Telemetrija stanja KOPACK vozila

Pejić, Klara

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Klara Pejić

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Prof. dr. sc. Mladen Crneković, dipl. ing.

Student:

Klara Pejić

Zagreb, 2019.

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

Zahvaljujem se prvenstveno svom mentoru prof. dr. sc. Mladenu Crnekoviću na ukazanom povjerenju, na podršci, strpljenju i razumijevanju te na silnoj pomoći i odgovorima na brojna pitanja.

Zahvaljujem se, također, svim svojim prijateljicama i prijateljima koji su uvijek bili tu za mene i vjerovali u mene te kolegama koji su učinili ovo studiranje lakšim i posebna im hvala na pruženoj pomoći oko završnog rada, ali i tijekom cijelog studiranja.

Veliku zahvalnost iskazujem cijeloj svojoj obitelji, a posebno roditeljima i bratu koji su me uvijek podržavali, pružali ljubav i bili prisutni u svim trenutcima moga života, pa tako i tijekom ovog studiranja.

Klara Pejić

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: **KLARA PEJIĆ** Mat. br.: 0035204938

Naslov rada na hrvatskom jeziku: **TELEMETRIJA STANJA KOPACK VOZILA**

Naslov rada na engleskom jeziku: **TELEMETRY STATE OF KOPACK VEHICLE**

Opis zadatka:

Za potrebe ocjene nekog sustava i njegovog daljnjeg unapređenja potrebno je poznavati realna eksploatacijska stanja procesa. Prikupljanje i obrada informacija za KOPACK invalidsko vozilo odvija se preko Bluetooth veze po unaprijed definiranom protokolu, a sastoji se od slušanja RS485 komunikacije između motora i centralne upravljačke jedinice.

U radu je potrebno izgraditi sučelje koje će omogućiti prikupljanje i grafički prikaz odabranih elemenata procesa (pozicija, brzina, struja) za sve motore i inklinometar. Svi podaci moraju se pohranjivati na disk u svrhu naknadnog korištenja.

Potrebno je navesti korištenu literaturu i ostale izvore informacija, te eventualno dobivenu pomoć.

Zadatak zadan:
29. studenog 2018.

Rok predaje rada:
1. rok: 22. veljače 2019.
2. rok (izvanredni): 28. lipnja 2019.
3. rok: 20. rujna 2019.

Predviđeni datumi obrane:
1. rok: 25.2. - 1.3. 2019.
2. rok (izvanredni): 2.7. 2019.
3. rok: 23.9. - 27.9. 2019.

Zadatak zadao:

Prof.dr.sc. Mladen Crneković

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
A		Adresa slave uređaja
C		Check Sum
F		Informacija o motoru
I	%	Struja
P	°	Pozicija
V	%	Brzina

SAŽETAK

Ovaj završni rad bavi se temom telemetrije, točnije telemetrije vozila. Na početku rada objašnjeno je što je to pojam telemetrija i kada se počela koristiti. Navedena su neka područja primjene i objašnjen je način upotrebe te je istaknuta njena važnost. Napravljena je telemetrija KOPACK invalidskog vozila stoga je bolje opisano to vozilo i serijska komunikacija njegove centralne upravljačke jedinice s drugim računalima. Isprogramirano je sučelje koje prikuplja podatke i grafički ih prikazuje u realnom vremenu u programskom jeziku Python, što je detaljnije objašnjeno u drugom dijelu rada. Na kraju su prikazani i prokomentirani dobiveni rezultati.

Ključne riječi: telemetrija vozila, realno vrijeme, serijska komunikacija

SUMMARY

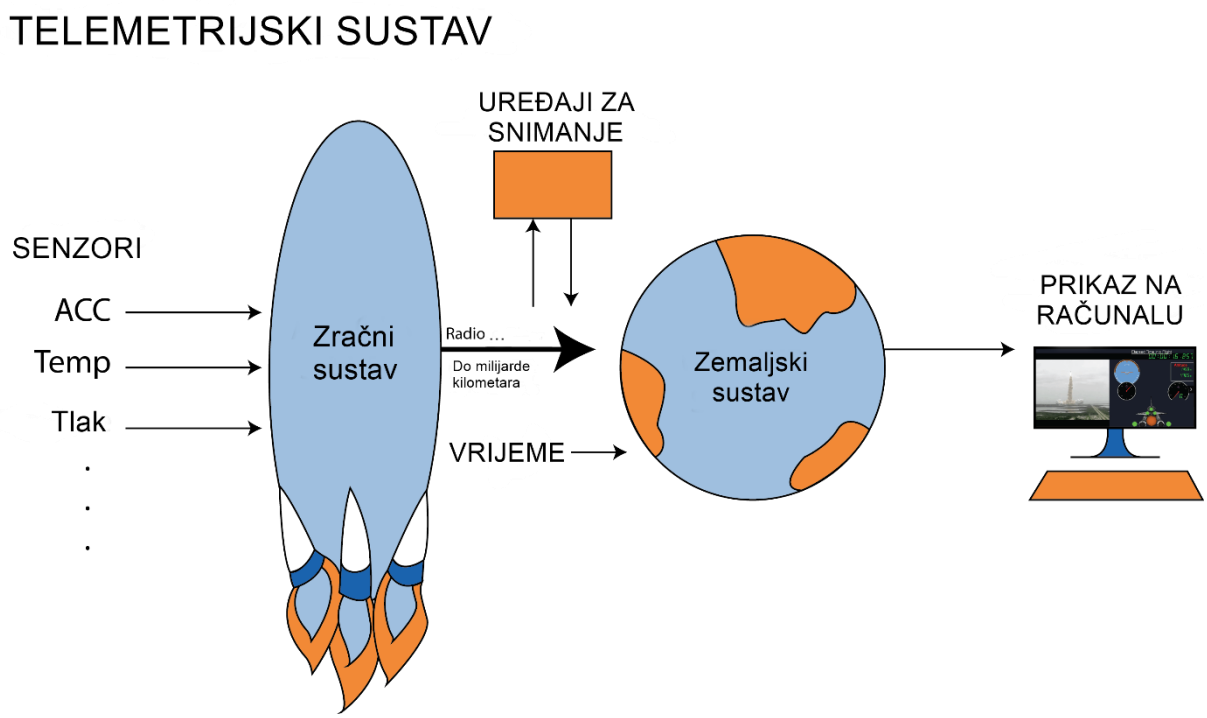
This final paper's topic is telemetry, specifically it deals with vehicular telemetry. Firstly, the term telemetry is detailedly explained and its first usage has been mentioned. Secondly, some specific areas in which telemetry is applied are outlined and its use in those areas is described, accentuating the importance of telemetry. The telemetry that has been made is of KOPACK wheelchair, therefore the wheelchair has been better described along with serial communication of his central processing unit with other computers. Furthermore, an interface that collects data and displays it graphically in real-time has been programmed in Python programming language, which is thoroughly explained in the latter part of the paper. Finally, the obtained results are presented and discussed.

Keywords: vehicle telemetry, real-time, serial communication

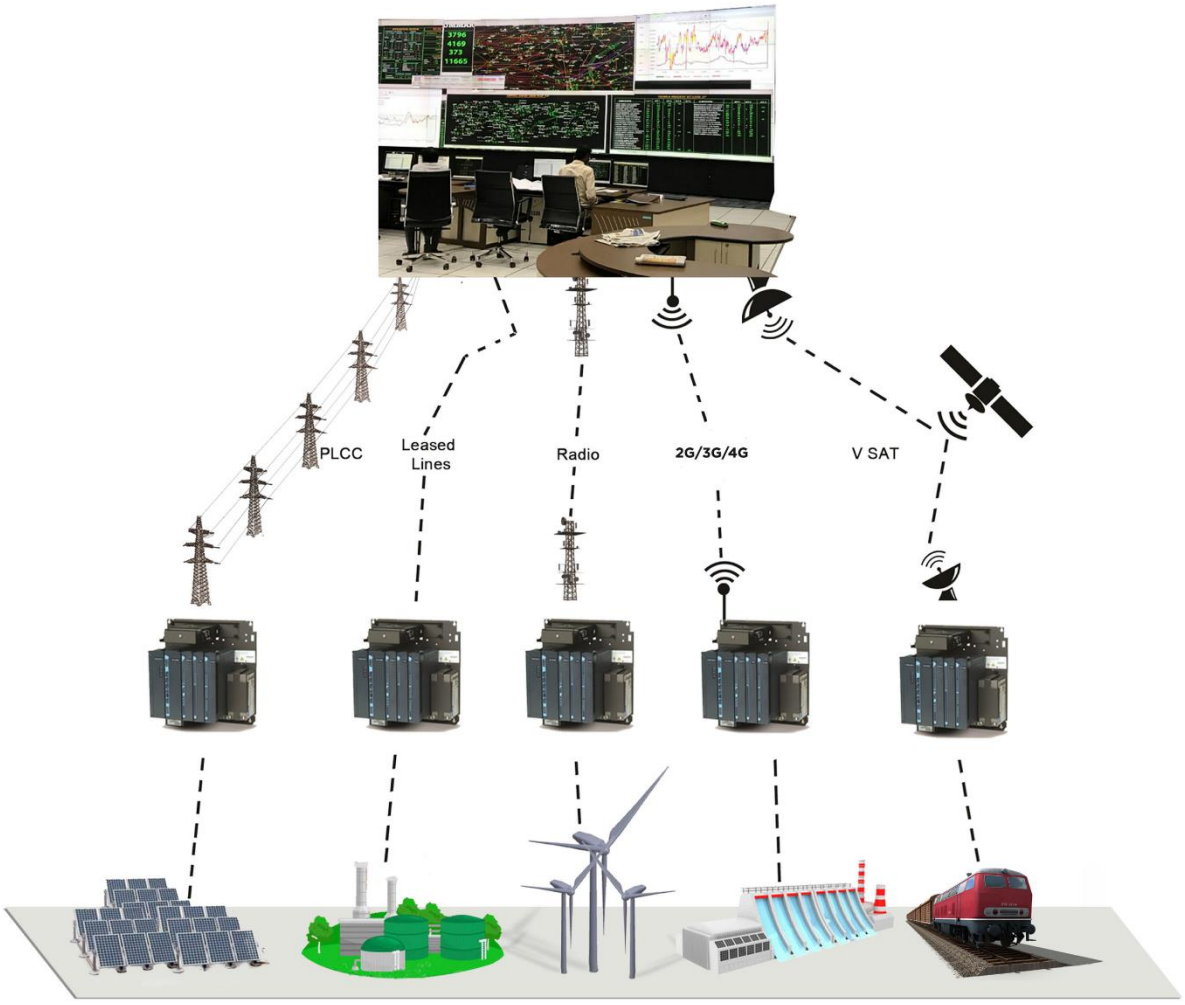
1. UVOD

Riječ telemetrija izvedena je iz grčkih riječi *tele* što znači daleko i *metron* što znači mjera, u doslovnom prijevodu telemetrija bi značila daleka mjerenja. Telemetrija se odnosi na prijenos podataka koje prikuplja senzor (jedan ili više njih) na mjernom mjestu i šalje ih računalu koje ispisuje podatke u zadanom obliku u realnom vremenu. [3]

Većina današnjih telemetrijskih sustava sastoji se od ulaznog uređaja, tj. pretvornika, medija prijenosa, instrumenta za primanje i obradu signala te neke vrste instrumenta za snimanje ili prikaz. Pretvarač preuzima sve što se mjeri ili nadzire te pretvara tu vrijednost u električni impuls. Nakon što je nešto izmjereno i pretvoreno u električni signal, mora se prenijeti do instrumenta koji će obraditi taj signal. Prijenos podataka može biti mehanički, električni ili putem elektromagnetskog zračenja. Najčešće se koristi radiofrekvencijski, infracrveni, ultrazvučni, GSM, satelitski ili kabelski. Međutim, podaci se mogu prenositi i putem žičane komunikacije, kao što je telefonska ili računalna mreža, optička veza i slično. Podaci se uglavnom šalju u digitalnom obliku jer se na taj način postiže najveća vjernost prijenosa podataka, a na velike udaljenosti gotovo bez smetnji i izobličenja. Na taj način podaci dolaze do računala koja ih mogu prikazivati u realnom vremenu i ujedno spremati radi naknadnog proučavanja. [1]



Slika 1. Shema telemetrijskog sustava [8]

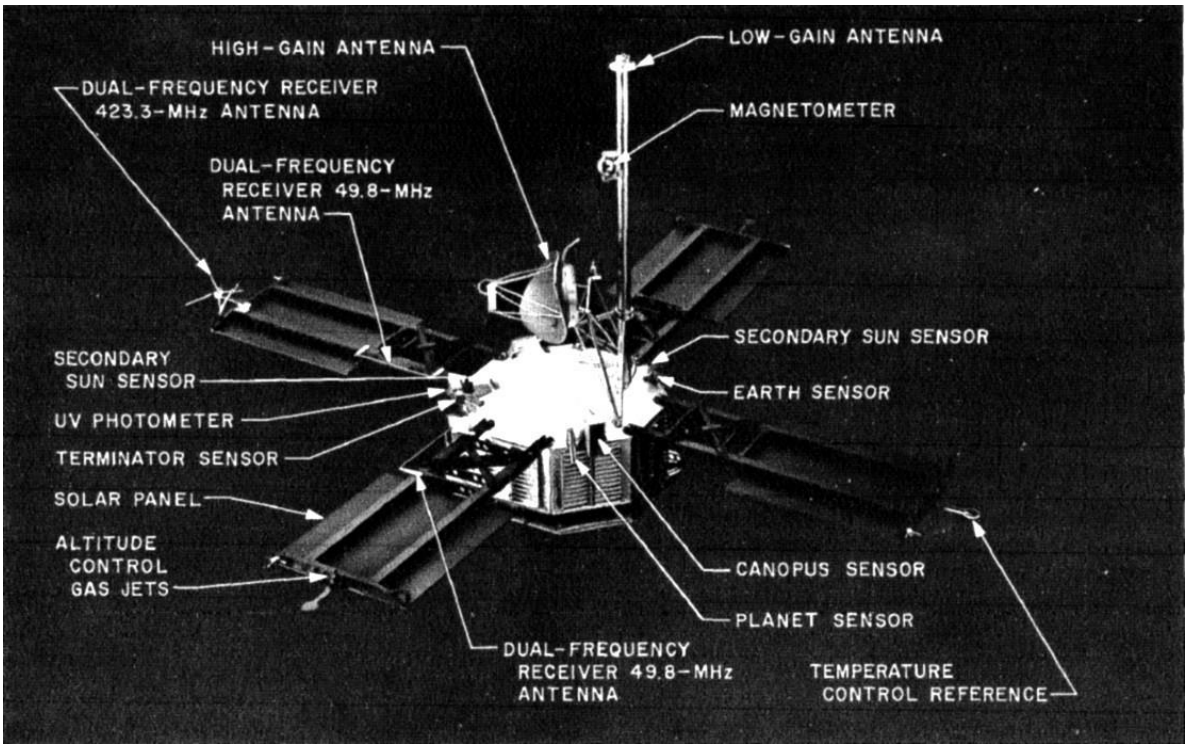


Slika 2. Prikaz telemetrijskog sustava [9]

2. POČETCI TELEMETRIJE

Početci telemetrije putem žice zabilježeni su još u 19. stoljeću. Jedan od prvih krugova za prijenos podataka razvijen je između Zimskog dvorca ruskog cara i vojnog stožera 1845. godine. Na Mont Blancu 1874. godine francuski inženjeri su izgradili sustav vremenskih senzora i senzora za dubinu snijega koji su slali podatke u Pariz u realnom vremenu. Američki izumitelj C. Michalke 1901. godine izumio je selsin, sustav za daljinski sinkroni prijenos kutnoga zakreta. Komplet seizmičkih stanica koje su telemetrirale podatke u opservatorij Pulkovo izgrađen je 1906. godine u Rusiji. Commonwealth Edison razvio je 1912. godine sustav telemetrije za nadgledanje električnih opterećenja na svojoj energetskoj mreži. Panamski kanal, koji je završen u godinama 1913./1914., koristio je opsežne telemetrijske sustave za praćenje brava i vodostaja. [3]

Što se tiče bežične telemetrije, ona se pojavila u prvoj polovici 20. stoljeća. Pojavila se u radionsondi koju su 1930. istovremeno razvili Robert Bureau u Francuskoj i Pavel Molchanov u Rusiji. Konstrukcija Pavela Molchanova postala je popularnija zbog svoje jednostavnosti jer je mjerenja sa senzora pretvarala u Morseov kod. Njemačka raketa V-2 koristila je sustav primitivnih multipleksiranih radio signala nazvan "Messina" za prijavljivanje četiri parametra rakete, ali je taj sustav bio vrlo nepouzdan. U SAD-u i SSSR-u sustav Messina brzo je zamijenjen naprednijim sustavima (u oba slučaja temeljenim na modulaciji položaja impulsa). Rani sovjetski raketni i svemirski telemetrijski sustavi koji su razvijeni u kasnim četrdesetima koristili su ili modulaciju položaja impulsa (npr. Tral telemetrijski sustav kojeg je razvio OKB MEI) ili modulaciju trajanja impulsa (npr. Sustav RTS-5 kojeg je razvio NII- 885). U SAD-u u početku su se koristili sličnim sustavima, ali ih je kasnije zamijenila impulsno kodna modulacija (PCM) (na primjer u sondi namijenjenoj istraživanju Marsa, Mariner 4 [Slika 3.]). Kasnije sovjetske interplanetarne sonde koristile su redundantne radio sustave, prenoseći telemetriju PCM-om na decimetarski opseg, a PPM na centimetarskom opsegu. [3]



Slika 3. Mariner 4 [10]

3. VAŽNOST I PRIMJENA TELEMETRIJE

Telemetrija je danas iznimno važna u širokom rasponu područja primjene jer omogućava praćenje trenutnog stanja stvari i omogućuje praćenje na udaljenim područjima i područjima koja nisu lako dostupna čovjeku. Odnosno, omogućuje čovjeku da zna što se događa u svakom trenutku bez da je fizički prisutan na mjestu promatranja. Širok raspon područja primjene telemetrije uključuje meteorologiju, uljnu i naftnu industriju, automobilsku industriju te utrke vozila, transport, poljoprivredu, istraživanje svemira (raketnu tehniku, ispitivanje leta, astronautiku), vojnu tehnologiju, raspodjelu resursa, medicinu, rudarstvo, praćenje divljih životinja, maloprodaju, provedbu zakona, razvoj softvera i još mnoga druga područja. Može se primijetiti da se telemetrija pojavljuje u gotovo svim aspektima ljudske aktivnosti, gotovo da nema područja gdje se ne bi mogla primijeniti. U nastavku će biti opisana primjena telemetrije u nekoliko odabranih područja.

3.1. Automobilska industrija i utrke vozila

U automobilskoj industriji telemetrija se koristi prilikom testiranja vožnje vozila, prate se parametri rada motora i ostalih komponenti vozila dok se vozilo kreće po stazi. Ona je danas ključna u utrkama, omogućava inženjerima utrke tumačenje podataka prikupljenih tijekom test vožnje ili za vrijeme prave utrke kako bi pravilno podesili vozilo za optimalne performanse. Najpoznatija primjena je u utrkama Formule 1 gdje su sustavi toliko napredni da se može izračunati potencijalno potrebno vrijeme automobila za jedan krug što se onda očekuje od vozača da ostvari. Ono što se mjeri i prati kod ovih vozila je ubrzanje u tri osi, temperatura, brzina kotača i pomaci ovjesa. U Formuli 1 jedno kratko vrijeme koristila se i dvosmjerna telemetrija. To je telemetrija koja omogućuje dvosmjernu komunikaciju, odnosno mogućnost izmjene parametara rada iz glavnog stožera, što znači manipuliranje krajnjom brzinom vozila, utjecaj na broj okretaja i slično. Upravo iz tog razloga ovakva telemetrija brzo je ukinuta. Osim Formule 1, telemetrija se primjenjuje i u utrkama jedrilica te RC automobila. [4]

Slika 4. Telemetrija trkaćeg automobila [11]

3.2. Poljoprivreda

U poljoprivredi većina aktivnosti vezanih uz zdrave usjeve i dobre prinose ovisi o pravovremenoj dostupnosti podataka o vremenu i tlu. Stoga tu veliku ulogu imaju bežične meteorološke stanice. One omogućavaju prevenciju bolesti i precizno navodnjavanje te šalju baznoj stanici informacije potrebne za odlučivanje oko postupanja s usjevima, to su informacije o temperaturi zraka, relativnoj vlažnosti zraka, oborinama, vlažnosti lišća, sunčevom zračenju, brzini vjetra, manjku vode i sl. S obzirom da se lokalna mikroklima može poprilično razlikovati, ovakve informacije trebaju dolaziti direktno iz usjeva i zato se koristi telemetrija. [3]

Slika 5. Telemetrija u poljoprivredi [12]

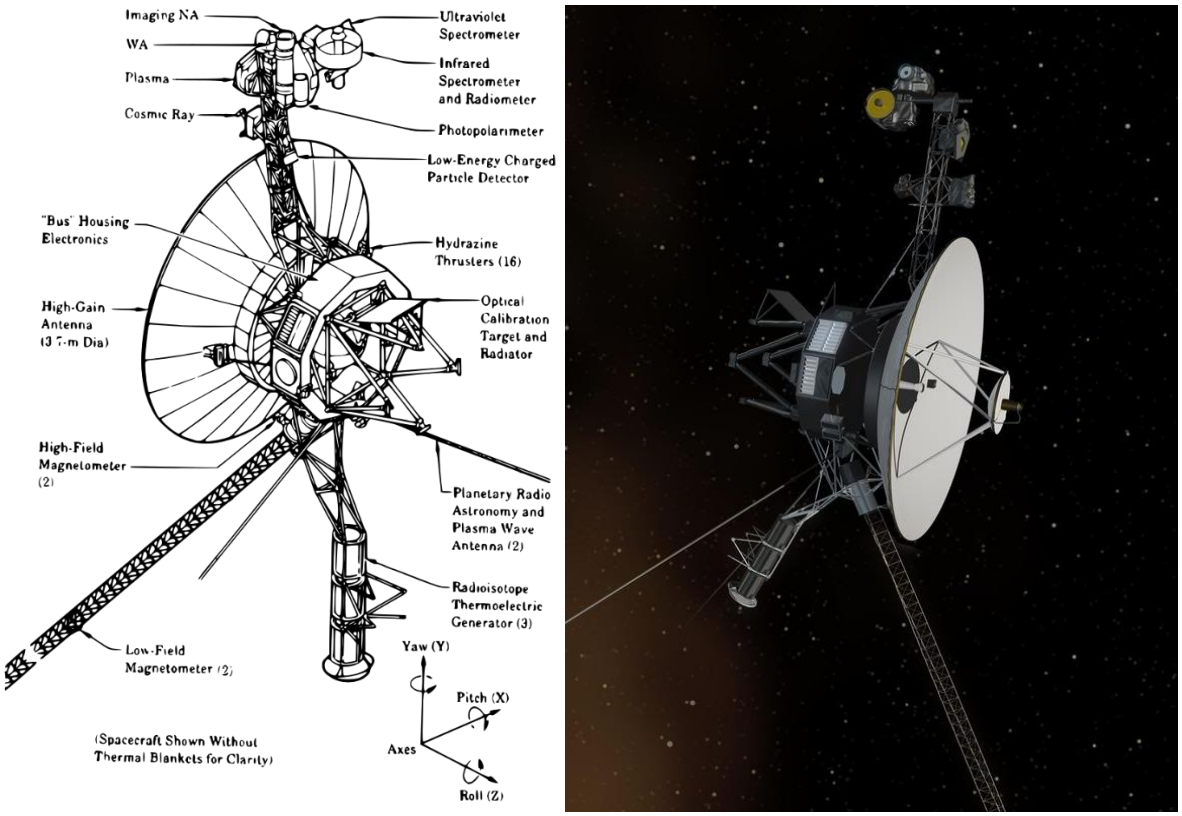
3.3. Istraživanje svemira

Svemirske agencije koriste telemetrijske i/ili telekomunikacijske sustave za prikupljanje podataka sa svemirskih letjelica i satelita. Telemetrija je od vitalne važnosti za razvoj projektila, satelita i zrakoplova jer se sustavi često unište za vrijeme ili nakon ispitivanja, a inženjerima su potrebni kritični parametri sustava kako bi analizirali i stoga poboljšali performanse sustava. Danas gotovo svaka vrsta zrakoplova, rakete ili svemirskih letjelica ima bežični telemetrijski sustav dok se provodi testiranje. Zrakoplovna mobilna telemetrija koristi se za sigurnost pilota i osoba na zemlji tijekom ispitivanja leta budući da je osnovni izvor podataka o mjerenju i stanju letjelice u stvarnom vremenu.

Kod svemirskih letjelica telemetrija je važna zbog velikih udaljenosti na kojima se one nalaze od Zemlje. Neke letjelice nastavljaju putovati duboko u svemir iako je dogovorena misija završena. Primjer za to je Voyager 1 [Slika 6.] koji je sada udaljen više od 15 milijardi kilometara od Zemlje. Prvi put je lansiran 1977. godine, a njegovi telemetrijski sustavi još uvijek funkcioniraju, radio signali putuju brzinom svjetlosti i sada im je potrebno više od 16 sati da stignu do Zemlje. [5] Još jedan primjer svemirske letjelice koja se služi telemetrijom je letjelica Mars Helicopter Scout (MHS) čije se polijetanje očekuje 2021. godine. MHS je robotski helikopter koji će testirati tehnologiju potrebnu za istraživanje terena na Marsu. Ovo

je test vozilo koje će pomoći znanstvenicima da u budućnosti izrađuju kvalitetnije letjelice za buduće misije u svemir. MHS ima postavljene senzore i kamere pomoću kojih dobiva kontinuiranu procjenu letjelice u stvarnom vremenu i na temelju nje se podešavaju parametri letjelice, a svi podaci koje prikupi, iskoristiti će se u daljnjim razvijanjima svemirskih letjelica. [6] Svemirska telemetrija također se koristi tijekom svemirskih misija kada se događaji izdogađaju prebrzo da bi ljudi ulovili sve informacije na vrijeme i dostavili ih kontroli misije, stoga te kritične informacije sa senzora prikupljaju računala koja ih onda šalju na Zemlju u realnom vremenu. To je vrlo važno jer nijedan čovjek ili skupina ljudi ne može tom brzinom proći kroz sve podatke kao što to može računalo, ono će prije uočiti određenu anomaliju i izdati upozorenje.

U raketnoj tehnici telemetrijska oprema je sastavni dio sredstava koja se koriste za nadgledanje položaja i stanja lansirnog vozila. Problemi koji se tu javljaju su ekstremno okruženje (temperatura, ubrzanje, vibracije), opskrba energijom, poravnavanje antene te vrijeme putovanja signala (na velikim udaljenostima, iz svemira). [3]

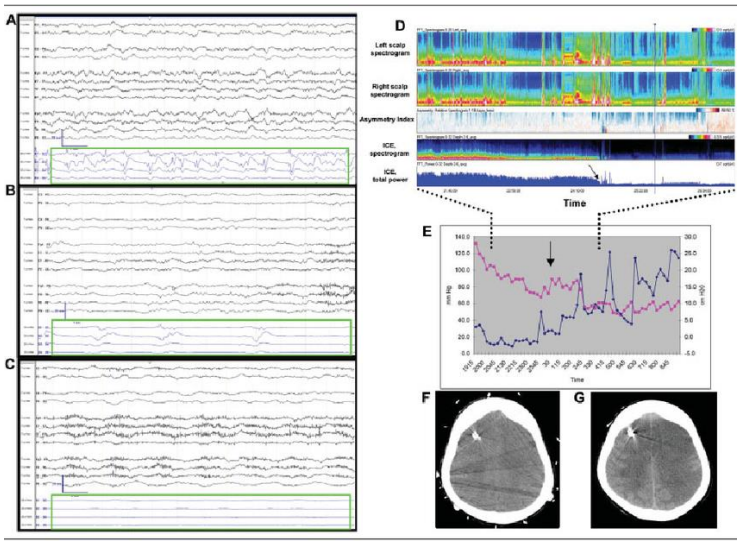


Slika 6. NASA-ina svemirska sonda Voyager 1 [5] [13]

3.4. Medicina

Biotelemetrija ili medicinska telemetrija koristi se kod pacijenata koji su u riziku od abnormalne srčane aktivnosti, uglavnom u odjelu za koronarnu skrb. Specijalisti za telemetriju često nadgledaju u bolnicama takve pacijente koji su opremljeni mjernim uređajem te uređajima za snimanje i slanje podataka. U uređajima se nalazi alarmantna funkcija koja upozorava kada se pacijent nalazi u akutnom ili opasnom stanju. Dnevnik podataka može biti koristan za dijagnozu pacijentovog stanja. Biotelemetrija se također koristi za praćenje reakcije na antiaritmičke lijekove. [3]

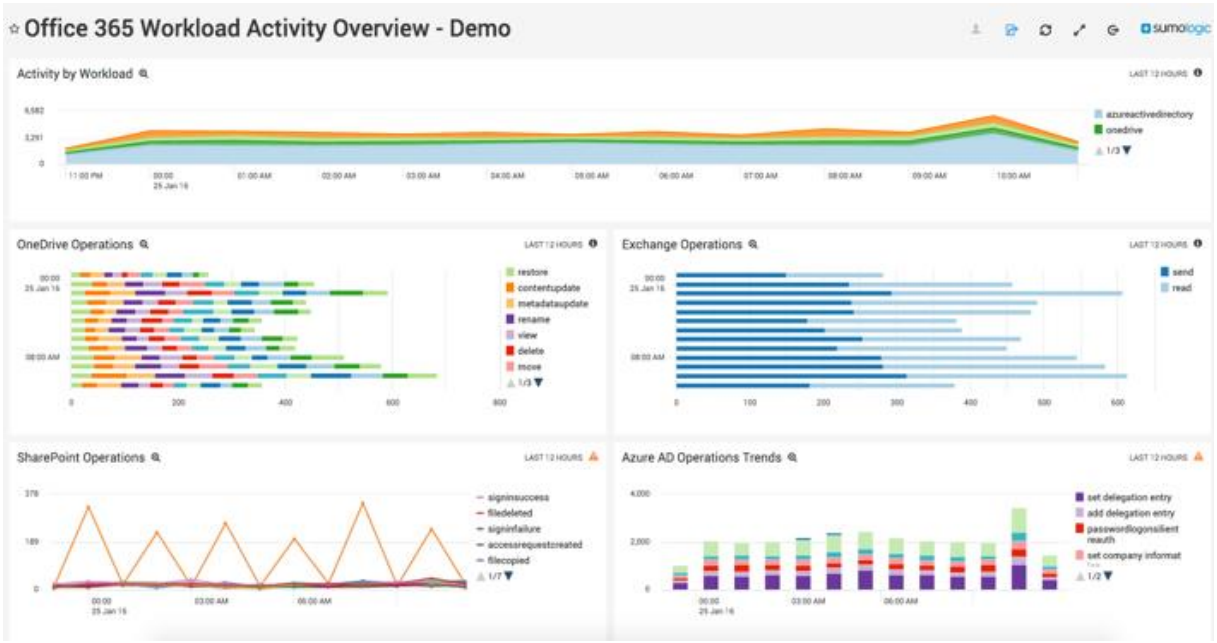
Telemetrija se počinje sve više koristiti i u području neurofiziologije, tzv. neurotelemetrija. Neurofiziologija proučava središnji i periferni živčani sustav putem snimanja bioelektrične aktivnosti, bilo spontane ili potaknute. U neurotelemetriji (NT) registrirani EEG tehnolog daljinski nadgleda elektroencefalogram (EEG) pacijenta koristeći napredni komunikacijski softver. NT omogućava neposredno praćenje aktivnosti moždanih valova i prepoznavanje pogoršavanja pacijentovog stanja prije pojave fizičkih znakova i simptoma. Sve to pomaže liječniku u bržem postavljanju dijagnoze i odlučivanju o načinu liječenja. NT se koristi kod praćenja epilepsije pacijenata, u neuro i pedijatrijskim jedinicama intenzivnog liječenja te jedinici intenzivnog liječenja za novorođenčad. Zbog radno-intenzivne prirode kontinuiranog praćenja EEG-a, NT se obično izvodi u većim akademskim bolnicama koristeći interne programe koji uključuju EEG tehnologe, osoblje za informatičku podršku, neurologa, neurofiziologa i osoblje za nadgledanje. Neurotelemetrija i kontinuirano praćenje EEG-a pružaju dinamične informacije o funkciji mozga koje omogućuju rano otkrivanje promjena u neurološkom statusu, što je posebno korisno kada je klinički pregled ograničen. [7]



Slika 7. Elektroencefalogram (EEG) [14]

3.5. Razvoj softvera

U razvoju softvera telemetrija se koristi za prikupljanje podataka o korištenju i performansama aplikacija i komponenti aplikacija, npr. podataka o tome koliko često se koriste određene značajke, o mjerenju vremena pokretanja i vremena obrade, o hardveru, padu sustava i općoj statistici uporabe i/ili ponašanju korisnika. Ponekad se prate i detaljniji podaci poput metrike pojedinačnog prozora, broja korištenih značajki i pojedinačnog vremena korištenja. Ova vrsta telemetrije je od ključne važnosti za softverske inženjere zato što dobivaju podatke iz širokog spektra točaka koje je gotovo nemoguće sve testirati interno te također dobivaju podatke o popularnosti određenih značajki i o tome treba li im dati prednost ili razmatrati njihovo uklanjanje. Međutim, kod telemetrije u softverima dolazi do problema privatnosti budući da se ona može lako koristiti za profiliranje korisnika. Stoga se od korisnika traži odobrenje za korištenje ove funkcije u obliku ugrađene značajke ili tijekom postupka instalacije softvera. [3]



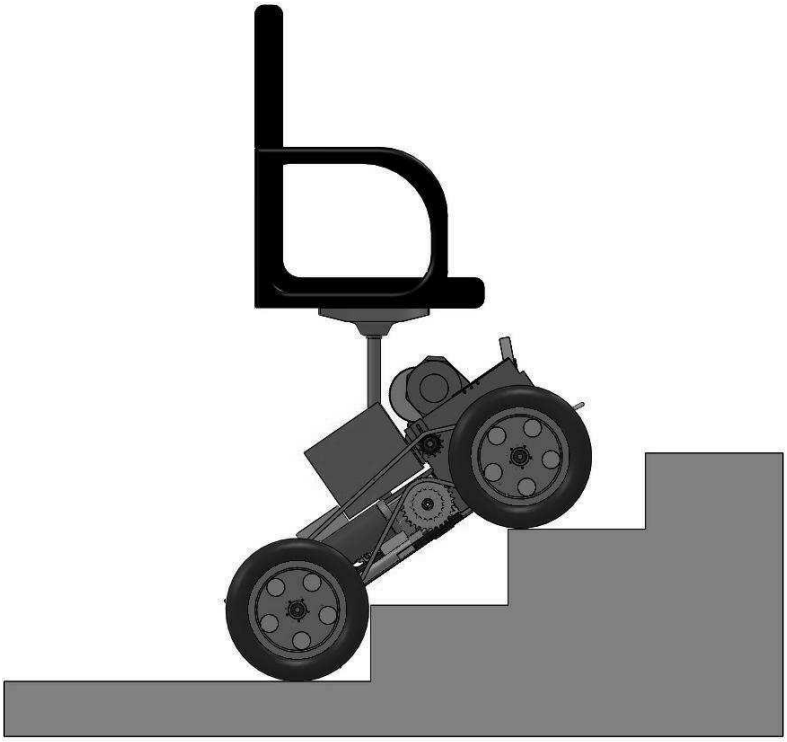
Slika 8. Praćenje rada aplikacije [15]

Može se zaključiti da je telemetrija danas od iznimne važnosti u gotovo svim područjima, primjenjiva je na raznorazne načine te olakšava i pojednostavljuje određene aktivnosti. Neke aktivnosti čak bi bilo nemoguće provesti bez nje, omogućava pristup nedostupnim i udaljenim mjestima, prima sve informacije koje bi čovjeku mogle promaknuti, i sl.

4. KOPACK INVALIDSKO VOZILO

U ovom radu napravljena je telemetrija invalidskog vozila tvrtke KOPACK. KOPACK invalidsko vozilo pripada kategoriji statičkog stabilnog vozila koje osigurava visok stupanj sigurnosti u slučaju kvara ili nestanka struje. Vozilo ima četiri kotača koja pokreću četiri istosmjerna motora, također sadrži pomoćni kotač koji olakšava vožnju u zatvorenim prostorima i smanjuje silu potrebnu za rotaciju. Korisnik upravlja vozilom putem komandne palice (*joystick-a*), a na konzoli vozila mogu se odabrati način i parametri vožnje. Težina praznog vozila iznosi oko 80 kg. Vozilo sadrži dva para penjalica s prednje i stražnje strane, a te penjalice su glavni element vozila jer omogućuju penjanje ili spuštanje po stepenicama. Operacija penjanja, odnosno spuštanja, dijelom se izvodi automatski, a dijelom kontrolom korisnika i za vrijeme cijele operacije sjedalo ostaje u uspravnom položaju. Najveći nagib koji vozilo može prevladati iznosi 37°.

Vozilo je povezano bežično (putem bluetootha) s osobnim računalom. S nadzornim programom moguće je nadzirati i snimati kinematiku vozila i ostale parametre vožnje. Također, vozilom se može upravljati s udaljenosti, bez da se osoba fizički nalazi u sjedalu vozila. Vožnja je vrlo intuitivna i može se brzo savladati.



Slika 9. Bočni prikaz KOPACK vozila

4.1. Opis mehaničkog sustava

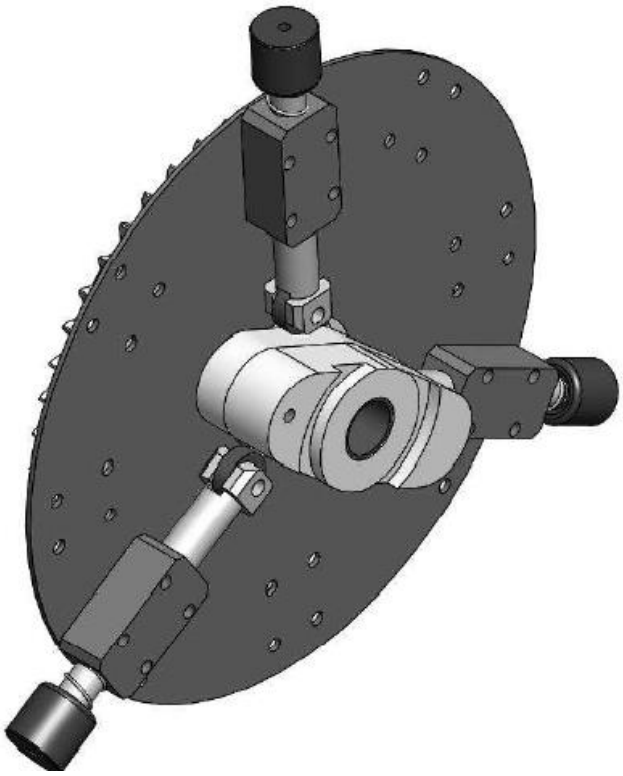
Vozilo ima dva pogonska kotača sa svake strane. Svaki kotač pokreće istosmjerni motor. Promjer kotača je 300 mm, udaljenost između prednjeg i stražnjeg kotača iznosi 530 mm, a udaljenost između lijevog i desnog 680 mm. Veći međuosovinski razmak kotača pridonosi većoj stabilnosti vozila na stepenicama, ali može biti problem s „nasukavanjem“ središnjeg dijela vozila kada penjanje završava, odnosno spuštanje počinje. Stoga bi manji međuosovinski razmak spriječio nasukavanje, ali bi uvelike smanjio stabilnost vozila na stepenicama. Na stražnjoj strani vozila nalaze se i pomoćni kotači koji se mogu spustiti i podići po potrebi, njihova osnovna funkcija je smanjenje potrebne sile tijekom vožnje, pogotovo tijekom skretanja.

Između prednjih i stražnjih kotača nalaze se penjalice koje omogućuju vozilu da fizički prevlada stepenice. Na svojim krajevima one imaju kotačiće i mogu se izvlačiti i uvlačiti ovisno o potrebi. One olakšavaju penjanje i spuštanje vozila po stepenicama. Izvučene su i pružaju potporu vozilu kada se ono nalazi na stepenicama, u suprotnom su uvučene kako ne bi oštetile podlogu. Prednje i stražnje penjalice imaju zasebne motore.

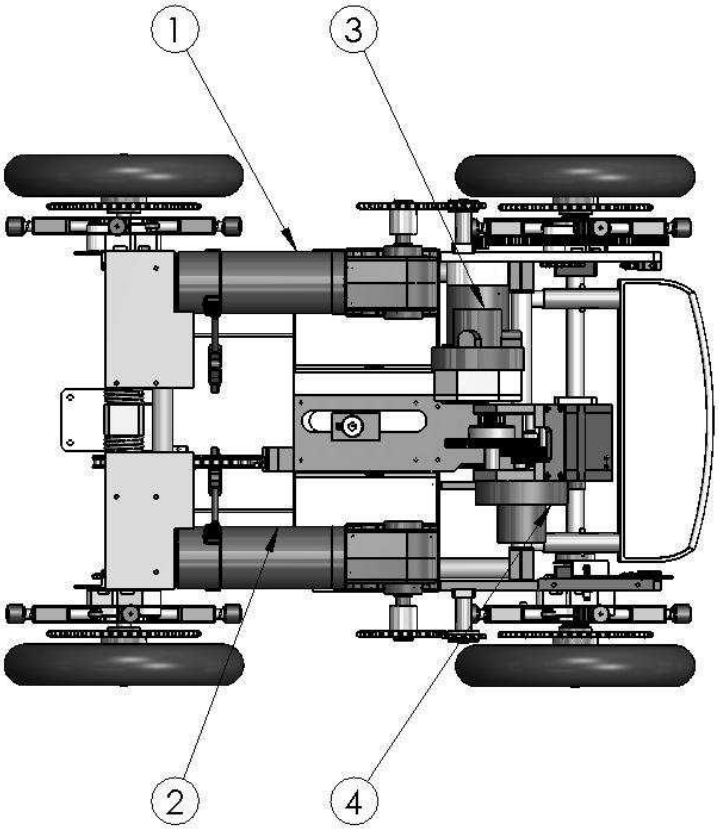
Tijekom spuštanja ili penjanja sjedalo uvijek mora biti u uspravnom položaju, stoga je dodan još jedan istosmjerni motor. Raspon pomicanja stolice određuje maksimalan nagib koji vozilo može nadvladati, kao što je već gore spomenuto, to je nagib od 37°.

Slika 10. Trenutni izgled KOPACK invalidskog vozila

Prijašnji model poprilično se razlikovao od sadašnjeg. Taj model imao je četiri kotača koja su bila pokretana s dva istosmjerna motora, jedan za lijevu stranu vozila, drugi za desnu. Ključni element ovog modela, koji je omogućavao penjanje ili spuštanje po stepenicama, bio je tzv. penjač (eng. *climber*). Penjač je zakrivljeni mehanizam pokretan dodatnim motorom koji pomiče njegove poluge [Slika 11.]. Na slici 12. prikazan je poprečni presjek prijašnjeg modela odozgo. Oznaka s brojem 1 označava lijevi pogon, broj 2 je desni, broj 3 je motor koji pomiče stolicu, a broj 4 je penjač.



Slika 11. Mehanizam penjača



Slika 12. Poprečni presjek starijeg modela KOPACK vozila odozgo

4.2. Opis upravljačkog sustava

Upravljački sustav vozila podijeljen je u dva dijela. Glavno računalo nalazi se na vozilu i upravlja cijelim vozilom do ugrađenog hardvera. Na ovom računalu putem serijske linije spojena je terminalna jedinica koja vozilu šalje naredbe. Terminal može biti bilo koje računalo koje može komunicirati preko te serijske linije i ima ugrađen protokol komunikacije, s tim da ne mora poznavati hardver vozila. Za središnje računalo vozila odabran je mikrokontroler Atmel AT89C51RE2 s dvije serijske komunikacije. Jedna komunikacija koristi se za komunikaciju s terminalnom jedinicom, a druga je za pomoćni procesor. Pomoćni procesor je jednostavan AT89C4051 jer središnje računalo nema dovoljno ulazno/izlaznih linija. Pomoćna jedinica obrađuje signale iz enkodera i digitalnih senzora. Motorima se upravlja pomoću modulacije širine impulsa (PWM) i preko linije koja određuje smjer vrtnje. Svaki motor ima kočnicu kojoj je potrebno oko 0,3 sekunde za promjenu stanja (uključeno/isključeno). Ako je vozilo zaustavljeno duže od 5 sekundi, motori kotača automatski se koče. Motori kotača imaju inkrementalni enkoder s 2000 impulsa po okretaju i koriste se za mjerenje brzine kotača. Brzinu kotača kontrolira modificirani PI regulator svakih 100 ms. Terminalna jedinica postavlja translacijsku i rotacijsku brzinu vozila, a zatim se izračunavaju potrebne brzine lijevog i desnog kotača prema kinematskom modelu vozila. Položaj penjalica također se mjeri pomoću inkrementalnog enkodera, a njihov referentni položaj određuje se induktivnim senzorom.

Nagib sjedala unaprijed je postavljen i aktivira se automatski kada je na terminalnom uređaju odabran način rada koji nije penjanje ili spuštanje. Razlozi za održavanje sjedala u uspravnom položaju prilikom spuštanja ili penjanja su udobnost korisnika i stabilnost vozila, koja je naravno puno važnija jer sprječava prevrtanje vozila. Položaj sjedala kontinuirano se mjeri potenciometrom, a određeni položaj provjerava se induktivnim senzorom. Za kontrolu nagiba stolice također je potrebno izmjeriti apsolutni kut nagiba vozila od vodoravne ravnine, što se izvodi s inklinometrom SCA61T. No, budući da je senzor za nagib u stvari pretvoren akcelerometar za mjerenje nagiba, mjeri također i dinamičku komponentu gibanja vozila. Stoga se izmjereni signal mora filtrirati kako bi se dobile korisne informacije za regulator nagiba stolice.

Prije nego što se vozilo počne penjati, važno je da prednji dio vozila ne bude na manje od dvadeset centimetara od stepenica. Odabire se penjanje kao način rada i sjedalo se namješta automatski blago prema naprijed radi veće stabilnosti, a prednje penjalice zauzimaju početni položaj za penjanje. Cijelo vrijeme dok je vozilo u ovom načinu rada, kotači lagano povlače vozilo prema naprijed konstantnom silom koja iznosi otprilike 15% od maksimalne (upravljanje

brzinom je isključeno) i vožnja unatrag je onemogućena. Kad korisnik utvrdi da prednji kotači dodiruju sljedeću stepenicu (u tom trenutku penjalice trebaju biti iznad te stepenice), može pritisnuti gumb (strelica naprijed) kako bi zatražio penjanje na nju. Tada se kotačići na penjalicama počinju okretati prema zadanoj brzini (što se zadaje u terminalnoj jedinici) skupa s kotačima. Okretni moment na kotačima ovisi o stupnju prevladavanja stepenica, a određuje se kutom penjalica. U početku je okretni moment mali, ali kada se penjalice dovoljno podignu, okretni moment kotača se poveća, a smanjuje se ponovno kada je vozilo u zadnjoj fazi prevladavanja stepenice kako se ne bi velikom brzinom zaletilo u iduću. Kada upravljački sustav primi informaciju da je vozilo u vodoravnom položaju, kotači se ne vuku naprijed i nema straha od nekontroliranog gibanja.

Slično se događa i kada se vozilo spušta niz stepenice, bitna razlika je što su kod spuštanja kretanje vozila i gravitacija u istom smjeru pa je proces manje stabilan. Kada vozilo dođe do ruba, mora se pomaknuti malo naprijed kako bi se oslonilo na penjalice. Cijelo vrijeme tijekom spuštanja kotači vozila polako se kreću prema natrag tako da osoba u vozilu ima osjećaj da je vozilo „zalijepljeno“ za stepenice. Kada se želi spustiti na iduću stepenicu, osjetljivost komandne palice uvelike se smanjuje i ne dopušta podešavanje veće brzine.

Upravljačka konzola [slika 13.] sastoji se od komandne palice, četiri tipke i LCD-a s četiri reda po dvadeset znakova, što omogućava komunikaciju s korisnikom. Četiri tipke su MENU, ENTER, strelica gore i strelica dolje. Način vožnje i neki parametri vožnje odabiru se u izborniku pritiskom na tipku MENU, svi parametri zadržavaju se čak i nakon isključivanja napajanja.

Slika 13. Upravljačka konzola

4.3. Komunikacija KOPACK vozila

Komunikacija KONZOLA ← → SLAVES

1. Komunikacija prema motor kontroleru odvija se preko RS485 komunikacije na 57.600 bps, u binarnom modu po pravilu drugog komplementa. Naredba koju motor kontroler prima uvijek ima 6 bajtova i ima sljedeće značenje:

   } f v i p C

gdje je:

} - znak za početak naredbe koju šalje master (kod 125)

f - bit 0-3 adresa A=0-15

bit 4 otkoči(0)/zakoči(1) motor

bit 5 reg. po brzini(0)/ reg. po položaju(1)

bit 6 krutost regulacije (za kotače pri penjanju)

v - brzina (-100 % .. 100 %) za reg. po brzini ili max. brzina gibanja (0 % .. 100 %) za reg. po položaju

i - max. struja (moment), od 0 % .. 100 % tj. apsolutni iznos

p - pozicija, od -120° .. 120°

C - Check Sum, vrijedi da je f + v + i + p + C = 0

Ako je zadana naredba da se motor zakoči (f – bit 4) ostali dio naredbe se ignorira. Ako motor kontroler ne primi nikavu naredbu 1 sekundu – zaustavlja motor. Vrijeme prijenosa je 1.1 ms.

Adrese slave uređaja su sljedeće:

Tablica 1. Adrese pojedinih motora

MOTOR	ADRESA
Prednji lijevi kotač	1
Prednji desni kotač	2
Stražnji lijevi kotač	3
Stražnji desni kotač	4
Skretanje kotača	5
Stolica	6
Prednja penjalica	7
Stražnja penjalica	8
Inklinometar	14

Dakle, moguće su dvije vrste naredbi:

I. } 0A v i 0 C Regulacija po brzini uz ograničenje struje (držanje momenta)

II. } 1A v i p C Regulacija po položaju uz ograničenje brzine i struje (samo za motore penjalice i stolice)

2. Odmah po primitku naredbe svaki motor kontroler vraća podatke u 6 bajtova sljedećeg značenja:

   { f v i p C

gdje je:

{ - znak za početak niza koji vraća slave (kod 123)

f - status motor kontrolera

  bit 0-3 adresa A=0-15

  bit 4 motor otkočen(0)/zakočen(1)

  bit 5 stanje senzora

  bit 6 struja motora u limitu (1) ili nije (0)

v - izmjerena brzina gibanja, od -100 % .. 100 %

i - izmjerena struja (moment), od -100 % .. 100 %

p - izmjerena pozicija, od -120° .. 120° (za kotače 0),

C - Check Sum. Vrijedi da je f + v + i + p + C = 0

3. Za inklinometar vrijedi:

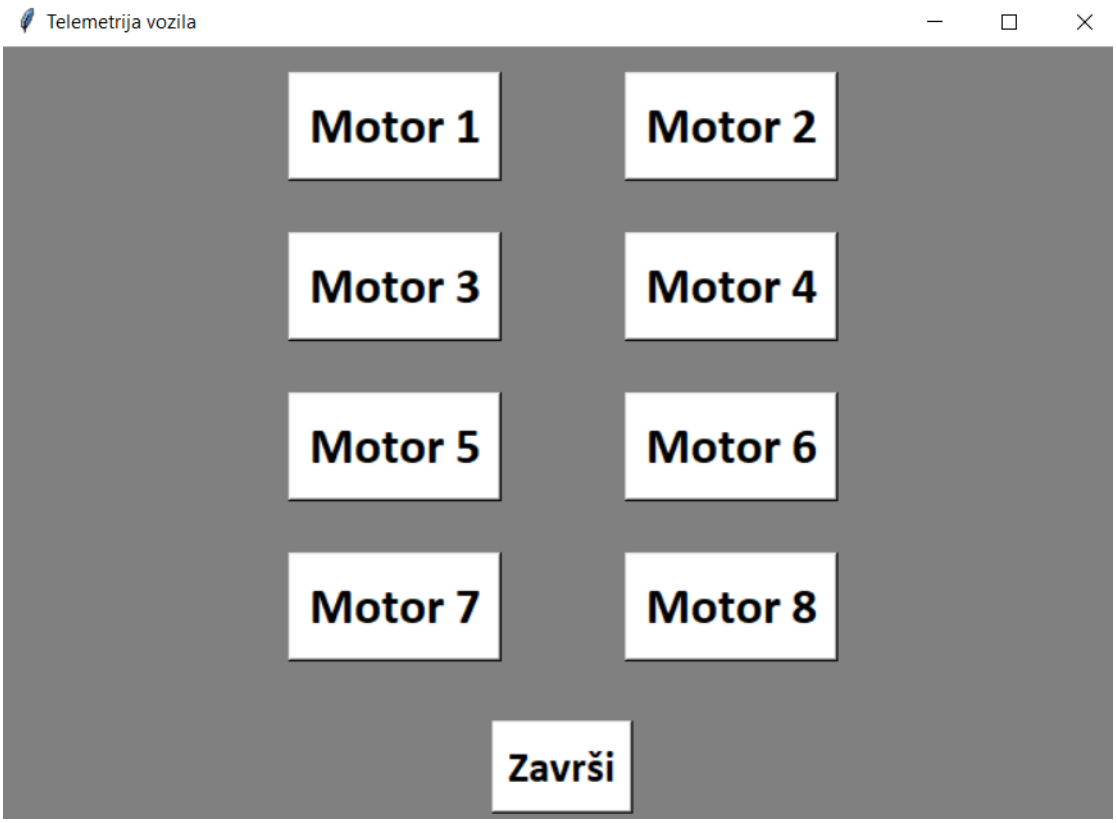
   } 0E 00 00 00 F2

   { 0E 00 Lo Hi CS

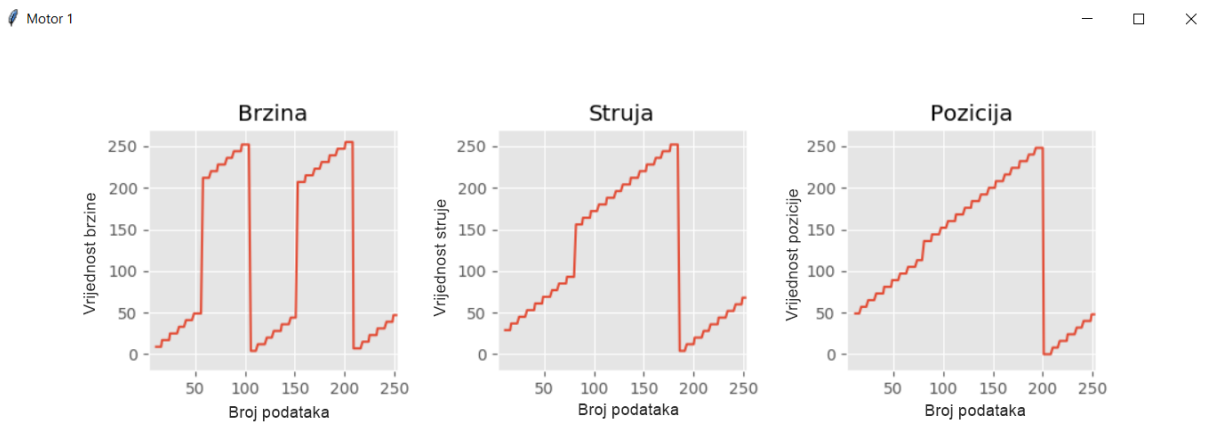
   pitch= 256*Hi + Lo (° x 100) > 0 nagib naprijed

5. PROGRAM

Alat odabran za izradu sučelja i prikupljanja podataka je programski jezik Python. Pomoću paketa Tkinter, koji se nalazi unutar Pythona, napravljeno je grafičko korisničko sučelje (GUI, eng. g*raphical user interface*). Grafičko sučelje sastoji se od glavnog prozora u kojem se nalazi izbornik s osam motora [Slika 14.]. Izborom određenog motora pritiskom na tipku otvara se prozor s grafičkim prikazima odabranih elemenata procesa, tj. brzine, struje i pozicije kako je prikazano na Slici 15., analogno je za sve motore. Na apscisi se nalazi broj podataka preko kojeg se može pratiti stvarno vrijeme, u jednoj sekundi prima se deset podataka.



Slika 14. Glavni izbornik



Slika 15. Primjer rada programa: simulacija brzine, struje i pozicije

5.1. Objašnjenje programskog koda

Prije samog pisanja koda potrebno je učitati sve potrebne pakete, a to su paketi koji će omogućiti stvaranje grafičkog korisničkog sučelja, ostvarenje serijske komunikacije, istovremeni rad navedenog te crtanje grafova u stvarnom vremenu. Učitavanje paketa vrši se pomoću naredbe *import*. [Slika 16.]

```
from tkinter import *   #grafičko sučelje

import serial   #serijska komunikacija
import threading    #istovremeni rad komunikacije i sučelja

#paketi za crtanje grafova
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,NavigationToolbar2Tk
matplotlib.use('TkAgg')
from matplotlib.figure import Figure
```

Slika 16. Potrebni programski paketi

Nakon učitavanja paketa, idući korak je uspostavljanje serijske komunikacije [Slika 17.]. Pokretanjem programskog koda, prvo što se događa je otvaranje ulaza za serijsku komunikaciju (prva linija koda na slici 17.). Nakon što se otvori ulaz za serijsku komunikaciju, potrebno je definirati što će se događati s primljenim podacima. Definira se funkcija *SERIAL* unutar koje se na početku stvara tekstualna datoteka u koju će se kasnije spremati primljeni podaci, a koja se na početku prazni kako bi se svakim pokretanjem programskog koda spremali podaci za tu

sesiju. Dalje se definiraju globalne varijable koje će se kasnije koristiti u funkciji izvan ove. Potom se stvara *while* petlja koja kaže da se događa sljedeće dok god je serijski ulaz otvoren, a sljedeće se odnosi na čitanje podataka u dekodiranom obliku, gdje prvi podatak predstavlja znak „{“ koji označava početak niza koji se šalje, drugi podatak je informacija o motoru, treći je vrijednost brzine, četvrti struje, peti pozicije i konačno šesti je *Check sum*, odnosno suma prethodne četiri veličine (f+v+i+p). Ovih šest podataka stavlja se u listu i na taj način sprema u tekstualnu datoteku. Unutar ove funkcije postoje dva brojača, to su *i* i *k*, *i* služi za brojanje ukupnog broja primljenih nizova podataka, a *k* broji motore, odnosno služi za razvrstavanje podataka ovisno o motoru kojem ti podaci pripadaju.

Po završetku definiranja funkcije stvara se tzv. *threading* izvan nje kojim se omogućava rad u sučelju dok se u pozadini cijelo vrijeme primaju podaci jer se podaci i primaju dok god je ulaz otvoren, a to je i potrebno s obzirom da treba dobiti grafove koji teku u realnom vremenu. [Slika 18.]

Zatim se započinje s izradom sučelja, prvo se stvara izgled glavnog prozora u koji će se smjestiti izbornik motora. Stvara se klasa koja nosi naziv *Telemetrija* i u njoj se na početku definira devet tipki, osam tipki pripada motorima, a deveta tipka služi za izlazak iz programa, nosi naziv „Završi“. [Slika 18.]

```
ser=serial.Serial('COM3', 57600)  #otvaranje ulaza serijske komunikacije

def SERIAL():
    f1=open('text.txt','r+')
    f1.truncate(0)              #pražnjenje tekstualne datoteke na početku rada
    global Data,i,M1,M2,M3,M4,M5,M6,M7,M8
    i=0
    k=1

    while 1:
        Data1=ser.read().decode('utf-8') #'{'
        Data2=ord(ser.read())            #f
        Data3=ord(ser.read())            #v-brzina
        Data4=ord(ser.read())            #i-struja
        Data5=ord(ser.read())            #p-pozicija
        Data6=ord(ser.read())            #CS-suma
        Data = [Data1,Data2,Data3,Data4,Data5,Data6]
        print (Data)
        i+=1

        #petlja koja služi razvrstavanju podataka
        if k==1:
            M1=Data
            k+=1
        elif k==2:
            M2=Data
            k+=1
        elif k==3:
            M3=Data
            k+=1
        elif k==4:
            M4=Data
            k+=1
        elif k==5:
            M5=Data
            k+=1
        elif k==6:
            M6=Data
            k+=1
        elif k==7:
            M7=Data
            k+=1
        elif k==8:
            M8=Data
            k=1

        with open('text.txt','a') as f2:        #tekstualna datoteka u koju se spremaju podaci koji se šalju serijskom komunikacijom
            f2.write('%s \n' %str(Data[1:6]))
```

Slika 17. Definiranje funkcije SERIAL

```
t1=threading.Thread(target=SERIAL,daemon=True)  #threading omogućuje rad u sučelju dok se u pozadini konstantno primaju podaci serijskom komunikacijom
t1.start()


#izgled glavnog prozora
glavniProzor=Tk()
glavniProzor.geometry('700x500')
glavniProzor.configure(bg='gray')
glavniProzor.title('Telemetrija vozila')

class Telemetrija():
    def __init__(self):
        #stvaranje gumba
        gumb1=Button(text='Motor 1',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_1)
        gumb1.place(relx=0.35, rely=0.1, anchor=CENTER)
        gumb2=Button(text='Motor 2',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_2)
        gumb2.place(relx=0.65, rely=0.1, anchor=CENTER)
        gumb3=Button(text='Motor 3',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_3)
        gumb3.place(relx=0.35, rely=0.3, anchor=CENTER)
        gumb4=Button(text='Motor 4',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_4)
        gumb4.place(relx=0.65, rely=0.3, anchor=CENTER)
        gumb3=Button(text='Motor 5',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_5)
        gumb3.place(relx=0.35, rely=0.5, anchor=CENTER)
        gumb3=Button(text='Motor 6',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_6)
        gumb3.place(relx=0.65, rely=0.5, anchor=CENTER)
        gumb3=Button(text='Motor 7',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_7)
        gumb3.place(relx=0.35, rely=0.7, anchor=CENTER)
        gumb3=Button(text='Motor 8',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_8)
        gumb3.place(relx=0.65, rely=0.7, anchor=CENTER)

        gumb=Button(text='Završi',fg='black', bg='white',font=('Calibri', 20, 'bold'),command=self.iskljucenje)
        gumb.place(relx=0.5,rely=0.9,anchor=CENTER)
```

Slika 18. Threading i stvaranje klase Telemetrija

Pritiskom na tipke motora pozivaju se funkcije koje su dalje definirane. U nastavku će biti objašnjena funkcija za motor 1, a funkcije za sve ostale motore analogne su ovoj. Unutar

funkcije *motor_1* kreira se prozor u koji će se smjestiti sve definirano ovom funkcijom, zatim se kreira okvir za grafove te se stvaraju prazne liste u koje će se pospremati podaci koji će se crtati na grafovima. [Slika 19.]

```
def motor_1(self):
    #kreiranje prozora za prvi motor
    self.prozor_1=Tk()
    self.prozor_1.title('Motor 1')
    self.prozor_1.geometry('1100x700')
    self.prozor_1.resizable()
    self.prozor_1.config(bg='white')

    #postavljanje okvira za grafove
    style.use('ggplot')
    fig=plt.Figure(figsize=(12, 8))
    ax1=fig.add_subplot(2,3,1)
    ax2=fig.add_subplot(2,3,2)
    ax3=fig.add_subplot(2,3,3)
    fig.subplots_adjust(bottom=0.2,wspace=0.4)

    canvas=FigureCanvasTkAgg(fig,master=self.prozor_1)
    canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

    #liste u koje se spremaju podaci za crtanje grafova
    x1=[]
    y1=[]
    x2=[]
    y2=[]
    x3=[]
    y3=[]
```

Slika 19. Definiranje funkcije motor_1

Unutar funkcije definira se nova funkcija koja se zove *animate*, njome se crtaju grafovi. Na početku funkcije nalazi se *if* petlja kojom se vrši provjera motora. Ukoliko se uistinu radi o motoru broj 1, izvršava se sve što se nalazi unutar petlje, odnosno stvaraju se tri grafa. Na apscisama se nalazi brojač *i* koji je definiran gore na samom početku, kako se njegov broj povećava, apscisa se pomiče. Na ordinate se dodaju određeni primljeni podaci ovisno o tipu grafa, na prvi graf idu podaci za brzinu, na drugi podaci za struju, a na treći za poziciju. Podaci se uzimaju kako dolaze putem serijske komunikacije i odmah ucrtavaju na graf. Ispod svakog grafa dodana je labela u koju se upisuje trenutna vrijednost brzine, struje i pozicije. Na kraju se izvan ove funkcije, odnosno unutar primarne funkcije (*motor_1*), definira animacija koja će omogućiti kretanje grafa u realnom vremenu. [Slika 20.]

```
def animate(z):
    if M1[1]==1:
        x1.append(i)
        y1.append(M1[2])

        ax1.clear()
        ax1.plot(x1,y1)
        ax1.set_title('Brzina')
        ax1.set_xlim(i-250,i)
        ax1.set_ylim(-20,270)

        trenutna_brzina=Label(self.prozor_1,text='v=%s' %M1[2],font=('Calibri', 12),height=1,width=6)
        trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

        x2.append(i)
        y2.append(M1[3])

        ax2.clear()
        ax2.plot(x2,y2)
        ax2.set_title('Struja')
        ax2.set_xlim(i-250,i)
        ax2.set_ylim(-20,270)

        trenutna_struja=Label(self.prozor_1,text='i=%s' %M1[3],font=('Calibri', 12),height=1,width=6)
        trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

        x3.append(i)
        y3.append(M1[4])

        ax3.clear()
        ax3.plot(x3,y3)
        ax3.set_title('Pozicija')
        ax3.set_xlim(i-250,i)
        ax3.set_ylim(-20,270)

        trenutna_pozicija=Label(self.prozor_1,text='p=%s' %Data[4],font=('Calibri', 12),height=1,width=6)
        trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)


self.ani=animation.FuncAnimation(fig,animate,interval=10)    #animation omogućava kretanje grafa u vremenu
plt.show()
```

Slika 20. Definiranje funkcije animate

Unutar klase *Telemetrija* potrebno je još definirati funkciju koja će se pridodati tipki „Završi“. Ta funkcija zove se *iskljucenje* i unutar nje definirano je zatvaranje prozora i ulaza serijske komunikacije. [Slika 21.]

I na samom kraju programskog koda definirano je pozivanje glavnog prozora pokretanjem programa. [Slika 21.]

Cijeli programski kod nalazi se u prilogu.

```
#zatvaranje prozora i komunikacije
def iskljucenje(self):
    glavniProzor.destroy()
    ser.close()


#pozivanje glavnog prozora pokretanjem programa
gumb=Telemetrija()
glavniProzor.mainloop()
```

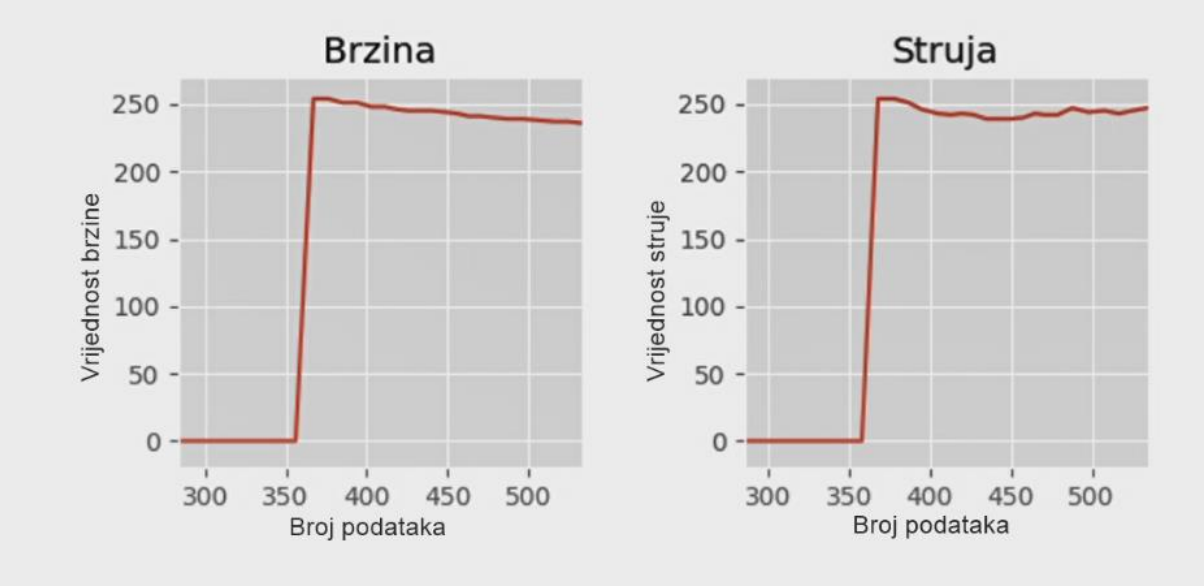
Slika 21. Definiranje funkcije *iskljucenje* i pozivanje glavnog prozora

6. REZULTATI

Putem bluetooth komunikacije računalo se spaja s KOPACK invalidskim vozilom, u programski kod se upisuje odgovarajući ulaz za serijsku komunikaciju te se pokreće program i vrše se testiranja. Gibanjem vozila prate se događanja na grafu, prati se što se događa u danom trenutku s brzinom motora, strujom i pozicijom.

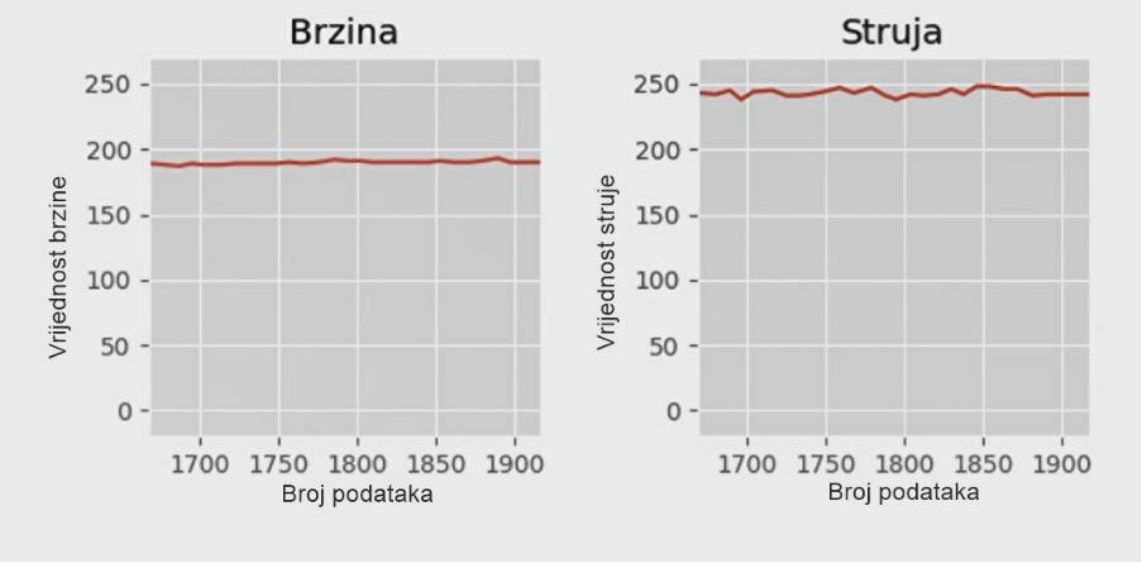
Budući da su rezultati videozapisi, uslikani su njihovi pojedini dijelovi radi razmatranja, a kako se podaci spremaju tijekom svakog praćenja i u tekstualnu datoteku, naknadno su napravljeni grafovi u programskom jeziku MATLAB kako bi se bolje prikazali dobiveni rezultati.

Na slici 22. prikazani su grafički prikazi brzine i struje za motor broj 1, to je motor koji pokreće prednji lijevi kotač. U danom trenutku vozilo je kretalo iz stanja mirovanja pravocrtno prema naprijed i na grafovima se može vidjeti kako je brzina u kotaču naglo porasla kad je vozilo krenulo, a isto tako i struja je skočila, vidi se koliko je struje bilo potrebno u motoru za ostvarenje ove brzine.



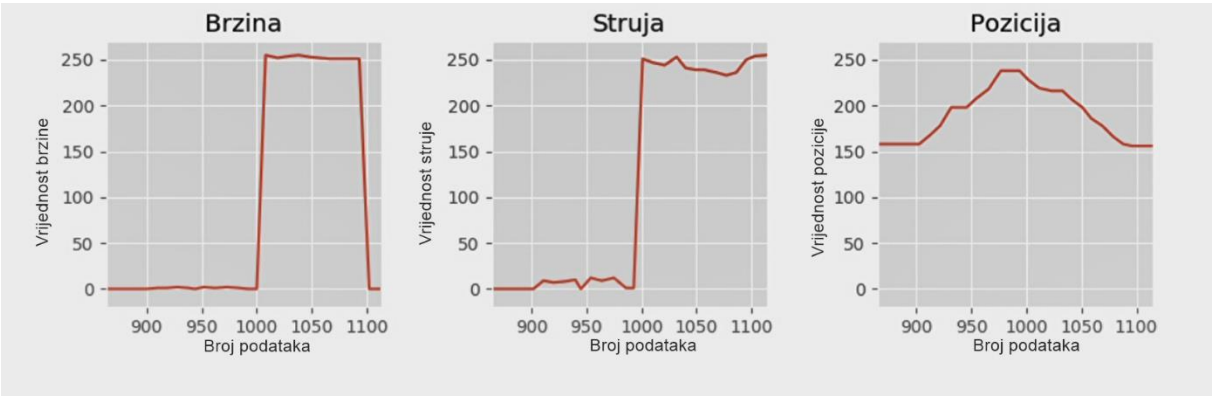
Slika 22. Dio videozapisa promatranja motora 1

Na istom videozapisu uslikan je dio kada se vozilo kreće konstantnom brzinom još uvijek pravocrtno i tu se može primijetiti da je i struja u kotaču otprilike cijelo vrijeme konstantna. [Slika 23.]



Slika 23. Drugi dio videozapisa promatranja motora 1

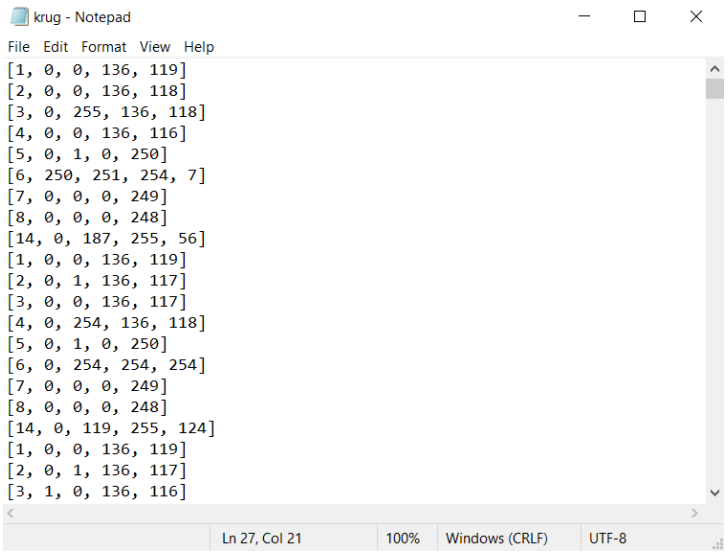
Motor broj 5 zadužen je za skretanje kotača, a na idućoj slici vidi se što se događa kada se vozilo zakreće. Tijekom početka skretanja može se primijetiti nagli porast brzine, a ujedno i isti takav porast struje, iz rada motora vidi se kolika je iskorištena struja za tu zadanu brzinu. Na zadnjem grafu vidi se postupan prijelaz pozicije kotača iz početne vrijednosti do maksimalne vrijednosti skretanja te njegovo vraćanje u početno stanje. [Slika 24.]



Slika 24. Dio videozapisa promatranja motora 5

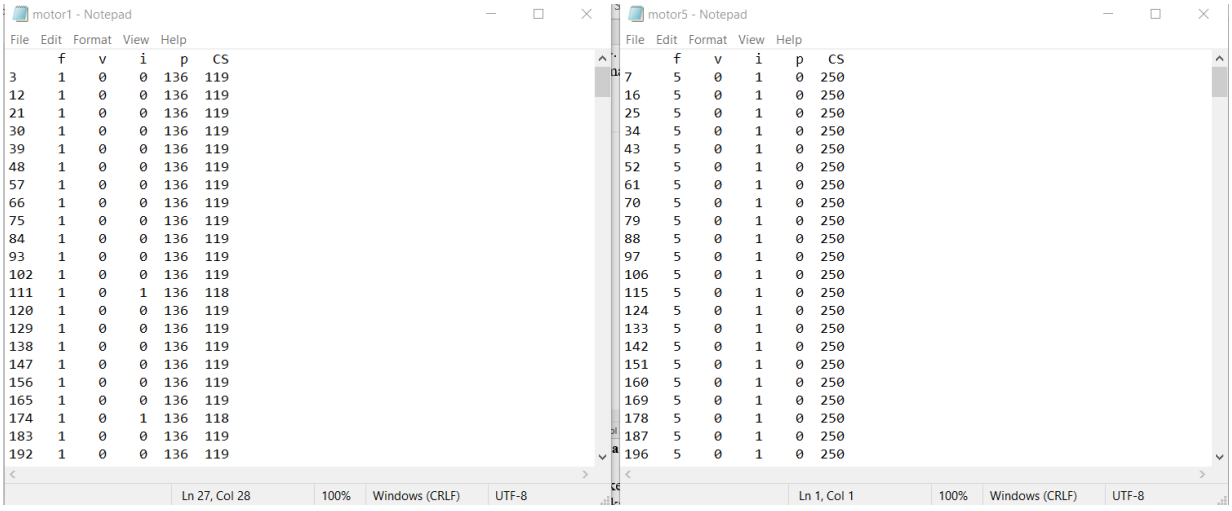
6.1. Rezultati u MATLAB-u

U nastavku će biti prikazani rezultati dobiveni pomoću MATLAB-a. Kao što je već spomenuto, sa svakim pokretanjem programa, podaci se spremaju u tekstualnu datoteku i to na način kako je prikazano na slici 25. gdje je prvi podatak u listi broj motora, drugi brzina, treći struja, četvrti pozicija i zadnji suma.



Slika 25. Tekstualna datoteka sa spremljenim podacima

Kako bi se podaci iz tekstualne datoteke mogli iskoristiti, prvo ih treba razvrstati po motorima i spremiti u drugačijem obliku koji se kasnije može importirati u MATLAB. U Pythonu je napisan manji programski kod koji razvrstava podatke i sprema ih u obliku stupaca s pripadajućim vrijednostima. [Slika 26. i 27.]



Slika 26. Tekstualna datoteka s podacima za MATLAB

```python
from ast import literal_eval
import pandas as pd
import numpy as np
import csv

M1=open('motor1.txt','a+')
M1.truncate(0)
M2=open('motor2.txt','a+')
M2.truncate(0)
M3=open('motor3.txt','a+')
M3.truncate(0)
M4=open('motor4.txt','a+')
M4.truncate(0)
M5=open('motor5.txt','a+')
M5.truncate(0)
M6=open('motor6.txt','a+')
M6.truncate(0)
M7=open('motor7.txt','a+')
M7.truncate(0)
M8=open('motor8.txt','a+')
M8.truncate(0)


lst=[]
with open('krug.txt','r') as N:
    for line in N:
        vec=literal_eval(line)
        lst.append(vec)

podaci=pd.DataFrame(np.array(lst),columns=['f','v','i','p','CS'])
pd.set_option("display.max_rows",4000)
with open('podaci.txt','w') as P:
    P.write(str(podaci))


p1=podaci.loc[podaci['f']==1]
p1.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor1.xlsx',index=None, header=True)
p2=podaci.loc[podaci['f']==2]
p2.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor2.xlsx',index=None, header=True)
p3=podaci.loc[podaci['f']==3]
p3.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor3.xlsx',index=None, header=True)
p4=podaci.loc[podaci['f']==4]
p4.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor4.xlsx',index=None, header=True)
p5=podaci.loc[podaci['f']==5]
p5.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor5.xlsx',index=None, header=True)
p6=podaci.loc[podaci['f']==6]
p6.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor6.xlsx',index=None, header=True)
p7=podaci.loc[podaci['f']==7]
p7.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor7.xlsx',index=None, header=True)
p8=podaci.loc[podaci['f']==8]
p8.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor8.xlsx',index=None, header=True)
```

```python
with open('motor1.txt','a') as M1:
    M1.write(str(p1))

with open('motor2.txt','a') as M2:
    M2.write(str(p2))

with open('motor3.txt','a') as M3:
    M3.write(str(p3))

with open('motor4.txt','a') as M4:
    M4.write(str(p4))

with open('motor5.txt','a') as M5:
    M5.write(str(p5))

with open('motor6.txt','a') as M6:
    M6.write(str(p6))

with open('motor7.txt','a') as M7:
    M7.write(str(p7))

with open('motor8.txt','a') as M8:
    M8.write(str(p8))
```

Slika 27. Programski kod za spremanje podataka u odgovarajući oblik

Nakon što se podaci spreme u odgovarajući oblik, importiraju se u MATLAB i crtaju se potrebni grafovi pomoću jednostavnog koda koji je prikazan na slici 28. Programski kod je analogan za sve prikazane grafove i svi su priloženi na kraju dokumenta.

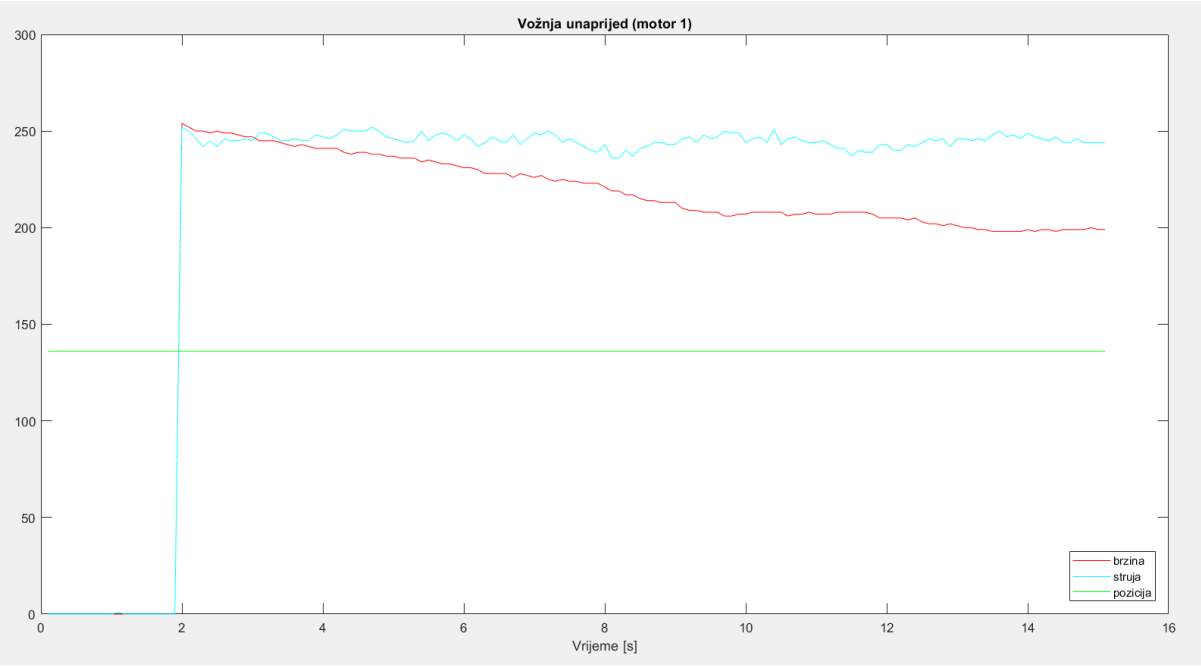
naprijed_lagano.m | naprijed_nazad.m | rotacija_udesno.m | voznja_ukrug.m*

```
x1=1:287;
x=x1/10;
plot(x,v,'r',x,i1,'y',x,p,'b')
legend('brzina','struja','pozicija','Location','southeast')
xlabel('Vrijeme [s]')
title('Vožnja ukrug (motor 5)')
```

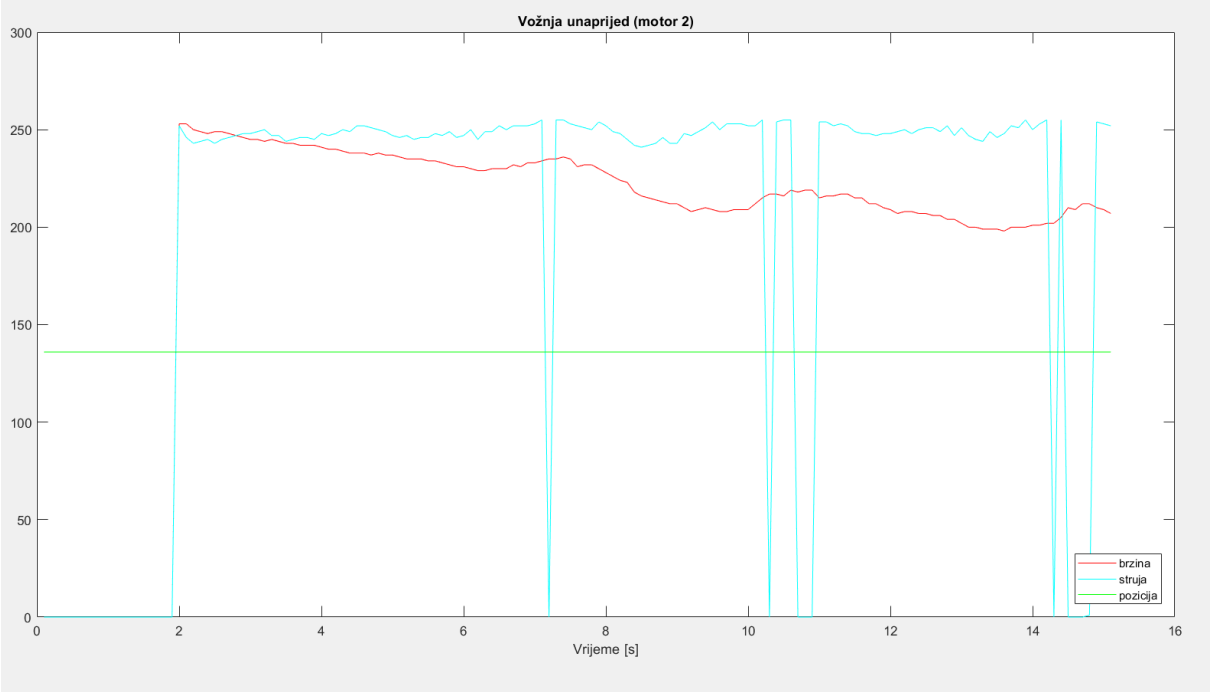
Slika 28. Programski kod u MATLAB-u za crtanje grafova

Vozilo je testirano na vožnju unaprijed, vožnju naprijed-nazad, vožnju ukrug te okretanje u mjestu. Za to vrijeme praćeno je što se događa s brzinom, strujom i pozicijom i to je prikazano na idućim grafovima.

Tijekom vožnje unaprijed brzina motora prednjeg lijevog i prednjeg desnog kotača naglo je porasla kada se vozilo pokrenulo, isto kao i struja. Zatim je do kraja praćenja polagano usporavalo što se može vidjeti na oba grafa, brzina oba motora je polagano opadala. Međutim, struja je u prvom motoru bila donekle konstantna dok je struja u drugom motoru imala iznenadne padove. Pozicija motora cijelo vrijeme je bila konstantna. [Slika 29. i Slika 30.]

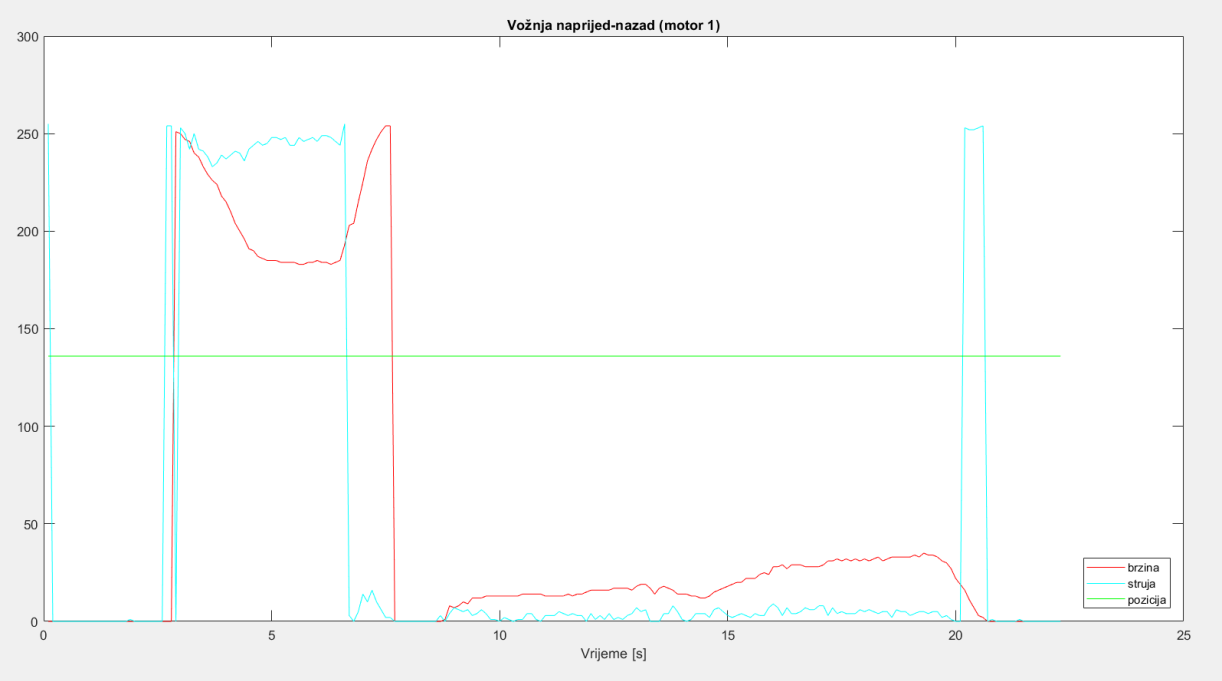


Slika 29. Grafički prikaz za motor 1 tijekom vožnje unaprijed

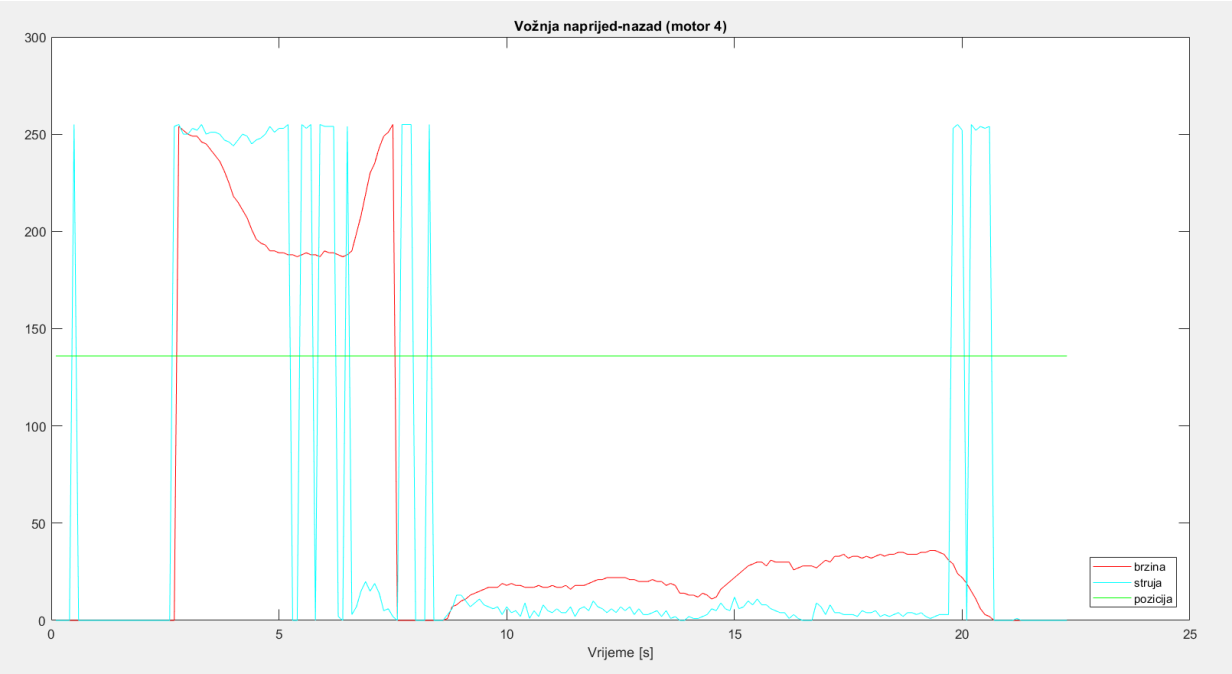


Slika 30. Grafički prikaz za motor 2 tijekom vožnje unaprijed

Napravljena je vožnja naprijed-nazad i promotreno je što se događa s motorom prednjeg lijevog kotača i stražnjeg desnog kotača. Vozilo se pokrenulo i prvo malo usporilo pa ponovno ubrzalo te je zatim naglo stalo i krenulo voziti unazad. Sve to može se primijetiti na grafovima prateći crvenu liniju za brzinu, može se uočiti da brzine u oba motora imaju približno jednake iznose. Struja u motoru 1 više-manje prati zbivanja vozila, odnosno brzinu motora, s iznenadnim skokom pri kraju vožnje, dok struja motora 4 ima iznenadne padove tijekom vožnje naprijed, te kao i struja motora 1 iznenadne skokove na kraju vožnje. Pozicija je ponovno konstantnog iznosa. [Slika 31. i Slika 32.]



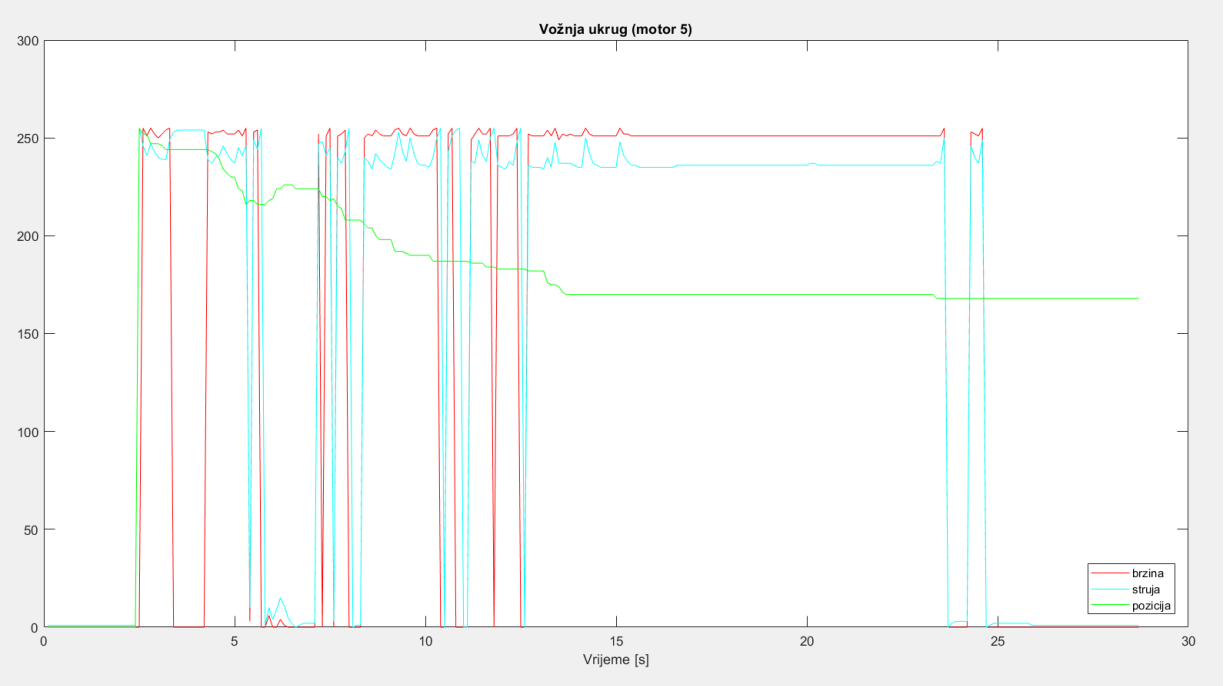
Slika 31. Grafički prikaz za motor 1 tijekom vožnje naprijed-nazad



Slika 32. Grafički prikaz za motor 4 tijekom vožnje naprijed-nazad

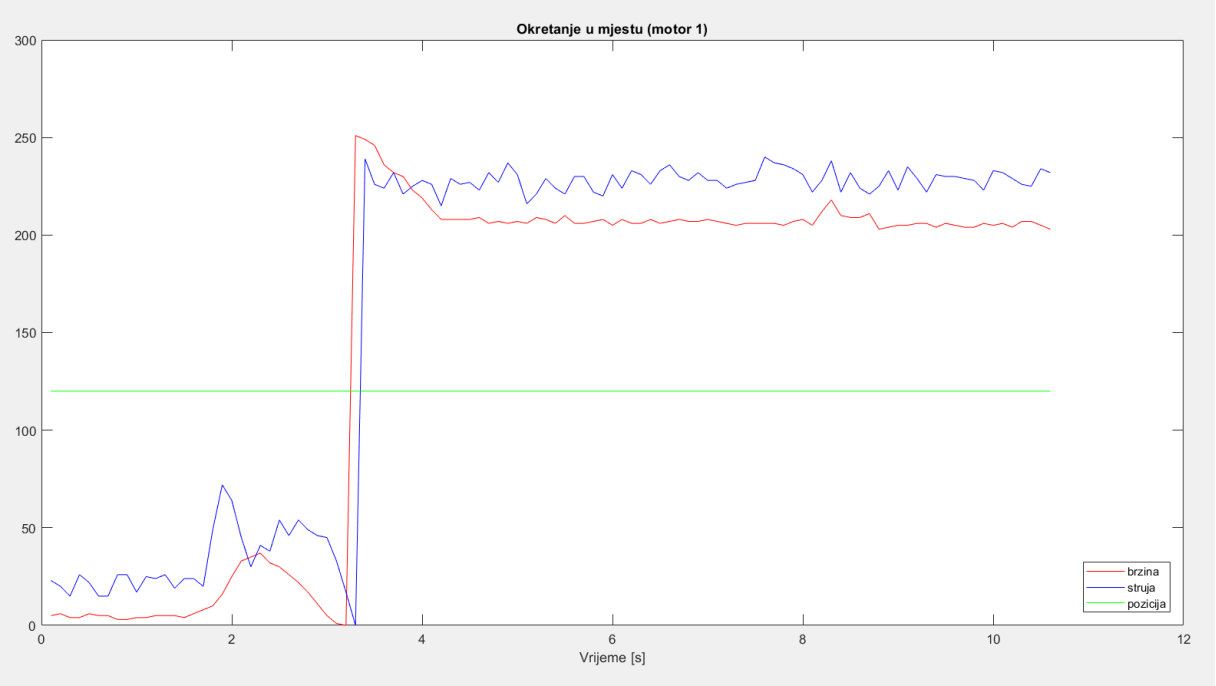
Dalje je testirano što se zbiva kada se vozilo kreće ukrug. Za ovaj primjer odabran je motor broj 5 jer on služi za zakretanje kotača pa je u ovom slučaju primarno praćenje pozicije motora. Tu se vidi kako se pozicija kotača mijenja dok vozilo ide ukrug i kako se smiruje oko određene

vrijednosti kada vozilo ponovno krene voziti pravocrtno. Brzina i struja ovog motora imaju stalne skokove i padove tijekom vožnje ukrug. [Slika 33.]



Slika 33. Grafički prikaz za motor 5 tijekom vožnje ukrug

Za kraj je napravljeno okretanje vozila u mjestu. Na motoru 1 vidi se da je to napravljeno otprilike konstantnom brzinom i strujom s povremenim manjim padovima i skokovima, dok je pozicija konstantna. Na motoru 5 vidi se slabi porast i postepeni pad struje, dok su brzina i pozicija na nuli. [Slika 34. i Slika 35.]



Slika 34. Grafički prikaz za motor 1 tijekom okretanja vozila u mjestu

7. ZAKLJUČAK

Izgrađeno je sučelje koje prikuplja podatke i grafički ih prikazuje u programskom jeziku Python. Budući da je Python interpreterski jezik, programi napisani u njemu izvršavaju se sporije u usporedbi s kompajlerskim jezicima, kao što su to npr. C, C++ i sl. Program unosi određena zakašnjenja i zato nije najbolji način za prikazivanje rezultata u stvarnom vremenu. Stoga je zaključak iz navedenog da bi se za ovakve potrebe izvedbe trebao upotrijebiti ipak neki brži programski jezik s obzirom da je u telemetriji od iznimne važnosti točnost i preciznost. Međutim, Python se pokazao kao koristan i jednostavan alat za prikupljanje podataka u tekstualne datoteke te manipulaciju podacima u tim datotekama. Za vjerniji prikaz prikupljenih podataka izabran je u konačnici MATLAB kako bi se mogli razmotriti odabrani elementi procesa za sve motore. MATLAB je izvrstan alat za grafičko prikazivanje željenih procesa.

Ipak, ovim radom pokazano je da je moguće isprogramirati zadani zadatak i ostavljen je prostor za unapređenje i optimizaciju. Osim toga, ovaj rad pokazao je zašto je telemetrija vrlo važna u današnjem svijetu i kako gotovo da nema područja gdje se ona danas ne pojavljuje, stoga i dalje treba nastaviti raditi na njenom usavršavanju i traženju novih područja gdje bi se ona mogla primijeniti.

LITERATURA

[1] https://science.jrank.org/pages/6722/Telemetry-System-components.html?fbclid=IwAR1cR3I_QpSJUePiUvOXeOZIxo4iZD6GpDAkwhcKZ1mbQZDmCX_iwkpNtw0

[2] https://hr.m.wikipedia.org/wiki/Daljinsko_mjerenje?fbclid=IwAR1WcV8wK7PukEms06H86CG_W1SRBBdDJyVz23G4ilh0DTAa_X5OXyQXNAQ

[3] https://en.wikipedia.org/wiki/Telemetry#cite_note-6

[4] http://arhiva.vidiauto.com/autotech/telemetrijom_do_ekonomicne_voznje/?fbclid=IwAR2-FEAJxrbN-UEiSRircnnaQ_WHbUyAwxtlG0uPTwL5Tg4jLppovXRLepA

[5] https://en.wikipedia.org/wiki/Voyager_1

[6] https://en.wikipedia.org/wiki/JPL_Mars_Helicopter_Scout

[7] https://www.intradiagnostics.com/eeg--neurotelemetry.html

[8] https://www.neotys.com/blog/neotyspac-telemetry-essential-ingredient-to-success-by-todd-decapua/

[9] https://www.mbcontrol.com/telemetry-system/

[10] https://space.stackexchange.com/questions/26526/what-are-the-louver-like-structures-on-the-sides-of-the-mariner-4-probe?rq=1

[11] https://www.kaspersky.com.br/blog/carros-de-producao-permitem-correr-como-no-video-game/4144/

[12] https://www.persistencemarketresearch.com/news/2017/october/technologies-used-for-precision-farming

[13] https://www.jpl.nasa.gov/spaceimages/details.php?id=PIA17462

[14] https://www.researchgate.net/figure/Electroencephalographic-EEG-and-multimodality-monitoring-in-a-70-year-old-woman-with_fig5_274636758

[15] https://www.sumologic.com/insight/what-is-telemetry/

PRILOZI

I. CD-R disc

II. Python kod - telemetrija vozila

III. Python kodovi – tekstualne datoteke za MATLAB

IV. MATLAB kodovi - grafovi

Python kod – telemetrija vozila

```
##ZAVRŠNI RAD, Klara Pejić, 2019., Zagreb
##Telemetrija KOPACK vozila
##Fakultet strojarstva i brodogradnje
##Sveučilište u Zagrebu

from tkinter import *   #grafičko sučelje

import serial   #serijska komunikacija
import threading    #istovremeni rad komunikacije i sučelja

#paketi za crtanje grafova
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,NavigationToolbar2Tk
matplotlib.use('TkAgg')
from matplotlib.figure import Figure

ser=serial.Serial('COM3', 57600)  #otvaranje ulaza serijske komunikacije
def SERIAL():
    f1=open('text.txt','r+')
    f1.truncate(0)              #pražnjenje tekstualne datoteke na početku rada
    global Data,i,M1,M2,M3,M4,M5,M6,M7,M8
    i=0
    k=1

    while 1:
        Data1=ser.read().decode('utf-8') #'{'
        Data2=ord(ser.read())           #f
        Data3=ord(ser.read())           #v-brzina
```

```
Data4=ord(ser.read())          #i-struja
Data5=ord(ser.read())          #p-pozicija
Data6=ord(ser.read())          #CS-suma
Data = [Data1,Data2,Data3,Data4,Data5,Data6]
print (Data)
i+=1

#petlja koja služi razvrstavanju podataka
if k==1:
    M1=Data
    k+=1
elif k==2:
    M2=Data
    k+=1
elif k==3:
    M3=Data
    k+=1
elif k==4:
    M4=Data
    k+=1
elif k==5:
    M5=Data
    k+=1
elif k==6:
    M6=Data
    k+=1
elif k==7:
    M7=Data
    k+=1
elif k==8:
    M8=Data
    k=1
```

```
        with open('text.txt','a') as f2:        #tekstualna datoteka u koju se spremaju podaci koji se šalju serijskom komunikacijom
            f2.write('%s \n' %str(Data[1:6]))

t1=threading.Thread(target=SERIAL,daemon=True)  #threading omogućuje rad u sučelju dok se u pozadini konstantno primaju podaci serijskom komunikacijom
t1.start()



#izgled glavnog prozora
glavniProzor=Tk()
glavniProzor.geometry('700x500')
glavniProzor.configure(bg='gray')
glavniProzor.title('Telemetrija vozila')

class Telemetrija():
    def __init__(self):
        #stvaranje gumba
        gumb1=Button(text='Motor 1',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_1)
        gumb1.place(relx=0.35, rely=0.1, anchor=CENTER)
        gumb2=Button(text='Motor 2',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_2)
        gumb2.place(relx=0.65, rely=0.1, anchor=CENTER)
        gumb3=Button(text='Motor 3',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_3)
        gumb3.place(relx=0.35, rely=0.3, anchor=CENTER)
        gumb4=Button(text='Motor 4',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_4)
        gumb4.place(relx=0.65, rely=0.3, anchor=CENTER)
        gumb3=Button(text='Motor 5',fg='black',bg='white',font=('Calibri', 24, 'bold'),command=self.motor_5)
        gumb3.place(relx=0.35, rely=0.5, anchor=CENTER)
```

```
        gumb3=Button(text='Motor 6',fg='black',bg='white',font=('Calibri', 24,
'bold'),command=self.motor_6)
        gumb3.place(relx=0.65, rely=0.5, anchor=CENTER)
        gumb3=Button(text='Motor 7',fg='black',bg='white',font=('Calibri', 24,
'bold'),command=self.motor_7)
        gumb3.place(relx=0.35, rely=0.7, anchor=CENTER)
        gumb3=Button(text='Motor 8',fg='black',bg='white',font=('Calibri', 24,
'bold'),command=self.motor_8)
        gumb3.place(relx=0.65, rely=0.7, anchor=CENTER)

        gumb=Button(text='Završi',fg='black', bg='white',font=('Calibri', 20,
'bold'),command=self.iskljucenje)
        gumb.place(relx=0.5,rely=0.9,anchor=CENTER)



    def motor_1(self):
        #kreiranje prozora za prvi motor
        self.prozor_1=Tk()
        self.prozor_1.title('Motor 1')
        self.prozor_1.geometry('1100x700')
        self.prozor_1.resizable()
        self.prozor_1.config(bg='white')

        #postavljanje okvira za grafove
        style.use('ggplot')
        fig=plt.Figure(figsize=(12, 8))
        ax1=fig.add_subplot(2,3,1)
        ax2=fig.add_subplot(2,3,2)
        ax3=fig.add_subplot(2,3,3)
        fig.subplots_adjust(bottom=0.2,wspace=0.4)

        canvas=FigureCanvasTkAgg(fig,master=self.prozor_1)
        canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)
```

```
#liste u koje se spremaju podaci za crtanje grafova
x1=[]
y1=[]
x2=[]
y2=[]
x3=[]
y3=[]


def animate(z):
    if M1[1]==1:
        x1.append(i)
        y1.append(M1[2])

        ax1.clear()
        ax1.plot(x1,y1)
        ax1.set_title('Brzina')
        ax1.set_xlim(i-250,i)
        ax1.set_ylim(-20,270)

        trenutna_brzina=Label(self.prozor_1,text='v=%s' %M1[2],font=('Calibri',
12),height=1,width=6)
        trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

        x2.append(i)
        y2.append(M1[3])

        ax2.clear()
        ax2.plot(x2,y2)
        ax2.set_title('Struja')
        ax2.set_xlim(i-250,i)
        ax2.set_ylim(-20,270)
```

```
            trenutna_struja=Label(self.prozor_1,text='i=%s' %M1[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M1[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_1,text='p=%s' %Data[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)


        self.ani=animation.FuncAnimation(fig,animate,interval=10)    #animation omogućava
kretanje grafa u vremenu
        plt.show()



    #analogno za sve ostale motore
    def motor_2(self):
        self.prozor_2=Tk()
        self.prozor_2.title('Motor 2')
        self.prozor_2.geometry('1100x700')
        self.prozor_2.resizable()
        self.prozor_2.config(bg='white')

        style.use('ggplot')
        fig=plt.Figure(figsize=(8, 8))
```

```
        ax1=fig.add_subplot(2,3,1)
        ax2=fig.add_subplot(2,3,2)
        ax3=fig.add_subplot(2,3,3)
        fig.subplots_adjust(bottom=0.2,wspace=0.4)

        canvas=FigureCanvasTkAgg(fig,master=self.prozor_2)
        canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

        x1=[]
        y1=[]
        x2=[]
        y2=[]
        x3=[]
        y3=[]

        def animate(z):
            if M2[1]==2:
                x1.append(i)
                y1.append(M2[2])

                ax1.clear()
                ax1.plot(x1,y1)
                ax1.set_title('Brzina')
                ax1.set_xlim(i-250,i)
                ax1.set_ylim(-20,270)

                trenutna_brzina=Label(self.prozor_2,text='v=%s' %M2[2],font=('Calibri',
12),height=1,width=6)
                trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

                x2.append(i)
                y2.append(M2[3])

                ax2.clear()
```

```
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_2,text='i=%s' %M2[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M2[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_2,text='p=%s' %M2[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)


        self.ani=animation.FuncAnimation(fig,animate,interval=10)
        plt.show()


    def motor_3(self):
        self.prozor_3=Tk()
        self.prozor_3.title('Motor 3')
        self.prozor_3.geometry('1100x700')
        self.prozor_3.resizable()
        self.prozor_3.config(bg='white')
```

```
        style.use('ggplot')
        fig=plt.Figure(figsize=(8, 8))
        ax1=fig.add_subplot(2,3,1)
        ax2=fig.add_subplot(2,3,2)
        ax3=fig.add_subplot(2,3,3)
        fig.subplots_adjust(bottom=0.2,wspace=0.4)

        canvas=FigureCanvasTkAgg(fig,master=self.prozor_3)
        canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

        x1=[]
        y1=[]
        x2=[]
        y2=[]
        x3=[]
        y3=[]

        def animate(z):
            if M3[1]==3:
                x1.append(i)
                y1.append(M3[2])

                ax1.clear()
                ax1.plot(x1,y1)
                ax1.set_title('Brzina')
                ax1.set_xlim(i-250,i)
                ax1.set_ylim(-20,270)

                trenutna_brzina=Label(self.prozor_3,text='v=%s' %M3[2],font=('Calibri',
12),height=1,width=6)
                trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

                x2.append(i)
                y2.append(M3[3])
```

```
            ax2.clear()
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_3,text='i=%s' %M3[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M3[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_3,text='p=%s' %M3[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)



        self.ani=animation.FuncAnimation(fig,animate,interval=10)
        plt.show()

    def motor_4(self):
        self.prozor_4=Tk()
        self.prozor_4.title('Motor 4')
        self.prozor_4.geometry('1100x700')
        self.prozor_4.resizable()
        self.prozor_4.config(bg='white')
```

```
        style.use('ggplot')
        fig=plt.Figure(figsize=(8, 8))
        ax1=fig.add_subplot(2,3,1)
        ax2=fig.add_subplot(2,3,2)
        ax3=fig.add_subplot(2,3,3)
        fig.subplots_adjust(bottom=0.2,wspace=0.4)

        canvas=FigureCanvasTkAgg(fig,master=self.prozor_4)
        canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

        x1=[]
        y1=[]
        x2=[]
        y2=[]
        x3=[]
        y3=[]

        def animate(z):
            if M4[1]==4:
                x1.append(i)
                y1.append(M4[2])

                ax1.clear()
                ax1.plot(x1,y1)
                ax1.set_title('Brzina')
                ax1.set_xlim(i-250,i)
                ax1.set_ylim(-20,270)

                trenutna_brzina=Label(self.prozor_4,text='v=%s' %M4[2],font=('Calibri',
12),height=1,width=6)
                trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

                x2.append(i)
```

```
            y2.append(M4[3])

            ax2.clear()
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_4,text='i=%s' %M4[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M4[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_4,text='p=%s' %M4[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)


        self.ani=animation.FuncAnimation(fig,animate,interval=10)
        plt.show()


    def motor_5(self):
        self.prozor_5=Tk()
        self.prozor_5.title('Motor 5')
        self.prozor_5.geometry('1100x700')
```

```
        self.prozor_5.resizable()
        self.prozor_5.config(bg='white')

        style.use('ggplot')
        fig=plt.Figure(figsize=(8, 8))
        ax1=fig.add_subplot(2,3,1)
        ax2=fig.add_subplot(2,3,2)
        ax3=fig.add_subplot(2,3,3)
        fig.subplots_adjust(bottom=0.2,wspace=0.4)

        canvas=FigureCanvasTkAgg(fig,master=self.prozor_5)
        canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

        x1=[]
        y1=[]
        x2=[]
        y2=[]
        x3=[]
        y3=[]

        def animate(z):
            if M5[1]==5:
                x1.append(i)
                y1.append(M5[2])

                ax1.clear()
                ax1.plot(x1,y1)
                ax1.set_title('Brzina')
                ax1.set_xlim(i-250,i)
                ax1.set_ylim(-20,270)

                trenutna_brzina=Label(self.prozor_5,text='v=%s' %M5[2],font=('Calibri',
12),height=1,width=6)
                trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)
```

```
            x2.append(i)
            y2.append(M5[3])

            ax2.clear()
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_5,text='i=%s' %M5[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M5[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_5,text='p=%s' %M5[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)


        self.ani=animation.FuncAnimation(fig,animate,interval=10)
        plt.show()


    def motor_6(self):
        self.prozor_6=Tk()
```

```
self.prozor_6.title('Motor 6')
self.prozor_6.geometry('1100x700')
self.prozor_6.resizable()
self.prozor_6.config(bg='white')

style.use('ggplot')
fig=plt.Figure(figsize=(8, 8))
ax1=fig.add_subplot(2,3,1)
ax2=fig.add_subplot(2,3,2)
ax3=fig.add_subplot(2,3,3)
fig.subplots_adjust(bottom=0.2,wspace=0.4)

canvas=FigureCanvasTkAgg(fig,master=self.prozor_6)
canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

x1=[]
y1=[]
x2=[]
y2=[]
x3=[]
y3=[]

def animate(z):
    if M6[1]==6:
        x1.append(i)
        y1.append(M6[2])

        ax1.clear()
        ax1.plot(x1,y1)
        ax1.set_title('Brzina')
        ax1.set_xlim(i-250,i)
        ax1.set_ylim(-20,270)
```

```
            trenutna_brzina=Label(self.prozor_6,text='v=%s' %M6[2],font=('Calibri',
12),height=1,width=6)
            trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

            x2.append(i)
            y2.append(M6[3])

            ax2.clear()
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_6,text='i=%s' %M6[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M6[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_6,text='p=%s' %M6[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)


        self.ani=animation.FuncAnimation(fig,animate,interval=10)
        plt.show()
```

```
def motor_7(self):
    self.prozor_7=Tk()
    self.prozor_7.title('Motor 7')
    self.prozor_7.geometry('1100x700')
    self.prozor_7.resizable()
    self.prozor_7.config(bg='white')

    style.use('ggplot')
    fig=plt.Figure(figsize=(8, 8))
    ax1=fig.add_subplot(2,3,1)
    ax2=fig.add_subplot(2,3,2)
    ax3=fig.add_subplot(2,3,3)
    fig.subplots_adjust(bottom=0.2,wspace=0.4)

    canvas=FigureCanvasTkAgg(fig,master=self.prozor_7)
    canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

    x1=[]
    y1=[]
    x2=[]
    y2=[]
    x3=[]
    y3=[]

    def animate(z):

        if M7[1]==7:
            x1.append(i)
            y1.append(M7[2])

            ax1.clear()
            ax1.plot(x1,y1)
            ax1.set_title('Brzina')
```

```
            ax1.set_xlim(i-250,i)
            ax1.set_ylim(-20,270)

            trenutna_brzina=Label(self.prozor_7,text='v=%s' %M7[2],font=('Calibri',
12),height=1,width=6)
            trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

            x2.append(i)
            y2.append(M7[3])

            ax2.clear()
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_7,text='i=%s' %M7[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M7[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_7,text='p=%s' %M7[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.8,rely=0.56,anchor=CENTER)
```

```
    self.ani=animation.FuncAnimation(fig,animate,interval=10)
    plt.show()


def motor_8(self):
    self.prozor_8=Tk()
    self.prozor_8.title('Motor 8')
    self.prozor_8.geometry('1100x700')
    self.prozor_8.resizable()
    self.prozor_8.config(bg='white')

    style.use('ggplot')
    fig=plt.Figure(figsize=(8, 8))
    ax1=fig.add_subplot(2,3,1)
    ax2=fig.add_subplot(2,3,2)
    ax3=fig.add_subplot(2,3,3)
    fig.subplots_adjust(bottom=0.2,wspace=0.4)

    canvas=FigureCanvasTkAgg(fig,master=self.prozor_8)
    canvas.get_tk_widget().pack(side=BOTTOM,fill=BOTH)

    x1=[]
    y1=[]
    x2=[]
    y2=[]
    x3=[]
    y3=[]

    def animate(z):
        if M8[1]==8:
            x1.append(i)
            y1.append(M8[2])

            ax1.clear()
```

```
            ax1.plot(x1,y1)
            ax1.set_title('Brzina')
            ax1.set_xlim(i-250,i)
            ax1.set_ylim(-20,270)

            trenutna_brzina=Label(self.prozor_8,text='v=%s' %M8[2],font=('Calibri',
12),height=1,width=6)
            trenutna_brzina.place(relx=0.16,rely=0.56,anchor=CENTER)

            x2.append(i)
            y2.append(M8[3])

            ax2.clear()
            ax2.plot(x2,y2)
            ax2.set_title('Struja')
            ax2.set_xlim(i-250,i)
            ax2.set_ylim(-20,270)

            trenutna_struja=Label(self.prozor_8,text='i=%s' %M8[3],font=('Calibri',
12),height=1,width=6)
            trenutna_struja.place(relx=0.44,rely=0.56,anchor=CENTER)

            x3.append(i)
            y3.append(M8[4])

            ax3.clear()
            ax3.plot(x3,y3)
            ax3.set_title('Pozicija')
            ax3.set_xlim(i-250,i)
            ax3.set_ylim(-20,270)

            trenutna_pozicija=Label(self.prozor_8,text='p=%s' %M8[4],font=('Calibri',
12),height=1,width=6)
            trenutna_pozicija.place(relx=0.73,rely=0.56,anchor=CENTER)
```

```
        self.ani=animation.FuncAnimation(fig,animate,interval=10)
        plt.show()



    #zatvaranje prozora i komunikacije
    def iskljucenje(self):
        glavniProzor.destroy()
        ser.close()


#pozivanje glavnog prozora pokretanjem programa
gumb=Telemetrija()
glavniProzor.mainloop()
```

Python kodovi – tekstualne datoteke za MATLAB

```
##vožnja unaprijed

from ast import literal_eval
import pandas as pd
import numpy as np
import csv

M1=open('motor1.txt','a+')
M1.truncate(0)
M2=open('motor2.txt','a+')
M2.truncate(0)
M3=open('motor3.txt','a+')
M3.truncate(0)
M4=open('motor4.txt','a+')
M4.truncate(0)
M5=open('motor5.txt','a+')
M5.truncate(0)
M6=open('motor6.txt','a+')
M6.truncate(0)
M7=open('motor7.txt','a+')
M7.truncate(0)
M8=open('motor8.txt','a+')
M8.truncate(0)

lst=[]
with open('naprijed lagano.txt','r') as N:
    for line in N:
        vec=literal_eval(line)
        lst.append(vec)

podaci=pd.DataFrame(np.array(lst),columns=['f','v','i','p','CS'])
pd.set_option("display.max_rows",4000)
```

```
with open('podaci.txt','w') as P:
    P.write(str(podaci))

p1=podaci.loc[podaci['f']==1]
p1.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor1.xlsx',index=None, header=True)
p2=podaci.loc[podaci['f']==2]
p2.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor2.xlsx',index=None, header=True)
p3=podaci.loc[podaci['f']==3]
p3.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor3.xlsx',index=None, header=True)
p4=podaci.loc[podaci['f']==4]
p4.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor4.xlsx',index=None, header=True)
p5=podaci.loc[podaci['f']==5]
p5.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor5.xlsx',index=None, header=True)
p6=podaci.loc[podaci['f']==6]
p6.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor6.xlsx',index=None, header=True)
p7=podaci.loc[podaci['f']==7]
p7.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor7.xlsx',index=None, header=True)
p8=podaci.loc[podaci['f']==8]
p8.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed
lagano\motor8.xlsx',index=None, header=True)

with open('motor1.txt','a') as M1:
    M1.write(str(p1))
with open('motor2.txt','a') as M2:
    M2.write(str(p2))
with open('motor3.txt','a') as M3:
    M3.write(str(p3))
```

```
with open('motor4.txt','a') as M4:
    M4.write(str(p4))
with open('motor5.txt','a') as M5:
    M5.write(str(p5))
with open('motor6.txt','a') as M6:
    M6.write(str(p6))
with open('motor7.txt','a') as M7:
    M7.write(str(p7))
with open('motor8.txt','a') as M8:
    M8.write(str(p8))

##vožnja naprijed-nazad

from ast import literal_eval
import pandas as pd
import numpy as np
import csv

M1=open('motor1.txt','a+')
M1.truncate(0)
M2=open('motor2.txt','a+')
M2.truncate(0)
M3=open('motor3.txt','a+')
M3.truncate(0)
M4=open('motor4.txt','a+')
M4.truncate(0)
M5=open('motor5.txt','a+')
M5.truncate(0)
M6=open('motor6.txt','a+')
M6.truncate(0)
M7=open('motor7.txt','a+')
M7.truncate(0)
M8=open('motor8.txt','a+')
M8.truncate(0)
```

```
lst=[]
with open('naprijed_nazad.txt','r') as N:
    for line in N:
        vec=literal_eval(line)
        lst.append(vec)

podaci=pd.DataFrame(np.array(lst),columns=['f','v','i','p','CS'])
pd.set_option("display.max_rows",4000)
with open('podaci.txt','w') as P:
    P.write(str(podaci))

p1=podaci.loc[podaci['f']==1]
p1.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor1.xlsx',index=None, header=True)
p2=podaci.loc[podaci['f']==2]
p2.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor2.xlsx',index=None, header=True)
p3=podaci.loc[podaci['f']==3]
p3.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor3.xlsx',index=None, header=True)
p4=podaci.loc[podaci['f']==4]
p4.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor4.xlsx',index=None, header=True)
p5=podaci.loc[podaci['f']==5]
p5.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor5.xlsx',index=None, header=True)
p6=podaci.loc[podaci['f']==6]
p6.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor6.xlsx',index=None, header=True)
p7=podaci.loc[podaci['f']==7]
p7.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor7.xlsx',index=None, header=True)
p8=podaci.loc[podaci['f']==8]
```

```
p8.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\naprijed_nazad\motor8.xlsx',index=None, header=True)

with open('motor1.txt','a') as M1:
    M1.write(str(p1))
with open('motor2.txt','a') as M2:
    M2.write(str(p2))
with open('motor3.txt','a') as M3:
    M3.write(str(p3))
with open('motor4.txt','a') as M4:
    M4.write(str(p4))
with open('motor5.txt','a') as M5:
    M5.write(str(p5))
with open('motor6.txt','a') as M6:
    M6.write(str(p6))
with open('motor7.txt','a') as M7:
    M7.write(str(p7))
with open('motor8.txt','a') as M8:
    M8.write(str(p8))

##vožnja ukrug

from ast import literal_eval
import pandas as pd
import numpy as np
import csv

M1=open('motor1.txt','a+')
M1.truncate(0)
M2=open('motor2.txt','a+')
M2.truncate(0)
M3=open('motor3.txt','a+')
M3.truncate(0)
M4=open('motor4.txt','a+')
```

```
M4.truncate(0)
M5=open('motor5.txt','a+')
M5.truncate(0)
M6=open('motor6.txt','a+')
M6.truncate(0)
M7=open('motor7.txt','a+')
M7.truncate(0)
M8=open('motor8.txt','a+')
M8.truncate(0)

lst=[]
with open('krug.txt','r') as N:
    for line in N:
        vec=literal_eval(line)
        lst.append(vec)

podaci=pd.DataFrame(np.array(lst),columns=['f','v','i','p','CS'])
pd.set_option("display.max_rows",4000)
with open('podaci.txt','w') as P:
    P.write(str(podaci))

p1=podaci.loc[podaci['f']==1]
p1.to_excel(r'C:\Users\Klara\Desktop\Klara-
završni\SPREMLJENO\krug\motor1.xlsx',index=None, header=True)
p2=podaci.loc[podaci['f']==2]
p2.to_excel(r'C:\Users\Klara\Desktop\Klara-
završni\SPREMLJENO\krug\motor2.xlsx',index=None, header=True)
p3=podaci.loc[podaci['f']==3]
p3.to_excel(r'C:\Users\Klara\Desktop\Klara-
završni\SPREMLJENO\krug\motor3.xlsx',index=None, header=True)
p4=podaci.loc[podaci['f']==4]
p4.to_excel(r'C:\Users\Klara\Desktop\Klara-
završni\SPREMLJENO\krug\motor4.xlsx',index=None, header=True)
p5=podaci.loc[podaci['f']==5]
```

```
p5.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor5.xlsx',index=None, header=True)
p6=podaci.loc[podaci['f']==6]
p6.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor6.xlsx',index=None, header=True)
p7=podaci.loc[podaci['f']==7]
p7.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor7.xlsx',index=None, header=True)
p8=podaci.loc[podaci['f']==8]
p8.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\krug\motor8.xlsx',index=None, header=True)

with open('motor1.txt','a') as M1:
    M1.write(str(p1))
with open('motor2.txt','a') as M2:
    M2.write(str(p2))
with open('motor3.txt','a') as M3:
    M3.write(str(p3))
with open('motor4.txt','a') as M4:
    M4.write(str(p4))
with open('motor5.txt','a') as M5:
    M5.write(str(p5))
with open('motor6.txt','a') as M6:
    M6.write(str(p6))
with open('motor7.txt','a') as M7:
    M7.write(str(p7))
with open('motor8.txt','a') as M8:
    M8.write(str(p8))

##okretanje u mjestu

from ast import literal_eval
import pandas as pd
import numpy as np
```

```
import csv

M1=open('motor1.txt','a+')
M1.truncate(0)
M2=open('motor2.txt','a+')
M2.truncate(0)
M3=open('motor3.txt','a+')
M3.truncate(0)
M4=open('motor4.txt','a+')
M4.truncate(0)
M5=open('motor5.txt','a+')
M5.truncate(0)
M6=open('motor6.txt','a+')
M6.truncate(0)
M7=open('motor7.txt','a+')
M7.truncate(0)
M8=open('motor8.txt','a+')
M8.truncate(0)

lst=[]
with open('rotacija udesno.txt','r') as N:
    for line in N:
        vec=literal_eval(line)
        lst.append(vec)

podaci=pd.DataFrame(np.array(lst),columns=['f','v','i','p','CS'])
pd.set_option("display.max_rows",4000)
with open('podaci.txt','w') as P:
    P.write(str(podaci))

p1=podaci.loc[podaci['f']==1]
p1.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor1.xlsx',index=None, header=True)
p2=podaci.loc[podaci['f']==2]
```

```
p2.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor2.xlsx',index=None, header=True)
p3=podaci.loc[podaci['f']==3]
p3.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor3.xlsx',index=None, header=True)
p4=podaci.loc[podaci['f']==4]
p4.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor4.xlsx',index=None, header=True)
p5=podaci.loc[podaci['f']==5]
p5.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor5.xlsx',index=None, header=True)
p6=podaci.loc[podaci['f']==6]
p6.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor6.xlsx',index=None, header=True)
p7=podaci.loc[podaci['f']==7]
p7.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor7.xlsx',index=None, header=True)
p8=podaci.loc[podaci['f']==8]
p8.to_excel(r'C:\Users\Klara\Desktop\Klara-završni\SPREMLJENO\rotacija
udesno\motor8.xlsx',index=None, header=True)

with open('motor1.txt','a') as M1:
    M1.write(str(p1))
with open('motor2.txt','a') as M2:
    M2.write(str(p2))
with open('motor3.txt','a') as M3:
    M3.write(str(p3))
with open('motor4.txt','a') as M4:
    M4.write(str(p4))
with open('motor5.txt','a') as M5:
    M5.write(str(p5))
with open('motor6.txt','a') as M6:
    M6.write(str(p6))
with open('motor7.txt','a') as M7:
```

```
    M7.write(str(p7))
with open('motor8.txt','a') as M8:
    M8.write(str(p8))
```

MATLAB kod - grafovi

```
%vožnja unaprijed

figure(1)
x1=1:151;
x=x1/10;
plot(x,v,'r',x,i1,'c',x,p,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
title('Vožnja unaprijed (motor 1)')

figure(2)
x2=1:151;
xx=x2/10;
plot(xx,v1,'r',xx,i2,'c',xx,p1,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
title('Vožnja unaprijed (motor 2)')


%vožnja naprijed-nazad

figure(1)
x1=1:223;
x=x1/10;
plot(x,v,'r',x,i1,'c',x,p,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
title('Vožnja naprijed-nazad (motor 1)')

figure(2)
x2=1:223;
xx=x2/10;
plot(xx,v1,'r',xx,i2,'c',xx,p1,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
```

```
title('Vožnja naprijed-nazad (motor 4)')


%vožnja ukrug

x1=1:287;
x=x1/10;
plot(x,v,'r',x,i1,'c',x,p,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
title('Vožnja ukrug (motor 5)')


%okretanje u mjestu

figure(1)
x1=1:106;
x=x1/10;
plot(x,v,'r',x,i1,'b',x,p,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
title('Okretanje u mjestu (motor 1)')

figure(2)
x2=1:106;
xx=x2/10;
plot(xx,v1,'r',xx,i2,'b',xx,p1,'g')
legend('brzina','struja','pozicija','Location','southeast'
)
xlabel('Vrijeme [s]')
title('Okretanje u mjestu (motor 5)')
```