1) {
                                        shift_force_val = 2.1;
                              }
```

```
                                        shift_force_val = shift_force_val.toFixed(1);
                                        rot_value = rot_value.toFixed(2);
                                        rot_value_string = "rotate("+rot_value+"deg)";
                                        $('#shift').val(shift_force_val);
                                        $(".shift_wrapper > .knob-surround >
.knobvalue").html(shift_force_val);
                                        $('.shift_wrapper > .knob-surround >
.knob').css('transform', rot_value_string);
                                }
                        }
                        else if ($( ".switch-button > .left" ).hasClass( "active-case" )) {
                                shift_force_val = parseInt($( "#force" ).val());
                                if (shift_force_val >= 0 && shift_force_val <= 1000) {
                                        rot_value = (280.0/1000)*shift_force_val;
                                        if (dir == "up") {
                                                shift_force_val += 10;
                                                rot_value += 13.33;
                                        }
                                        else {
                                                shift_force_val -= 10;
                                                rot_value -= 13.33;
                                        }
                                        if (shift_force_val < 0) {
                                                shift_force_val = 0;
                                        }
                                        else if (shift_force_val > 1000) {
                                                shift_force_val = 1000;
                                        }
                                        rot_value_string = "rotate("+rot_value+"deg)";
                                        $('#force').val(shift_force_val);
                                        $(".force_wrapper > .knob-surround >
.knobvalue").html(shift_force_val);
                                        $('.force_wrapper > .knob-surround >
.knob').css('transform', rot_value_string);
                                }
                        }
                }

                //webkitURL is deprecated but nevertheless
                URL = window.URL || window.webkitURL;
                var gumStream;
                //stream from getUserMedia()
                var rec;
                //Recorder.js object
                var input;
                //MediaStreamAudioSourceNode we'll be recording
                // shim for AudioContext when it's not avb.
                var AudioContext = window.AudioContext || window.webkitAudioContext;
                //new audio context to help us record
                var recordButton = document.getElementById("recButton");

                $('#recButton').click(function(){
                        if($('#recButton').hasClass('notRec')){
                                startRecording();
                                $('#recButton').removeClass("notRec");
```

```
                    $('#recButton').addClass("Rec");
                    setTimeout(function(){
                              stopRecording();
                              $('#recButton').removeClass("Rec");
                              $('#recButton').addClass("notRec");
                    }, 2000);
          }
});

function startRecording() {
          var audioContext = new AudioContext;
          var constraints = {
                    audio: true,
                    video: false
          }

          navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
                    gumStream = stream;
                    input = audioContext.createMediaStreamSource(stream);
                    rec = new Recorder(input, {
                              numChannels: 1
                    })
                    rec.record()
          }).catch(function(err) {
                    recordButton.disabled = false;
          });
}

function stopRecording() {
          rec.stop();
          gumStream.getAudioTracks()[0].stop();
          rec.exportWAV(send_to_server);
}

function send_to_server(blob) {
          var filename = new Date().toISOString();
          $.ajax({
                    url :  "get_wav",
                    type: 'POST',
                    data: fd,
                    contentType: false,
                    processData: false,
                    success: function(data) {
                              $.each( data, function( key, val) {
                              alert(data);
                              })

                    },
                    error: function() {
                      alert("not so boa!");
                    }
          });
          var xhr = new XMLHttpRequest();
          xhr.onload = function(e) {
                    if (this.readyState === 4) {
```

```
                                        var server_return = JSON.parse(this.responseText);
                                        server_return = server_return["value"];
                                        update_html(server_return);
                                }
                        };
                        var fd = new FormData();
                        fd.append("audio_data", blob, filename);
                        xhr.open("POST", "get_wav", true);
                        xhr.send(fd);
                }

                var shift_force_up_int;
                var global_dir;

                function update_html(speech_value){
                        switch (speech_value) {
                                case ("empty"):
                                        console.log("EMPTY");
                                        break;
                                case ("dolje"):
                                        if (!shift_force_up_int) {
                                                shift_force_up_int =
setInterval(update_shift_force, 1000, ["down"]);
                                        }
                                        else if(global_dir == "up") {
                                                clearInterval(shift_force_up_int);
                                                shift_force_up_int =
setInterval(update_shift_force, 1000, ["down"]);
                                        }
                                        global_dir = "down";
                                        break;
                                case ("gore"):
                                        if (!shift_force_up_int) {
                                                shift_force_up_int =
setInterval(update_shift_force, 1000, ["up"]);
                                        }
                                        else if(global_dir == "down") {
                                                clearInterval(shift_force_up_int);
                                                shift_force_up_int =
setInterval(update_shift_force, 1000, ["up"]);
                                        }
                                        global_dir = "up";
                                        break;
                                case ("pomak"):
                                        if($( ".switch-button-case, .left" ).hasClass( "active-
case" )) {
                                                switchRight();
                                        }
                                        break;
                                case ("sila"):
                                        if($( ".switch-button-case, .right" ).hasClass( "active-
case" )) {
                                                switchLeft();
                                        }
                                        break;
```

```
                        case ("stani"):
                                if (shift_force_up_int) {
                                        clearInterval(shift_force_up_int);
                                        shift_force_up_int = null;
                                }
                                if ($('#main_check').is(':checked')) {
                                        if($( ".switch-button > .left" ).hasClass(
"active-case" )) {
                                                var send_force = parseInt($( "#force"
).val());
                                                send_force = 'f'.concat(send_force);
                                                socket_send(send_force);
                                        }
                                        else if($( ".switch-button > .right"
).hasClass( "active-case" )) {
                                                var send_shift = parseFloat($(
"#shift" ).val());
                                                send_shift = 's'.concat(send_shift);
                                                socket_send(send_shift);
                                        }
                                }
                                break;
                        case ("ugasi"):
                                if ($('#main_check').is(':checked')) {
                                        $('#main_check').prop('checked', false);
                                }
                                check_main_check();
                                break;
                        case ("upali"):
                                if ($('#main_check').is(':checked') === false) {
                                        $('#main_check').prop('checked', true);
                                }
                                break;
                        default:
                                console.log("DEFAULT");
                }
        }

        $( ".shift_wrapper > .knob-surround > .knob" ).mouseup(function() {
                if ($('#main_check').is(":checked")){
                        if($( ".switch-button > .right" ).hasClass( "active-case" )) {
                                let send_value = $( "#shift").val();
                                send_value = 's'.concat(send_value);
                                socket_send(send_value);
                        }
                };
        });

        $( ".force_wrapper > .knob-surround > .knob" ).mouseup(function() {
                if ($('#main_check').is(":checked")){
                        if($( ".switch-button > .left" ).hasClass( "active-case" )) {
                                let send_value = $( "#force").val();
                                send_value = 'f'.concat(send_value);
                                socket_send(send_value);
                        }
```

```
                }
        });

        var charts_ref;

        function socket_send(send_value){
                w.postMessage([send_value]);
                clearInterval(charts_ref);
                if (send_value != 0) {
                        charts_ref = setInterval(charts_upd, 1000);
                }

        }

        $("#main_check").click(function () {
                check_main_check();
        });

        function check_main_check() {
                if ($('#main_check').is(":checked")== false){
                        clearInterval(charts_ref);
                        w.postMessage([0]);
                }
        }
        var w;

        function startWorker() {
                if (typeof(Worker) !== "undefined") {
                        if (typeof(w) == "undefined") {
                        w = new Worker("\\pneumatic_press/static/js/main_js_ww.js");
                        }
                        w.onmessage = function(event) {
                                console.log(event.data);
                        };
                }
                else {
                        document.getElementById("result").innerHTML = "Sorry! No Web
Worker support.";
                }
        }

        function stopWorker() {
                w.terminate();
                w = undefined;
        }

        startWorker();

        window.onunload = window.onbeforeunload = (function(){
                socket_send(0);
        })
});
```

admin_calendar.js

```
$(document).ready(function() {
```

```
var inital_id = [];
var events = [];
var events_changed = [];
var counter=1;
var id_counter = [];
$.ajax({
          url:"values2",
          dataType: 'json',
          async: false,
          success:  function( data ) {
                    $.each( data, function( key, val) {
                              id counter.push( {
                                        id     : val[0],
                                        description: val[1]+'/'+val[2],

                              })
                    });
          }
});

$.ajax({
          url:"values",
          dataType: 'json',
          async: false,
          success:  function( data ) {
                    $.each( data, function( key, val) {
                              events.push( {
                                        id     : val[2],
                                        title  : 'Termin '+ val[2].substr(1),
                                        start  : val[0],
                                        end    : val[1]
                              })
                              inital_id.push(val[2]);
                    });
                    counter = 1;
                    for (var i=0; i<inital_id.length;i++) {
                              if (parseInt(inital_id[i].substr(1))+1 > counter) {
                                        counter = parseInt(inital_id[i].substr(1))+1;
                              }
                    }
          }
});

for (var i=0; i<id_counter.length;i++) {
          for (var j=0; j<events.length;j++) {
                    if (id_counter[i].id === events[j].id) {
                              events[j].description = id counter[i].description
                              break
                    }
                    else {}
          }
}

var students=[];
$.ajax({
          url:"students org",
          dataType: 'json',
          async: false,
          success:  function( data ) {
                    $.each( data, function( key, val) {
                              students.push( {
                                        student: val[0],
                                        appointment: val[1],
                                        press_number  : val[2]
                              })
                    });
          }
});
var press=[];
$.ajax({
          url:"press_initial",
          dataType: 'json',
          async: false,
          success:  function( data ) {
                    $.each( data, function( key, val) {
```

```
                                press.push( {
                                        name: val[0],
                                        status: val[1],
                                        rpi_ip: val[2],
                                        mc ip: val[3]
                                })
                        });
                }
        });

        $.ajax({
                url:"get restrictions",
                dataType: 'json',
                async: false,
                success:  function( data ) {
                        document.getElementById("press_start_num").value = data[0][0];
                        document.getElementById("press dur").value = data[0][1];
                }
        });

        for (var i=0; i<students.length; i++) {
                var create_trs = document.createElement('tr');
                document.getElementById("students body").appendChild(create trs)
                var create_ths = document.createElement('th');
                create_ths.setAttribute("scope", "row");
                var nodes = document.createTextNode(i+1);
                create_ths.appendChild(nodes);
                create trs.appendChild(create ths);
                var create_td1s = document.createElement('td');
                var create_td2s = document.createElement('td');
                var create_td3s = document.createElement('td');
                var node1s = document.createTextNode(students[i].student);
                var node2s = document.createTextNode(students[i].appointment);
                create td1s.appendChild(node1s);
                create td2s.appendChild(node2s);
                create_trs.appendChild(create_td1s);
                create_trs.appendChild(create_td2s);
                create_trs.appendChild(create_td3s);
                const create_button_3s = document.createElement('button');
                create button 3s.className = "close modal3 close logo2";
                create button 3s.title = "izbriši studenta";
                create_button_3s.type = "button";
                create_button_3s.innerHTML = "&times;"
                create td3s.appendChild(create button 3s);
        }

        var active_press = 0;
        for (var i=0; i<press.length; i++) {
                thenum2 = press[i].name.match(/\d+/)[0];
                var create_tr2 = document.createElement('tr');
                document.getElementById("press body").appendChild(create tr2);
                var create_th2 = document.createElement('th');
                create_th2.setAttribute("scope", "row");
                create_tr2.appendChild(create_th2);
                var node2 = document.createTextNode(i+1);
                create th2.appendChild(node2);
                var create_td12 = document.createElement('td');
                var node12 = document.createTextNode(press[i].name);
                create_td12.appendChild(node12);
                create_tr2.appendChild(create_td12);
                var create_td22 = document.createElement('td');
                create tr2.appendChild(create td22);
                var create_label2 = document.createElement('label');
                create_label2.setAttribute("class", "switch");
                create_td22.appendChild(create_label2);
                create_input2 = document.createElement('input');
                create input2.setAttribute("type", "checkbox");
                create input2.setAttribute("id", "press"+(thenum2));
                create input2.setAttribute("name", "Preša"+(thenum2));
                create_label2.appendChild(create_input2);
                var create_span2 = document.createElement('span');
                create_span2.setAttribute("class", "slider");
                create label2.appendChild(create span2);

                var create_tr2_ip = document.createElement('tr');
```

```
                    document.getElementById("press ip body").appendChild(create tr2 ip);
                    var create_th2_ip = document.createElement('th');
                    var node2_ip = document.createTextNode(i+1);
                    create_th2_ip.setAttribute("scope", "row");
                    create tr2 ip.appendChild(create th2 ip);
                    create th2 ip.appendChild(node2 ip);
                    var create_td12_ip = document.createElement('td');
                    var node12_ip = document.createTextNode(press[i].name);
                    create_td12_ip.appendChild(node12_ip);
                    create_tr2_ip.appendChild(create_td12_ip);
                    var create_td_ip_2 = document.createElement('td');
                    var create_td_ip_3 = document.createElement('td');
                    var create_td_ip_4 = document.createElement('td');
                    create_tr2_ip.appendChild(create_td_ip_2);
                    create_tr2_ip.appendChild(create_td_ip_3);
                    create_tr2_ip.appendChild(create_td_ip_4);
                    var create_input_ip_2 = document.createElement('input');
                    var create_input_ip_3 = document.createElement('input');
                    const create_button_ip_4 = document.createElement('button');
                    create_input_ip_2.setAttribute("type", "text");
                    create_input_ip_2.setAttribute("id", "rpi_"+(i+1));
                    create_input_ip_2.setAttribute("name", "rpi_"+(i+1));
                    create input ip 2.required = true;
                    create_input_ip_3.setAttribute("type", "text");
                    create_input_ip_3.setAttribute("id", "mc_"+(i+1));
                    create_input_ip_3.setAttribute("name", "mc_"+(i+1));
                    create_input_ip_3.required = true;
                    create button ip 4.className = "close modal3 close logo1";
                    create button ip 4.title = "izbriši prešu";
                    create_button_ip_4.type = "button";
                    create_button_ip_4.innerHTML = "&times;"
                    create_td_ip_2.appendChild(create_input_ip_2);
                    create_td_ip_3.appendChild(create_input_ip_3);
                    create td ip 4.appendChild(create button ip 4);

                    if (press[i].status === true) {
                              if (active_press%2 === 0) {
                              var video_div = document.createElement('div');
                              video_div.setAttribute("class", "row");
                              document.getElementById("video divs").appendChild(video div);
                              }
                              var inner_video_div = document.createElement('div');
                              inner_video_div.setAttribute("class", "column");
                              inner video div.setAttribute("class", "inner");
                              video div.appendChild(inner video div);
                              var video_h2 = document.createElement("h2");
                              video h2.innerHTML = press[i].name;
                              inner_video_div.appendChild(video_h2);
                              var video_img = document.createElement("IMG");
                              video img.setAttribute("height", "80%");
                              video img.setAttribute("width", "80%");
                              video img.setAttribute("src", press[i].rpi ip);
                              video_img.setAttribute("alt", "KONEKCIJA NEUSPJELA");
                              inner_video_div.appendChild(video_img);

                              $( "#press"+ press[i].name.match(/\d+/)[0]).prop( "checked", true
);
                              active_press +=1;
                    }
                    else {
                              $( "#press"+ press[i].name.match(/\d+/)[0]).prop( "checked",
false );
                    }
                    $( "#rpi_"+(i+1)).val(press[i].rpi_ip);
                    $( "#mc_"+(i+1)).val(press[i].mc_ip);
                    $( ".dot").val(active_press);
                    $( "#event user num2").val(active press);
                    document.querySelector("#date_start_2").valueAsDate = new Date();
                    document.querySelector("#data_end_2").valueAsDate = new Date();
          }
          $('#calendar').fullCalendar({
                    eventMouseover: function( event, jsEvent, view ) {
                              $(this).css('background-color', '#006699');
                              $(this).css('color', '#fff');
                    },
```

```
                    eventMouseout: function( event, jsEvent, view ) {
                              $(this).css('background-color', '#fff');
                              $(this).css('color', '#000');
                    },
          eventRender: function(event, element) {
                              element.find('.fc-title').append('<div class="hr-line-solid-no-
margin"></div><span style="font-size: 12px">'+event.description+'</span></div>');
               element.click(function(e) {
                                        if(e.shiftKey) {
                                                  event2=event;

          $("#event_user_num").val(event.description.split('/')[1]);

          $("#date_start_").val($.fullCalendar.moment(event.start._d).format(('YYYY-MM-DD')));

          $("#time start ").val($.fullCalendar.moment(event.start. d).format(('HH:mm:ss')));

          $("#data end ").val($.fullCalendar.moment(event.end. d).format(('YYYY-MM-DD')));

          $("#time_end_").val($.fullCalendar.moment(event.end._d).format(('HH:mm:ss')));
                                                  $("#myModal").modal();
                                                  empty =
document.getElementsByClassName("tbody_modal");
                                                  for (var i=0;i<empty.length; i++) {
                                                            empty[i].innerHTML = ""
                                                  }

                                                  var z = 1;
                                                  for (var i=0; i<students.length;i++){

          if(event.id===students[i].appointment){
                                                                                      var create_tbody =
document.createElement('tbody');

          create_tbody.setAttribute("class", "tbody_modal");

          document.getElementById("modal_table").appendChild(create_tbody);
                                                                                      var create_tr =
document.createElement('tr');

          create_tbody.appendChild(create_tr)
                                                                                      var create_th =
document.createElement('th');

          create th.setAttribute("scope", "row");
                                                                                      var node =
document.createTextNode(z);

          create th.appendChild(node);

          create tr.appendChild(create th);
                                                                                      var create_td1 =
document.createElement('td');
                                                                                      var create_td2 =
document.createElement('td');
                                                                                      var node1 =
document.createTextNode(students[i].student);
                                                                                      var node2 =
document.createTextNode(students[i].press_number);

          create td1.appendChild(node1);

          create_td2.appendChild(node2);

          create_tr.appendChild(create_td1);

          create tr.appendChild(create td2);
                                                                                      z+=1;
                                                                            }
                                                  }

                                                  $('#modal_enter').click(function(e){
                                                            e.stopImmediatePropagation();
```

```
                                                                update max =
$("#event_user_num").val();
                                                                if (isNaN(parseFloat(update_max)) ||
parseFloat(update_max)==null || update_max.indexOf('e') > -1 ||
Number.isInteger(parseFloat(update_max))==false){
                                                                }
                                                                else {
                                                                        var new_description =
event2.description;
                                                                        replacement = update_max;
                                                                        replace this =
event2.description.split('/')[1];
                                                                        var regex = new
RegExp(replace_this+"([^"+replace_this+"]*)$");
                                                                        new_description =
new_description.replace(regex,replacement+'$1');
                                                                        event2.description =
new description;

          $('#calendar').fullCalendar('updateEvent', event2);
                                                                }
                                                                new_start_time = new
Date($("#date start ").val() + 'T' + $("#time start ").val());
                                                                new_end_time = new
Date($("#data_end_").val() + 'T' + $("#time_end_").val());
                                                                if (new_start_time instanceof Date &&
!isNaN(new_start_time) && new_end_time instanceof Date && !isNaN(new_end_time) && new_end_time
> new start time) {
                                                                        event2.start =
new_start_time;
                                                                        event2.end = new_end_time;
                                                                }

          $('#calendar').fullCalendar('updateEvent', event2);
                                                                $('#myModal').modal('toggle');
                                                        });
                                        }
                                        if(e.ctrlKey) {

          $('#calendar').fullCalendar('removeEvents',event. id);
                                        }
                                });
        },
                eventDrop: function( event ) {
                        if (events_changed.includes(event)) {
                                events changed.push(event)}
                },
                eventResize: function( event ){
                        events_changed.push(event)
                },
                dayClick: dayClickCallback,
                locale: 'hr',
                themeSystem: 'bootstrap4',
                defaultView: 'agendaWeek',
                header: {
                        left: 'prev,next today',
                        center: 'title',
                        right: 'month,agendaWeek,agendaDay'
                },
                defaultDate: new Date(),
                allDaySlot: false,
                navLinks: true,
                timezone : 'local',
                editable: true,
                droppable: true,
                eventLimit: true,
                events: events
          });

          $('#calendar').fullCalendar({
                dayClick: dayClickCallback,
                eventRender: eventRenderCallback,
          });

          var slotDate;
```

```
		function dayClickCallback(date){
			slotDate = date;
			$("#calendar").on("mousemove", forgetSlot);
		}

		function eventRenderCallback(event,element){
			element.on("dblclick",function(){
				dblClickFunction(event.start)
			});
		}

		function forgetSlot(){
			slotDate = null;
			$("#calendar").off("mousemove", forgetSlot);
		}

		function dblClickFunction(date){
			var start = new Date(date.toJSON());
			var end = new Date(date.toJSON());
			end.setHours(end.getHours()+2);
			var users_number = $("#num_useres").val();
			if (isNaN(parseFloat(users_number)) || parseFloat(users_number)==null ||
users_number.indexOf('e') > -1) {
				users_number = "0"
			}
			var new_event = {
				id     : 't'+(counter),
				title  : 'Termin '+(counter),
				start  : moment(start),
				description: '0/'+users_number,
				end : moment(end),
				displayEventEnd : true
				}
			$("#calendar").fullCalendar( "renderEvent", new event, 'stick')
			counter+=1;
		}

		$("#calendar").dblclick(function() {
			if(slotDate){
				dblClickFunction(slotDate);
			}
		});

		$('.save').click(function(event) {
			$("body").css("cursor", "progress");
			event.preventDefault();
			var new_events = $('#calendar').fullCalendar('clientEvents');
			var event_start_date = [];
			var event_final_date = [];
			var event_appointment_id = [];
			var deleted_events = [];
			var max_users = [];
			var max_users2=[];

			for (var i=0; i<new_events.length; i++) {

		event_start_date.push($.fullCalendar.moment(new_events[i].start._i).format(('YYYY-
MM-DD HH:mm')))

		event final date.push($.fullCalendar.moment(new events[i].end. i).format(('YYYY-MM-
DD HH:mm')))
				event appointment id.push(new events[i].id);
				max_users2.push(new_events[i].description.split('/')[1])
				for (var j=0; j<id_counter.length; j++) {
					if (new_events[i].id === id_counter[j].id) {
						if (new_events[i].description !=
id counter[j].description) {
							max users.push([new events[i].id,
new_events[i].description.split('/')[1]]);
						}
					break
					}
					else if ((j+1)>=id_counter.length){
```

```
                                                  max users.push([new events[i].id,
new_events[i].description.split('/')[1]]);
                                        }
                              }
                    }

                    for (var i=0; i<inital_id.length; i++) {
                              if( $.inArray(inital_id[i], event_appointment_id) != -1){
                              }
                              else {
                                        deleted events.push(inital id[i])
                              }
                    }
                    $.ajax({
                              url:"save",
                              type: 'post',
                              data: { "event start date": event start date, "event final date":
event final date,
                                        "event appointment id": event appointment id,
"deleted_events": deleted_events,
                                        "max_users2":max_users2},
                              dataType: 'json',
                              success: function(data){
                                        $("body").css("cursor", "default");
                              },
                              error: function(jqXHR, textStatus, errorThrown){
                                        console.log(errorThrown);
                                        alert(textStatus);
                              }
                    });

          });

          $('.reset').click(function(e) {
                    e.preventDefault();
                    document.getElementById("modal6_title").innerHTML = "BRISANJE PODATAKA";
                    document.getElementById("modal_h3").innerHTML = "Jeste li sigurni da želite
izbrisati bazu podataka?";
                    $('#myModal6').modal('toggle');
                    $('#app_confirm').click(function(e) {
                              e.stopImmediatePropagation();
                              $("body").css("cursor", "progress");
                              $.ajax({
                                        url:"reset",
                                        success: function(data){
                                                  $('#calendar').fullCalendar( "removeEvents");
                                                  $('#myModal6').modal('toggle');
                                                  counter = 1;
                                                  $("body").css("cursor", "default");
                                        },
                                        error: function(jqXHR, textStatus, errorThrown){
                                                  console.log(errorThrown);
                                                  alert(textStatus);
                                        }
                              });
                    });
          });
          $("#li_holder").click(function(e){
                    e.preventDefault();
                    if(e.shiftKey) {
                              $('#myModal5').modal('toggle');

                    }
                    else {
                              $(".dot").removeAttr('disabled');
                              $( ".dot" ).addClass( "dot_append");
                              $( ".dot" ).focus();
                    }
          });
          $( "#li_holder" ).mouseleave(function() {
                    if ($( ".dot" ).hasClass( "dot_append" )){
                              $(".dot").attr('disabled');
                              $( ".dot" ).removeClass( "dot append" );
                              $( ".dot" ).blur();
                    }
```

```
        });


        var form_picker;
        $("#menu-button11").click(function(e){
                e.preventDefault();
                document.getElementById('h4 title').innerHTML = 'DODAVANJE TERMINE';
                form_picker = 1;
                $('#myModal2').modal('toggle');
        })

        $("#menu-button12").click(function(e){
                e.preventDefault();
                document.getElementById('event_user_holder').style.display = 'none';
                document.getElementById('time_period_holder').style.display = 'none';
                document.getElementById("event_user_num2").required = false;
                document.getElementById("time period").required = false;
                document.getElementById('h4 title').innerHTML = 'BRISANJE TERMINA';
                form picker = 2;
                $('#myModal2').modal('toggle');
        })

        $( "#form12" ).submit(function( event ) {
                event.preventDefault();
                if (form_picker===1) {
                        var data = $(this).serializeArray();
                        var start = new Date(data[0].value+'T'+data[1].value);
                        var stop = new Date(data[2].value+'T'+data[3].value);
                        var add = [];
                        var end_date = new Date(start);
                        var period = data[5].value;
                        period_hours = parseInt(period.split(':')[0]);
                        period_minutes = parseInt(period.split(':')[1]);
                        end_date.setHours(end_date.getHours()+period_hours);
                        end_date.setMinutes(end_date.getMinutes()+period_minutes);
                        var users_number = data[4].value;
                        if (isNaN(parseFloat(users_number)) ||
parseFloat(users_number)==null || users_number.indexOf('e') > -1) {
                                users_number = "0"
                        }
                        while (start < stop) {
                                var st_date = new Date(start);
                                var fi_date = new Date(end_date);
                                add.push( {
                                        id      : 't'+(counter),
                                        title   : 'Termin '+(counter),
                                        description: '0/'+users number,
                                        start   : st_date,
                                        end     : fi_date
                                })
                                start.setHours(start.getHours()+period_hours);
                                start.setMinutes(start.getMinutes()+period_minutes);
                                end_date.setHours(end_date.getHours()+period_hours);

        end_date.setMinutes(end_date.getMinutes()+period_minutes);
                                counter+=1;
                        }
                        $("#calendar").fullCalendar( 'addEventSource', add )
                }
                else {
                        var data = $(this).serializeArray();
                        var start = new Date(data[0].value+'T'+data[1].value);
                        var stop = new Date(data[2].value+'T'+data[3].value);
                        $('#calendar').fullCalendar( 'removeEvents', function(event) {
                                if(event.start._d>=new Date(start) && event.end._d<=new
Date(stop)){
                                        return true;
                                }
                        });
                }
                $('#calendar').fullCalendar('render');
                $('#myModal2').modal('toggle');
        });

        $('#myModal2').on('hidden.bs.modal', function () {
```

```
            document.getElementById('event user holder').style.display = 'flex';
            document.getElementById('time_period_holder').style.display = 'flex';
            document.getElementById("event_user_num2").required = true;
            document.getElementById("time_period").required = true;
            $('#form12')[0].reset();
            $( "#event user num2").val(active press);
            document.querySelector("#date_start_2").valueAsDate = new Date();
            document.querySelector("#data_end_2").valueAsDate = new Date();
})

$(".press_status").click(function(e){
            e.preventDefault();
            $('#myModal3').modal('toggle');
});


$( "#press_form" ).submit(function( event ) {
            event.preventDefault();
            $("body").css("cursor", "progress");
            var data = $(this).serializeArray();
            $.ajax({
                        url:"press_update",
                        type: 'post',
                        data: data,
                        dataType: 'json',
                        success: function(data){
                                    $("body").css("cursor", "default");
                                    location.reload();
                        },
                        error: function(jqXHR, textStatus, errorThrown){
                                    console.log(errorThrown);
                                    alert(textStatus);
                        }
            });
});

$(".press_conf").click(function(e){
            e.preventDefault();
            $('#myModal4').modal('toggle');
});

$(".press_restrictions").click(function(e){
            e.preventDefault();
            $('#myModal7').modal('toggle');
});

$( "#ip_form" ).submit(function( event ) {
            event.preventDefault();
            $("body").css("cursor", "progress");
            var data = $(this).serializeArray();
            var new_press_name = [];
            var new_press_rpi_ip = [];
            var new_press_mc_ip = [];
            var new_press_status = [];
            var new_press = [];
            for (var i=0;i<data.length;i+=2) {
                        thenum = data[i].name.match(/\d+/)[0];
                        new_press.push({name:'Preša'+thenum,
                                                                rpi_ip:data[i].value,
                                                                mc_ip:data[i+1].value,
                                                                status: false})
            }
            for (var i=0; i<new_press.length;i++) {
                        var inc_z=0;
                        for (var j=0; j<press.length; j++) {
                                    inc_z++;
                                    if (new_press[i].name === press[j].name){
                                                new_press[i].status = press[j].status;
                                                break
                                    }
                                    else if (inc_z >= j) {
                                                new_press[i].status=false;
                                    }
                                    else {
                                                continue
```

```
                                        }
                                }
                                new_press_name.push(new_press[i].name);
                                new_press_rpi_ip.push(new_press[i].rpi_ip);
                                new press mc ip.push(new press[i].mc ip);
                                new press status.push(new press[i].status);
                        }
                        $.ajax({
                                url:"press_conf_update",
                                type: 'post',
                                data: {"name":new press name, "mc ip":new press mc ip,
                                "rpi ip":new press rpi ip, "dc active":new press status},
                                dataType: 'json',
                                success: function(data){
                                        $("body").css("cursor", "default");
                                        location.reload();
                                },
                                error: function(jqXHR, textStatus, errorThrown){
                                        console.log(errorThrown);
                                        alert(textStatus);
                                }
                        });
        });

        $( "#restrictions_form" ).submit(function( event ) {
                event.preventDefault();
                $("body").css("cursor", "progress");
                var data = $(this).serializeArray();
                $.ajax({
                        url:"restrictions_update",
                        type: 'post',
                        data: data,
                        dataType: 'json',
                        success: function(data){
                                $('#myModal7').modal('toggle');
                                $("body").css("cursor", "default");
                        },
                        error: function(jqXHR, textStatus, errorThrown){
                                console.log(errorThrown);
                                alert(textStatus);
                        }
                });
        });

        $(document).on("click", ".logo1", function(){
                $(this).closest('tr').remove()
        });
        var deleted_students=[];
        $(document).on("click", ".logo2", function(){
                deleted_students.push($(this).parent().siblings("td:nth-of-
type(1)").text());
                $(this).closest('tr').remove();
        });
        $("#students_save").click(function () {
                $("body").css("cursor", "progress");
                $.ajax({
                        url:"delete students",
                        type: 'post',
                        data: {deleted_students},
                        dataType: 'json',
                        success: function(data){
                                $("body").css("cursor", "default");
                                location.reload();
                        },
                        error: function(jqXHR, textStatus, errorThrown){
                                console.log(errorThrown);
                                alert(textStatus);
                        }
                });
        });
        $("#addrows").click(function () {
                last_th_val=$('#press_ip_body th:last').html();
                last td val=$('#press ip body tr:last td:nth-last-of-type(4)').html();
                try {
                        last_td_val_num = last_td_val.match(/\d+/)[0];
```

```
                    }
                    catch(err) {
                              last_td_val_num = null;
                    }
                    if (last_th_val == null){
                              last th val="1";
                    }
                    else {
                              last_th_val=parseInt(last_th_val)+1;
                    }
                    if (last_td_val_num == null){
                              last td val num="1";
                              last td val="Preša1"
                    }
                    else {
                              last_td_val_num=parseInt(last_td_val_num)+1;
                              last td val="Preša"+last td val num;
                    }
                    var create_tr2_ip = document.createElement('tr');
                    document.getElementById("press_ip_body").appendChild(create_tr2_ip);
                    var create_th2_ip = document.createElement('th');
                    var node2_ip = document.createTextNode(last_th_val);
                    create th2 ip.setAttribute("scope", "row");
                    create_tr2_ip.appendChild(create_th2_ip);
                    create_th2_ip.appendChild(node2_ip);
                    var create_td12_ip = document.createElement('td');
                    var node12_ip = document.createTextNode(last_td_val);
                    create td12 ip.appendChild(node12 ip);
                    create tr2 ip.appendChild(create td12 ip);
                    var create_td_ip_2 = document.createElement('td');
                    var create_td_ip_3 = document.createElement('td');
                    var create_td_ip_4 = document.createElement('td');
                    create_tr2_ip.appendChild(create_td_ip_2);
                    create tr2 ip.appendChild(create td ip 3);
                    create tr2 ip.appendChild(create td ip 4);
                    var create_input_ip_2 = document.createElement('input');
                    var create_input_ip_3 = document.createElement('input');
                    var create_button_ip_4 = document.createElement('button');
                    create_input_ip_2.setAttribute("type", "text");
                    create input ip 2.setAttribute("id", "rpi "+(last td val num));
                    create input ip 2.setAttribute("name", "rpi "+(last td val num));
                    create_input_ip_2.required = true;
                    create_input_ip_3.setAttribute("type", "text");
                    create input ip 3.setAttribute("id", "mc "+(last td val num));
                    create input ip 3.setAttribute("name", "mc "+(last td val num));
                    create input ip 3.required = true;
                    create button ip 4.className = "close modal3 close btn btn-default logo1";
                    create_button_ip_4.title = "izbriši prešu";
                    create_button_ip_4.type = "button";
                    create button ip 4.innerHTML = "&times;"
                    create td ip 2.appendChild(create input ip 2);
                    create td ip 3.appendChild(create input ip 3);
                    create_td_ip_4.appendChild(create_button_ip_4);
          });
          $( ".video_surveillance" ).click(function(e) {
                    e.preventDefault();
                    $( ".overlay video" ).addClass('overlay-open');
          });

          $( ".overlay-close" ).click(function(e) {
                    e.preventDefault();
                    $( ".overlay video" ).removeClass( 'overlay-open' );
          });
});
```

user_calendar.js

```
$(document).ready(function() {
          var initial_events = [];
          $.ajax({
                    url:"values",
                    dataType: 'json',
                    async: false,
```

```
                    success:  function( data ) {
                                        $.each( data, function( key, val) {
                                                  initial_events.push( {
                                                            id      : val[2],
                                                            title   : 'Termin '+ val[2].substr(1),
                                                            start   : val[0],
                                                            end     : val[1]
                                                  })
                                        });
                    }
          });


          var exclude_id = [];
          $.ajax({
                    url:"check_if_6",
                    dataType: 'json',
                    async: false,
                    success:  function( data ) {
                              for (var i=0; i<data.length; i++){
                                        exclude_id.push(data[i])
                              }
                    }
          });

          var events=[];
          var current_date = new Date().toJSON();
          for (var i=0; i<initial_events.length; i++) {
                    var end_date = new Date(initial_events[i].end).toJSON();
                    var start_date = new Date(initial_events[i].start).toJSON()
                    if (end_date > current_date &&
exclude_id.includes(initial_events[i].id)==false) {
                              if (start_date < current_date) {
                                        events.push( {
                                                  id      : initial events[i].id,
                                                  title   : initial events[i].title,
                                                  start   : initial events[i].start,
                                                  end     : initial events[i].end,
                                                  className : 'first_event'
                                        })
                              }
                              else {
                                        events.push( {
                                                  id      : initial events[i].id,
                                                  title   : initial events[i].title,
                                                  start   : initial events[i].start,
                                                  end     :
initial_events[i].end
                                        })
                              }
                    }
          }

          var cal_header;
          var cal_height;
          var cal_view;
          var navLink_boolean;

          window.mobilecheck = function() {
                    var check = false;

          (function(a){if(/(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|comp
al|elaine|fennec|hiptop|iemobile|ip(hone|od)|iris|kindle|lge
|maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm(
os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|link)|vodafon
e|wap|windows ce|xda|xiino/i.test(a)||/1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a
wa|abac|ac(er|oo|s\-
)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s
)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-
|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|_)|g1 u|g560|gene|gf\-
5|g\-mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-|
| |a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-
|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt(
|\/)|klon|kpt |kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-
```

```
w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-
cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-
2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-
|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-
8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-a|qc(07|12|21|32|60|\-[2-
7]|i\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-
)|sdk\/|se(c(\-|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-
0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-
mo|to(pl|sh)|ts(70|m\-|m3|m5)|tx\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-
3]|\-v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g
|nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-/i.test(a.substr(0,4))) check =
true;})(navigator.userAgent||navigator.vendor||window.opera);
                    if (cal_header == null) {
                              if (check === true) {
                                        //mobile
                                        cal header = {
                                                  left: '',
                                                  center: 'title',
                                                  right: ''
                                        }
                                        cal_height = "parent";
                                        cal view = "listWeek";
                                        navLink_boolean = false;
                              }
                              else {
                                        //desktop
                                        cal header = {
                                                  left: 'prev,next today',
                                                  center: 'title',
                                                  right: 'month,agendaWeek,agendaDay'
                                        }
                                        cal_height = "default";
                                        cal view = "agendaWeek";
                                        navLink boolean = true;
                              }
                    }
                    return check;
          };

          window["mobilecheck"]()
          if (window["mobilecheck"]()===true) {
                    document.addEventListener('touchstart', handleTouchStart, false);
                    document.addEventListener('touchmove', handleTouchMove, false);

                    var xDown = null;
                    var yDown = null;

                    function getTouches(evt) {
                      return evt.touches ||
                                        evt.originalEvent.touches;
                    }

                    function handleTouchStart(evt) {
                              const firstTouch = getTouches(evt)[0];
                              xDown = firstTouch.clientX;
                              yDown = firstTouch.clientY;
                    };

                    function handleTouchMove(evt) {
                              if ( ! xDown || ! yDown ) {
                                        return;
                              }

                              var xUp = evt.touches[0].clientX;
                              var yUp = evt.touches[0].clientY;

                              var xDiff = xDown - xUp;
                              var yDiff = yDown - yUp;

                              if ( Math.abs( xDiff ) > Math.abs( yDiff ) ) {
                                        if ( xDiff > 0 ) {
                                                  $('#calendar').fullCalendar('next');
                                        } else {
                                                  $('#calendar').fullCalendar('prev');
```

```
                                        }
                                }
                                xDown = null;
                                yDown = null;
                        };

                        $('#holder').css('margin-top', '20%');
                        $('#holder').css('height', '100vh');
                        $('.fc-center h2').css('font-size', '24px');
                        $('#calendar').css('max-width', '100%');
                }
                $('#calendar').fullCalendar({
                        eventMouseover: function( event, jsEvent, view ) {
                                $(this).css('background-color', '#006699');
                                $(this).css('color', '#fff');
                                $(this).css('cursor', 'pointer');
                        },

                        eventMouseout: function( event, jsEvent, view ) {
                                $(this).css('background-color', '#fff');
                                $(this).css('color', '#000');
                        },

                        eventClick: function(calEvent, jsEvent, view) {
                                appointment_id_chosen = calEvent.id;
                                document.getElementById("h5_start").innerHTML = "";
                                document.getElementById("h5_stop").innerHTML = ""
                                document.getElementById("h5_start").innerHTML = "OD:  " +
$.fullCalendar.moment(calEvent.start._i).format('DD.MM.YYYY. HH:mm');
                                document.getElementById("h5_stop").innerHTML = "DO:  " +
$.fullCalendar.moment(calEvent.end._i).format('DD.MM.YYYY. HH:mm');
                                $( "div.demo-container" ).html();
                                $('#myModal6').modal('toggle');
                                $("#app_confirm").click(function(e){
                                        $("body").css("cursor", "progress");
                                        e.stopImmediatePropagation();
                                        $.ajax({
                                                url: "save_date",
                                                type: 'post',
                                                dataType: 'json',
                                                data: { "appointment_id":
appointment_id_chosen},
                                                success: function(data){
                                                        if (data.status === 200){
                                                                $("body").css("cursor",
"default");
                                                                window.location.href =
"appointment_chosen";
                                                        }
                                                        else if (data.status === 300) {
                                                                $("body").css("cursor",
"default");
                                                                alert(data.data);
                                                                window.location.href =
"calendar";
                                                        }
                                                },
                                                error: function(jqXHR, textStatus,
errorThrown){
                                                        $("body").css("cursor", "default");
                                                        console.log(errorThrown);
                                                        alert(textStatus);

                                                }
                                        });
                                });
                        },
                        locale: 'hr',
                        themeSystem: 'bootstrap4',
                        defaultView: cal_view,
                        header: cal_header,
                        defaultDate: new Date(),
                        timezone : 'local',
                        allDaySlot: false,
                        navLinks: navLink_boolean,
```

```
                    height: cal height,
                    editable: false,
                    eventLimit: true,
                    events: events
          });
});
```

main_js_ww.js

```
var starter;

function socket_send(send_value){
          var start_int = function start_int() {
                    var xhttp = new XMLHttpRequest();

                    var ref_value = new FormData();
                    ref value.append("socket value", send value);
                    xhttp.open("POST", "\\pneumatic_press/default/socket_communication", true);
                    xhttp.send(ref_value);
          }
          clearInterval(starter);
          if (send_value != 0) {
                    start_int();
                    send_value = [];
                    starter = setInterval(start_int, 500);
          }
          else {
                    start int();
          }
}

onmessage = function(e) {
          socket send(e.data[0]);
}
```