KLAIPĖDOS UNIVERSITETAS
JŪROS TECHNOLOGIJŲ IR GAMTOS MOKSLŲ FAKULTETAS
INFORMATIKOS IR STATISTIKOS KATEDRA

BRONIUS BERTAŠIUS

AUTOMATINĖ ELEKTRINIO TRAUKINIO GREIČIO KONTROLĖS SISTEMA MAŽINANTI ENERGIJOS SĄNAUDAS

Automatic speed control system for electric train designed to reduce energy consumption

Magistro baigiamasis darbas
Techninių informacinių sistemų inžinerijos studijų programa
valstybinis kodas – 621E15004

Klaipėda, 2016

MAGISTRO BAIGIAMOJO DARBO LYDRAŠTIS

Pildo magistro baigiamojo darbo autorius

Bronius Bertašius

(magistro baigiamojo darbo autoriaus vardas, pavardė)

Automatinė elektrinio traukinio greičio kontrolės sistema mažinanti energijos sąnaudas

(magistro baigiamojo darbo pavadinimas lietuvių kalba)

Patvirtinu, kad magistro baigiamasis darbas parašytas savarankiškai, nepažeidžiant kitiems asmenims priklausančių autorių teisių, visas baigiamasis magistro darbas ar jo dalis nebuvo panaudotas Klaipėdos universitete ir kitose aukštosiose mokyklose.

Bronius Bertašius

(magistro baigiamojo darbo autoriaus vardas, pavardė ir parašas)

Sutinku, kad magistro baigiamasis darbas būtų naudojamas neatlygintinai 5 m. Klaipėdos universiteto studijų procese.

Bronius Bertašius

(magistro baigiamojo darbo autoriaus vardas, pavardė ir parašas)

Pildo magistro baigiamojo darbo vadovas

Magistro baigiamąjį darbą ginti .....

(įrašyti – leidžiu arba neleidžiu)

doc. dr. Beatričė Andziulienė

(data) (magistro baigiamojo darbo vadovo vardas, pavardė ir parašas)

Pildo katedros, kuruojančios studijų programą, administratorius (sekretorius)

Baigiamasis darbas įregistruotas katedroje .....

(data)

Laima Brazdeikienė

(katedros sekretorės vardas, pavardė ir parašas)

Pildo katedros, kuruojančios studijų programą, vedėjas

Magistro baigiamąjį darbą ginti .....

(įrašyti – leidžiu arba neleidžiu)

..... prof. dr. Vitalijus Denisovas

(data) (katedros vedėjo vardas, pavardė ir parašas)

Recenzentu(-ais) skiriu .....

.....

(įrašyti recenzento(ų) vardą, pavardę)

..... prof. dr. Arūnas Andziulis

(data) (programos vadovas. vardas, pavardė ir parašas)

Bronius Bertašius

Automatinė elektrinio traukinio greičio kontrolės sistema mažinanti energijos sąnaudas: techninių informacinių sistemų inžinerijos magistro darbas. Vadovė doc. dr. B. Andziulienė. Klaipėdos universitetas, Jūros technologijų ir gamtos mokslų fakultetas, Informatikos ir statistikos katedra. Klaipėda, 2016. Darbo apimtis – 67 p. teksto , 40 paveikslas., 5 lentelės, 24 bibliografiniai šaltiniai, priedai – 16 p.

SANTRAUKA

Pasaulyje sparčiai besivystant elektra varomoms transporto priemonėms atsiranda vis daugiau problemų susijusių su jų energijos sąnaudomis. Labai didelė problema šioje srityje yra elektros energijos sukaupimas ir naudojimas. Siekiama, kad būtų galima elektra varoma transporto priemone nuvažiuoti kuo didesnį atstumą su kuo mažesnėmis energijos sąnaudomis. Tam reikalingi labai talpūs akumuliatoriai ir atitinkami valdymo algoritmai. Blogai parinkti valdymo algoritmai gali be reikalo ir greitai iššvaistyti akumuliatoriuose sukauptą energijos rezervą ir sumažinti nuvažiuojamą atstumą. Šiame darbe apžvelgiami labiausiai paplitę sistemų valdymo algoritmai, tokie kaip PID reguliatorius, greičio profiliai ir neraiškioji logika. Apžvelgus kiekvieno iš jų panaudojimo atvejus mokslinėje literatūroje nutarta elektrinio traukinio greičio kontrolės sistemą mažinančią energijos sąnaudas kurti remiantis greičio profiliais. Darbe taip pat nagrinėjami įtolintų sistemų valdymo būdai Bluetooth ryšio pagalba. Darbe aprašomas tiek techninės, tiek programinės sistemos kūrimo procesas. Sukurta sistema yra automatinė, o jos inicijavimas ir duomenų rinkimas vyksta nuotoliniu būdu. Atliekant sistemos patikrinimą eksperimentiniu būdu buvo stebimas sistemos elektros energijos suvartojimas, o po to palyginamas su prieš tai buvusios sistemos energijos suvartojimu. Atlikus palyginimą nustatyta, kad naujos sistemos veikimas užtikrina tolydų elektrinio traukinio modelio judėjimą ir mažesnes energijos sąnaudas.

PAGRINDINIAI ŽODŽIAI**:** elektrinio traukinio modelis, valdymo algoritmai, speed consumption greičio kontrolė, sistemos suvartojamos energijos mažinimas.

Bronius Bertašius

Automatic speed control system for electric train designed to reduce energy consumption: Master Thesis of Informatics. Supervisor: Assoc. prof. dr. B. Andziulienė. Klaipeda University, Faculty of Marine Technology and Natural Sciences, Department of Informatics and Statistics. Klaipeda, 2016. Work size - 69 p. text, 40 figures, 5 tables, 34 bibliographic sources, 24 annex –15 p.

SUMMARY

In the world there is rapid advance in electric vehicles and with the advance there are more and more problems associated with this industy. It is a big problem in this area with the electricity accumulation and use. The aim is to increase posibility for electric vehicles to travel the greaterst possible distance with the lowest possible energy consumption. This requires a high capacity batteries and adequate control algorithms. Bad selection of control algorithms may lead to unnecessarily wasted bateries' accumulated energy reserves and reduce the range of traveled distance. This paper provides an overview of the most common system control algorithms, such as PID controler, speed profiling and fuzzy logic. After review of each of these methods use cases in the scientific literature, it was decided to create automatic speed control system for electric train designed to reduce energy consumption on the basis of speed profiling. The paper also reviews wireless systems management techniques with Bluetooth. The paper describes both the hardware and the software system development processes. The developed system is automatic and the initiation and collection of data is only done remotely. The system has been experimentally tested in the power consumption and after that compared with the previous system power consumption. The comparison showed that the new system is capable to sutain fluent movements of the electric train model and can reduce energy consumption.

Keywords**:** electric train model, control algorithm, speed control, speed profiling, reducing system energy consumption.

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

SANTRUMPŲ IR TERMINŲ ŽODYNĖLIS

D i n a m i k a (*angl. dynamics*) – mechanikos dalis, kurioje nagrinėjamos kūnų judėjimo greičio kitimo priežastys.

ITCS (*angl. Intelectual Train Control System*) – intelektuali traukinių valdymo sistema.

K o m p i u t e r i n ė s i m u l i a c i j a (*angl. Computer simulation*) – realybės imitavimas virtualioje aplinkoje, kompiuterio programoje

K o n t a k t ų d i a g r a m a (L D) (*angl. ladder)* – tai kontaktų diagramų kalba yra grafinė programavimo kalba, kurioje naudojami elementai – tai įvairiose būsenose esantys kontaktai ir ritės.

P a r a m e t r i š k a i o p t i m i z u o j a m i r e g u l i a t o r i a i (angl. *Information design*) – tai tokie reguliatoriai, kurių algoritme yra parametrai, kuriuos reikia optimaliai parinkti konkrečiam valdymo objektui.

P I D r e g u l i a t o r i u s – tai parametriškai optimizuojamas reguliatorius, kurį sudaro trys pagrindinės komponentės: proporcinė (P), integruojanti (I) bei diferencijuojanti (D) grandys.

P L V - programuojamas loginis valdiklis

P L V i š ė j i m a s – *angl. PLC output* – tai PLV modulis prie kurio jungiami vykdymo įrenginiai.

P L V į ė j i m a s – *angl. PLC input* – tai PLV modulis prie kurio jungiami jutikliai.

P r o g r a m a – tai algoritmo išraiška tikslia kalba, suprantama skaičiavimo mašinai. Tai yra programinės įrangos pavyzdys. Dauguma programų yra sudarytos iš instrukcijų sekos, nusakančios jos darbą.

P r o g r a m i n ė į r a n g a (P Į) – *angl. software* – tai kompiuterio vykdomų instrukcijų seka, skirta tam tikriems veiksmams atlikti. Dažniausiai tokia įranga sukuriama naudojant programavimo kalbas, o vėliau kompiliuojant ar interpretuojant kodą.

N e r a i š k i o j i l o g i k a *(*angl *fuzzy logic)* – tai specifine logika grįstas valdymo metodas.

M o d e l i a v i m a s *(angl. modeling)* – tiriamojo objekto savybių pakartojimas kitame objekte (modelyje) norint geriau pažinti tiriamąjį objektą.

S ą s a j a – *angl. interface* – tam tikras susitarimas tarp dviejų programinių ar techninių įrenginių. Sąsajos yra skirtos abstrakčiai aprašyti apsikeitimą duomenimis tarp atitinkamų sistemos komponentų, kad vienam komponentui nereikėtų žinoti nieko daugiau apie kitą komponentą. Komponentu gali būti įrenginys, programos modulis, programa, klasės objektas.

V a l d i k l i s (P L V) – *angl. controller (PLC)* – tai mikroprocesorius ir jam priklausančių įrenginių sistema, skirta pramonės procesų automatizavimui.

V a l d y m o t i k s l a s – išėjimo signalas $y\,t(\,)$ turi, kiek galima tiksliau siekti norimo signalo $r\,t()$ trajektoriją.

TURINYS

ĮVADAS

Pasaulyje sparčiai besivystant elektra varomoms transporto priemonėms atsiranda vis daugiau problemų susijusių su jomis. Labai didelė problema šioje srityje yra elektros energijos sukaupimas ir naudojimas [1]. Siekiama, kad būtų galima elektra varoma transporto priemone nuvažiuoti kuo didesnį atstumą su kuo mažesnėmis energijos sąnaudomis. Tam reikalingi labai talpūs akumuliatoriai ir atitinkami valdymo algoritmai. Blogai parinkti valdymo algoritmai gali be reikalo ir greitai iššvaistyti akumuliatoriuose sukauptą energijos rezervą ir sumažinti nuvažiuojamą atstumą [1-2]. Net jei transporto priemonė neturi akumuliatorių, o yra bėginė, kaip elektrinis traukinys, gaunantis elektros energiją per bėgius ar laidus, būtina tobuliniti jo valdymo sistemą taip mažinant varikliams tenkančias apkrovas ir prailginant jų tarnavimo laiką. Taip pat verta stengtis ir dėl aplinkosaugos ir ekonominių tikslų. Šiuolaikiniame gyvenime didelė dalis transporte, pramonėje ir buityje naudojamų modernių įrenginių reikalauja, kad juose būtų įdiegta automatinė valdymo sistema, kuri ne tik efektyviai funkcionuotų, bet ir taupiai naudotų elektros energiją. Automatizuotose valdymo sistemose vienas iš pagrindinių uždavinių yra efektyvus sistemų valdymas, t.y. užsibrėžto tikslo siekimas kuo mažesnėmis sąnaudomis (pavyzdžiui, mažesnio elektros energijos ar medžiagų sunaudojimo), todėl labai svarbu tokioms valdymo sistemoms parinkti tinkamus reguliatorius ir suderinti jų parametrus, kad būtų pasiektas priimtinas tokių sistemų valdymas [3]. Universalaus reguliatoriaus, kuris tiktų visoms valdymo sistemoms, sukurti yra neįmanoma. Pirma, vienos sistemos yra tiesinės, kitoms būdingi netiesiškumai. Antra, valdymo sistemose, turinčiose keletą įėjimų ir keletą išėjimų, dažnai pasitaiko sąveikos tarp skirtingų valdomųjų kintamų, todėl reikia realizuoti papildomus modulius, tokioms sąveikoms įvertinti. Trečia, vienoms valdymo sistemoms nėra keliamas reikalavimas turėti labai tikslų sistemos valdymą, tuo tarpu kitoms - valdymas turi būti preciziškas [3]. Šiame darbe nagrinėjama elektrinio traukinio greičio kontrolės sistema, kurios tikslas naudojant tinkamai suderintus judėjimo parametrus sumažinti elektros energijos sąnaudas ir užtikrinti tolydų judėjimą.

Darbo tikslas – sukurti elektrinio traukinio greičio kontrolės sistemą, mažinančia energijos sąnaudas.

Siekiant pasiekti užsibrėžtą tikslą, suformuluoti šie **darbo uždaviniai**:

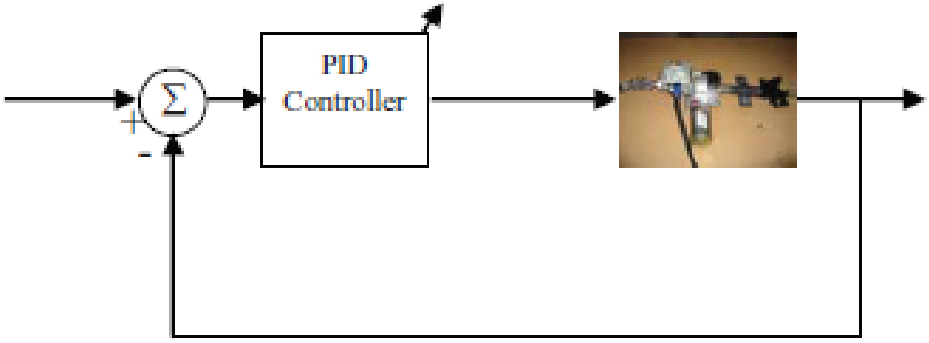
1. Atlikti dinaminių sistemų valdymo algoritmų analizę.
2. Atlikti įtolintų valdymo technologijų analizę.
3. Sukurti įtolinto valdymo greičio kontrolės sistemą.
4. Atlikti greičio kontrolės sistemos patikrą eksperimentiniu būdu.

1 DINAMINIŲ SISTEMŲ VALDYMO METODŲ ANALIZĖ

1.1 PID reguliatorius ir jo taikymas dinaminių sistemų valdyme

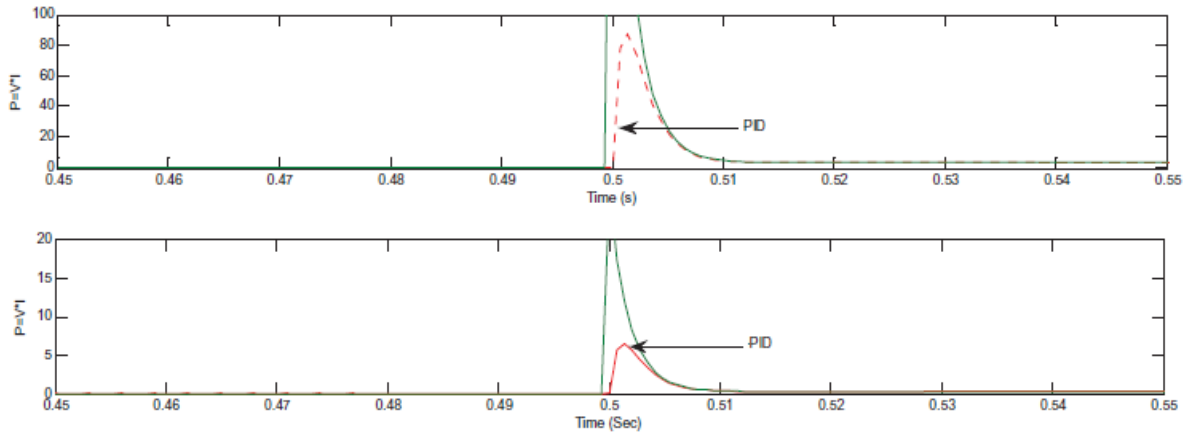
PID reguliatorius – tai vienas plačiausiai šiuolaikinėse valdymo sistemose naudojamų reguliatorių. Šis reguliatorius bando mažinti (koreguoti) paklaidą tarp išmatuoto proceso kintamojo (esamo lygio) ir siekiamo (norimo) lygio, apskaičiuodamas kitai iteracijai reikalingą valdymo signalą, kad esamo lygio reikšmės artėtų prie norimo lygio reikšmių [1].

Mokslininkai M.K Hassan, N.A.M. Azubir, Nizam H.M.I, S.F Toha ir B.S.K.K Ibrahim atliko tyrimus ir pasiūlė vieną iš būdų sumažinti energijos sąnaudas moderniuose automobiliuose. Jie pasiūlė atsisakyti hidraulinės vairo sistemos ir sukūrė elektrinę vairo pasukimo sistemą naudodami PID reguliatorių [4] Kadangi sistema elektrinė, nebereikia naudoti energijos hidraulinės pompos veikimui, kad būtų nuolatos palaikomas pakankamas hidraulinio skysčio spaudimas. Hidraulinę sistemą pakeitus elektrine, energija būtų naudojama tik tada, kai vairas sukamas, o ne nuolatos, kaip yra būtina esant hidraulinei sistemai [4]. Optimaliam siūlomos sistemos veikimui užtikrinti mokslininkai į elektrinę vairo pasukimo sistemą įdiegia PID reguliatorių, kad dar labiau sumažintų energijos sąnaudas. Principinė šios sistemos schema pavaizduota 1 paveiksle.



1 pav. Elektrinė vairo sukimo sistema [4]

Įdiegę PID reguliatorių mokslininkai suderino parametrus, taip, kad automobiliui važiuojant greitai sukimui būtų naudojama daugiau energijos, o važiuojant lėtai – mažiau energijos. Kai kūrimo procesas buvo baigtas, mokslininkai atliko tyrimus ir išmatavo kiek energijos sunaudoja jų sistema (2. pav).



2 pav. Elektrinės vairo sukimo sistemos energijos sąnaudos su PID valdikliu ir be jo [4]

Kaip matome 2 paveiksle, matavimai buvo atliekami kai sistema dirbo dideliu (viršutinis grafikas) ir mažu (apatinis grafikas) pajėgumais. Tiek vienu tiek kitu atveju vairo sukimo sistema sunaudojo mažiau energijos veikdama su PID reguliatoriumi.

Kita mokslininkų komanda kūrė sistemą, kuri padėtų sumažinti energijos sąnaudas anglimi kūrenamos elektros jėgainės ventiliatorių sistemoje. Mokslininkai Yuelan Wang, Zengyi Ma, Yueliang Shen, Yijun Tang, Mingjiang Ni, Yong Chi, Jianhua Ya, Kefa Cen savo kūriamoje sistemoje taip pat naudojo PID reguliatorių. Jo paskirtis buvo palaikyti pastovų oro srautą, kad būtų palaikomas degimo procesas. Senoje sistemoje ventiliatoriai nuolatos pūsdavo orą į degimo kameras, o naudojant PID reguliatorių buvo matuojamas ir reguliuojamas oro srautas, todėl ventiliatorių sistemai nebereikėjo nuolatos suktis pilu greičiu. Atlikę matavimus mokslininkai nustatė, kad naudodami PID reguliatorių gali sutaupyti iki 15 % energijos, sunaudojamos ventiliatorių sistemos veikimui [4].

Todėl galima daryti išvadą, kad PID reguliatoriaus integravimas sistemoje yra tinkamas būdas mažinti jos energijos sąnaudas [1-6].

1.1.1 PID reguliatoriaus veikimo principai

Klasikinis PID reguliatorius sudarytas iš 3 grandžių: proporcinės (P), integruojančios (I) ir diferencijuojančios (D).

Tarkime, kad:

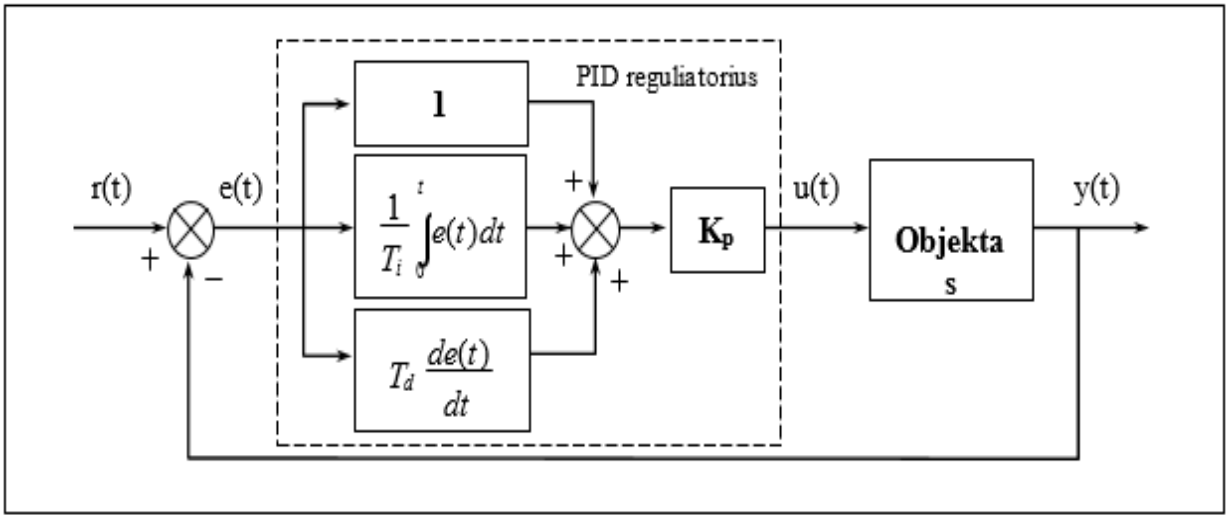
- $u\ t(\ )$- valdymo signalas;
- $r\ t(\ )$- užduoties (norimas) signalas;
- $y\ t(\ )$ - išėjimo (esamas) signalas;

- $e\,t(\,) = r\,t(\,) - y\,t(\,)$- valdymo paklaida;
- $K = P$ - proporcinės grandies koeficientas;
- T_I – integruojančios grandies laiko pastovioji;
- T_D – diferencijuojančios grandies laiko pastovioji;
- T_o - diskretizavimo žingsnis

Tuomet tolydinio PID reguliatoriaus išraiška yra užrašoma taip:

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau) + T_D\frac{de(t)}{dt}\right] \quad (1),$$

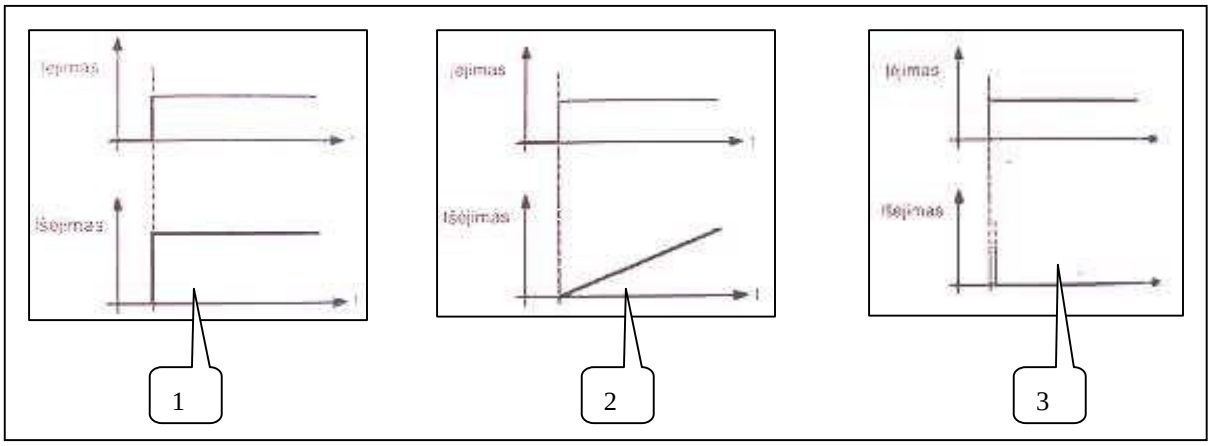
o objekto su PID reguliatoriumi valdymo sistema pavaizduota 3 paveiksle.



3 pav. Valdymo sistemos su PID reguliatoriumi schema [2]

Geresniam PID reguliatoriaus veiklos supratimui nagrinėjamos atskiros PID reguliatoriaus komponentės: P, I ir D reguliatoriai. Proporciniame reguliatoriuje (4 pav. 1) valdančiojo kintamojo išėjimas yra proporcingas sistemos nuokrypiui. Jei sistemos nuokrypis yra didelis, tai valdančiojo kintamojo vertė taip pat yra didelė. Jei sistemos nuokrypis yra mažas, tai ir valdančiojo kintamojo vertė yra nedidelė. Kadangi, valdantysis kintamasis yra proporcingas sistemos nuokrypiui, tai valdantysis kintamasis bus tik tada, jei bus sistemos nuokrypis. Dėl šios priežasties vien proporciniu valdikliu negalima pasiekti, kad sistemos nuokrypis būtų lygus 0. Tokiu atveju, nebus valdančiojo kintamojo, o tuo pačiu ir valdymo [1-2]. Integruojančio veikimo reguliatorius (4 pav. 2) atlieka sistemos nuokrypio sudėties veiksmą laikui bėgant, t. y. integruoja jį. Jei sistemos nuokrypis yra pastovus, tai valdančiojo kintamojo reikšmė pastoviai didės, kadangi ji priklauso nuo sumos, kuri bėgant laikui didėja. Tačiau didėjant valdančiojo kintamojo vertei, sistemos nuokrypio vertė mažėja.

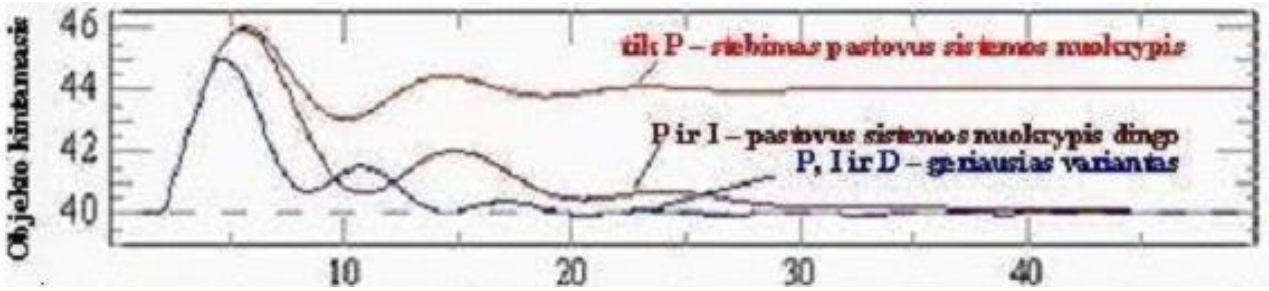
Toks procesas tęsiasi iki tol, kol sistemos nuokrypis pasidaro lygus 0. Integruojančio veiksmo reguliatoriai yra taikomi, kai norima išvengti pastovių sistemos nuokrypių. Diferencijuojanti grandis (2 pav. 4) įvertina, kokiu greičiu kinta sistemos nuokrypis. Jei sistemos nuokrypis kinta greitai, tai valdančiojo kintamojo vertė yra didelė [2]. Jei sistemos nuokrypis kinta lėtai, tai valdančiojo kintamojo vertė yra maža. Reguliatorius, kuris turi tik diferencijuojančią grandį, neturi prasmės, kadangi valdantysis kintamasis turės vertę tik tada, kai pasikeis sistemos nuokrypis.



4 pav. Reguliatorių P, I, D veikimo schemos [2].

Apibendrinus gauname, kad proporcinė grandis nusako valdymo signalo *ut*( ) proporcinę priklausomybę nuo valdymo paklaidos *et*( ), integruojanti grandis – priklausomybę nuo paklaidos *e t*( ) kaupimosi, diferencijuojanti grandis – priklausomybę nuo paklaidos *et*( ) kitimo greičio. Užtenka parinkti šių priklausomybių koeficientus konkrečiam objektui taip, kad būtų pasiektas valdymo tikslas su galimai mažesne valdymo paklaida *et*( )[2].

Bendruoju atveju kaip keičiasi valdymo paklaida bei valdymo signalas naudojant atskiras PID reguliatoriaus grandis ir visas kartu pavaizduota (3 pav.).



5 pav. Sistemos išėjimo signalai su skirtingomis PID reguliatoriaus grandimis [3].

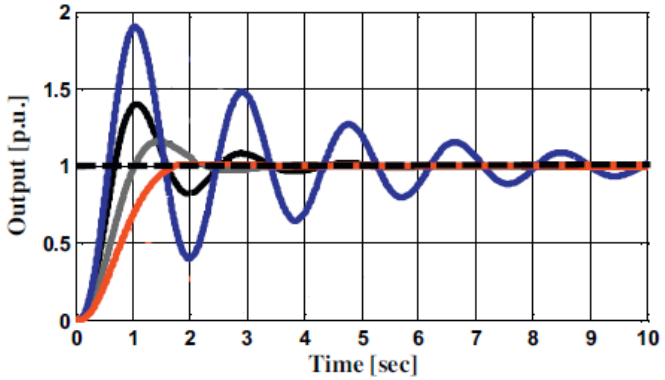
Kalbant apie PID reguliatorius, verta paminėti tokias pagrindines detales apie atskiras grandines:

jei proporcinės grandies (P) parametras yra parenkamas didelis, tai gali privesti prie sistemos nestabilumo, tuo pačiu jei šis parametras yra per mažas, tuomet gaunami per dideli nuokrypiai ir toks reguliatorius yra nepakankamai jautrus: labai keičiantis valdymo nuokrypiams – per mažai įtakojamas valdymo kintamasis, tuo būdu gaunamas prastas sistemos valdymas. Tiek integruojanti grandis (I), tiek proporcinė (P) gali sąlygoti vadinamojo peršokimo atsiradimą, kai objekto kintamojo reikšmė viršijusi norimą reikšmę, vėliau krenta žemiau norimos reikšmės ir tik tada vėl pradeda artėti prie jos (daromas svyravimas aplink norimą reikšmę). Integruojančios grandies nebuvimas gali sąlygoti ir tai, kad valdymo sistema niekada nepasieks savo norimo lygio reikšmės. Diferencijuojanti grandis (D) sulėtina reguliatoriaus išėjimo (valdymo kintamojo) pokytį, tai ypatingai jaučiama, kai objekto kintamojo reikšmės yra pakankamai arti norimo lygio reikšmių, tai naudojama integruojančios grandies (I) sugeneruojam peršokimui sumažinti bei padidinti sistemos stabilumui. Diferencijuojanti grandis stiprina triukšmus ir jei valdomas objektas yra jautrus triukšmams, tuomet tai taip pat gali išprovokuoti sistemos nestabilumą [1-6].

PID reguliatorius yra patikimas ir paprastas. Jis plačiai naudojami pramonėje, medicinoje bei transporto sistemose. Šis reguliatorius turi pakankamai lankstumo duoti puikių rezultatų įvairiose situacijose, o tai yra viena iš pagrindinių priežasčių dėl ko jis vis dar plačiai naudojamas. PID reguliatorius bei jo modifikacijos gerai tinka, kai nagrinėjami objektai yra tiesinės sistemos, kai galima sukonstruoti objektų matematinius modelius, tačiau kai susiduriama su netiesinėmis sistemomis, tuomet šie reguliatoriai ne visuomet tinkamai valdo objektus.

1.2 Greičio profilių panaudojimas objektų valdyme

Mokslininkas Mohamed S. Zaky savo moksliniame darbe, priešingai nei ankstesniame skyriuje minėti mokslininkai teigia, kad tiksliam sistemos valdymui neužtenka klasikinio PID reguliatoriaus. Jis daugiau dėmesio skyrė vieno variklio greičio valdymui. Šis mokslininkas teigia, kad tiksliam variklio greičio valdymui galima panaudoti greičio profilius. Parinkus reikiamus parametrus greičio profilius galima suderinti taip, kad jų valdomas variklis pasiektų norimą greitį per nustatytą laiko intervalą, ir tai padarytų geriau nei variklis valdomas PID valdikliu [7]. Palyginimui buvo atlikti matavimai ir gauti duomenys atvaizduoti grafiškai (6 pav.).

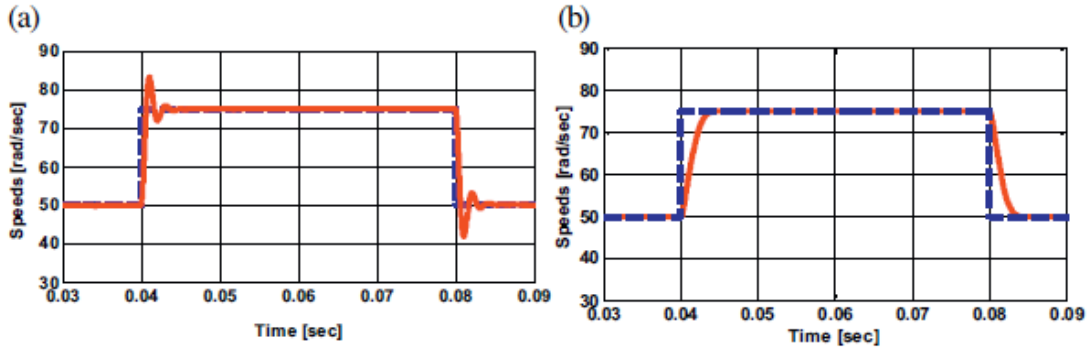


6 pav. PID ir S-kreivės profilio palyginimo grafikas [7]

6 paveiksle pavaizduota:

- mėlyna kreivė vaizduoja PID valdiklio, suderinto pagal Ziegler-Nichols metodą veikimas;
- pilka kreive pavaizduotas PID valdiklio suderinto pagal Root-Locus metodą veikimas;
- Juoda spalva pavaizduotas automatiškai save reguliuojantis (*angl. self-tuning*) PID valdiklio veikimas;
- Oranžine spalva pavaizduota, kaip nustatytą greitį pasiekia variklis valdomas s-kreivės profiliu.

Taigi matome, kad norint sumažinti variklio suvartojamos energijos kiekį, galima naudoti ne tik PID valdikį, bet ir tinkamai suderintą greičio profilį, kuris atitinkamose situacijose veikia geriau ir užtikrina tikslesnį variklio greičio valdymą.

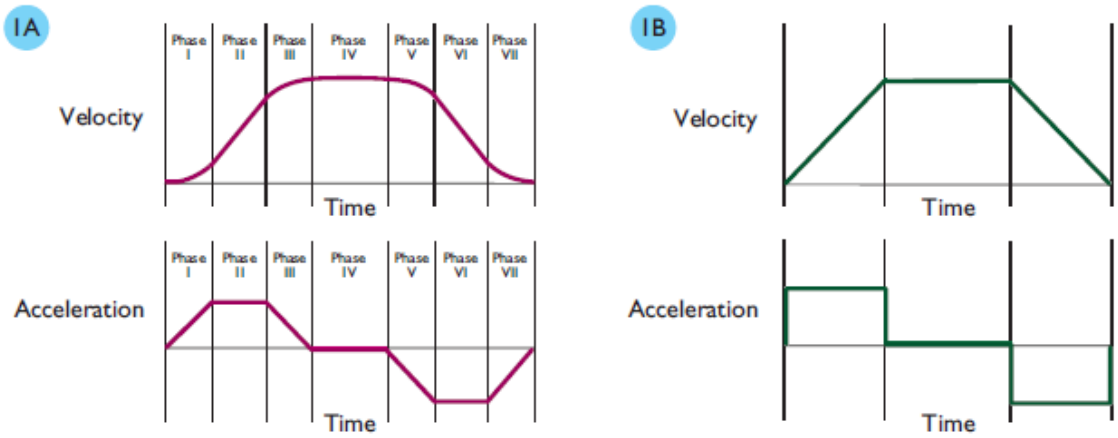


7 pav. Staigus greičio pasikeitimas: (a) – PID valdiklis, (b) – S-kreivės profilis

7 paveiksle pavaizduota kaip PID valdiklis ir greičio profilis susidoroja su staigiu greičio pokyčiu. Greičio profiliu valdomas variklio sukimosi greitis užtikrina, kad bus išvengta nereikalingų svyravimų, kurie atsiranda, kai variklis valdomas PID valdikliu. Taip išvengta papildomų energijos sąnaudų [7].

1.2.1 Greičio profiliai ir jų veikimo principai

Yra daug skirtingų greičio profilių, naudojamų medicinos įrangoje, moksliniuose instrumentuose, pramonėje, transporto sistemose ir kitose srityse, tačiau populiariausi ir labiausiai paplitę yra S-kreivės ir trapeciniai profiliai. Jie pavaizduoti 8 paveiksle.



8 pav. S-kreivės ir trapecinis greičio profiliai ir jų fazės [8]

Kaip matome iš paveikslo klasikinis S-kreivės greičio profilis (paveiksle pažymėtas 1A) susideda iš 7 atskirų judėjimo etapų. Pirmame etape variklis pajudinamas iš rimties būsenos, o sukimosi pagreitis tiesiškai didinamas tol, kol pasiekiama maksimali akceliaracija. Antrame etape akceliaracijos reikšmė yra išlaikoma maksimali tol, kol pasiekia tokį tašką, kai turi pradėti mažėti, nes priartėjama prie maksimalaus greičio. Trečiame etape, kai pasiekiamas maksimalus greitis, akceliaracija tiesiškai mažinama, kol jos reikšmė pasiekia 0. Ketvirtame etape akceliaracija lygi 0, todėl greitis yra išlaikomas pastovus. Jis išlaikomas pastovus iki tol, kol pradedamas lėtinimas, tada viskas vyksta būdu, simetrišku pirmam, antram ir trečiam etapams tol, kol greitis tampa lygus 0 [8].

Na o trapecinis greičio profilis (paveiksle pažymėtas 1B) susideda tik iš 3 etapų. Jis yra S-kreivės greičio profilio poaibis, turintis etapus atitinkančius šiuos S-kreivės etapus: antrą (tolygus greitėjimas), ketvirtą (pastovus greitis) ir šeštą (pastovus lėtėjimas). Šis sumažintas etapų skaičius pabrėžia S-kreivės greičio profilio ir trapecinio greičio profilio skirtumus. S-kreivės profilis turi papildomus judėjimo etapus, perėjimus tarp akceliaracijos reikšmės pakitimų. Trapecinis profilis tarp šių pakitimų turi momentinius perėjimus. Tai galima pamatyti ir 8 paveiksle, kur pavaizduoti akceliaracijos grafikai [8].

Judėjimo charakteristika kuri apibrėžia akceliaracijos pokytį arba perėjimo periodą vadinama timptelėjimu (*angl. jerk*). Šis timptelėjimas yra apibrėžiamas kaip akceliaracijos pakitimas laiko atžvilgiu. Trapecinio profilio atveju, timptelėjimas (akceleracijos pokytis) yra didelis ir nepastovus, o

S-kreivės profilio atveju jo vertė yra pastovi ir paskirstyta per tam tikrą laiko periodą. Jo dėka S-kreivės profilio valdomas judėjimas gali būti tolydus, nes kuo didesnė yra timptelėjimo vertė, tuo daugiau nepageidaujamų vibracijų yra sukelta ir platesnis jų dažnių spektras [9].

Kadangi S-kreivės profilis turi 7 atskirus judėjimo etapus, matematiškai jį aprašyti yra sudėtingiau nei trapecinius greičio profilius. Yra gana sudėtinga apskaičiuoti profilio valdomo objekto sustojimo tašką. Kai kuriose sistemose, kuriose yra įdiegtas objektų valdymas S-profiliais yra draudžiami pakeitimai prasidėjus profilio veikimui, taip pat nepageidaujami yra ir asimetriniai profiliai [9]. S-kreivės profilis gali būti nepertraukiamos (*angl. continuous*) formos ir diskretaus laiko (*angl. discrete time*) formos. Nepertraukiama forma užrašoma taip:

$$P_T = P_0 + V_0T + 1/2A_0T^2 + 1/6JT^3 \quad (2)$$

$$V_T = V_0 + A_0T + 1/2JT^2 \quad (3)$$

$$A_T = A_0 + JT \quad (4)$$

o diskretaus laiko forma užrašoma taip:

$$P_T = P_T + V_T + 1/2A_T + 1/6J \quad (5)$$

$$V_T = V_T + A_T + 1/2JT \quad (6)$$

$$A_T = A_T + JT \quad (7)$$

kur:

- P_0, V_0 ir A_0 yra atitinkamai pradinė pozicija, greitis ir akseliaracija.
- P_T, V_T ir A_T tra atitinkamai pozicija, greitis ir akseliaracija laiko momentu T.
- J yra profilio timptelėjimas (pagreičio pokyčio laikas).

Taigi tinkamai pasirinkus greičio profilį ir jo parametrus galima užtikrini tolydesnį judėjimą, sumažinti dėvėjimąsi ir pagerinti plataus spektro sistemų operacijų laiką. Trapeciniai profiliai gali būti naudingi, bet juos riboja tai, jog negali būti apibrėžti perėjimai tarp akceliaracijos pokyčių. Šia problemą išsprendžia S-kreivės profiliai, tačiau atitinkamai yra sudėtingesni matematiškai [8-9].

1.3 Neraiškiosios logikos reguliatorius

Neraiškioji logika (angl. *fuzzy logic*) – tai sprendinio radimo (angl. *problem-solving*) valdymo sistemos metodologija, kuri suteikia galimybę realizuoti sistemos valdymą didelėms sistemoms. Ji gali būti realizuota įvairiai: techninėje, programinėje įrangose, o taip pat naudojant kombinuotą, abiejų įrangų panaudojimą . Neraiškioji logika labai tinka tokių sprendimų priėmimui, kur reiškiniai ar procesai yra neaiškūs, dviprasmiški, netikslūs ar paprasčiausiai trūksta informacijos apie juos [10].

Neraiškioji logika apjungia paprastą taisyklių „*JEIGU* X *IR* Y *TUOMET Z*“ principu pagrįstą sprendinių radimą, priešingai nei tradiciniuose būduose, kuriuose naudojami sistemų matematiniai modeliai. „Fuzzy“ logika yra pagrįsta empiriniu modeliavimu, daugiau priklausanti nuo operatoriaus patirties nei nuo sistemos techninio supratimo. Pavyzdžiui, vietoj terminų apibūdinančių temperatūros valdymą: „SP=500F“, „T<1000F“ arba „$210^{\circ}C < Temp < 220^{\circ}C$ “, naudoti terminus:

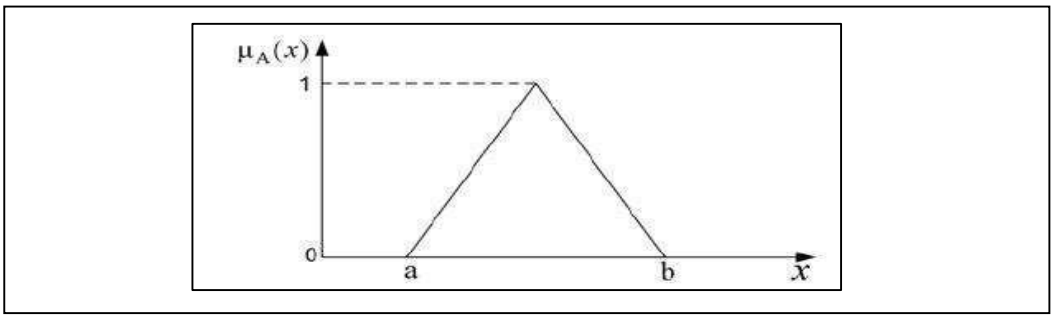
„JEIGU „temperatūra“ yra „žema“ IR „temperatūros pokytis“ yra „mažėja“ TUOMET „termoreguliatoriaus veiksmas“ yra „šildyti“ arba „JEIGU „temperatūra“ yra „aukšta“ IR „temperatūros pokytis“ yra „didėja“ TUOMET „termoreguliatoriaus veiksmas“ yra „vėsinti“.

Šie terminai nėra tikslūs, bet labai išsamūs (angl. descriptive) ir suprantamai nusako, kas turi iš tikrųjų vykti [10].

Vieni iš svarbiausių neraiškios logikos elementų yra lingvistiniai kintamieji, kurie yra suformuojami iš eilės persidengiančių „fuzzy“ aibių, fizikine prasme apibūdinančių tą kintamąjį. Kiekvienas lingvistinis kintamasis yra sudarytas iš baigtinio skaičiaus „fuzzy“ aibių, kurios persidengdamos pilnai padengia atitinkamo kintamojo visų galimų reikšmių sritį. Tradicinėje aibių teorijoje elementas gali arba priklausyti konkrečiai aibei, arba ne. Tuo tarpu neraiškioje logikoje leidžiama elementui ir iš dalies priklausyti aibei, suteikiant jam atitinkamą priklausomumo koeficientą. Elemento priklausomybė konkrečiai aibei yra išreiškiama priklausomumo funkcija μ( ), kuri gali įgyti reikšmes iš intervalo [0,1]. 0 reiškia, visišką nepriklausomybę, o 1 – visišką priklausomybę, o tarpinės reikšmės – dalinę priklausomybę. „Fuzzy“ aibė yra sutvarkytų porų aibė

$$A = \{(x, \mu_A(\,))x \mid x \in X\}, \qquad (8)$$

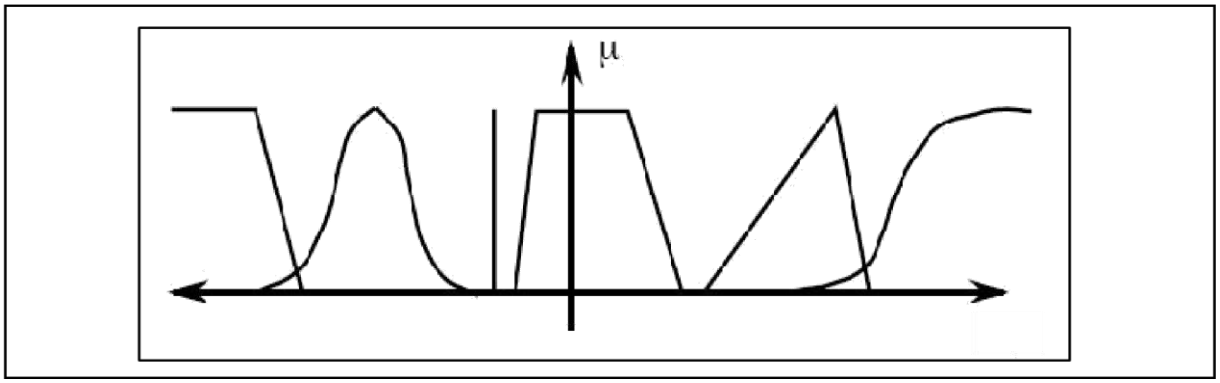
kur x yra aibės X elementas, o $\mu_A()$ yra funkcija, nusakanti elemento x priklausomumą aibei X, kitaip dar vadinama „fuzzy“ aibės paviršiumi [10-11].



9 pav. Priklausomybės aibei A funkcija [11]

Kiekviena „fuzzy“ aibė turi atskirą, jos paviršių aprašančią t.y. elementų priklausomumo aibei dydžius nustatančią funkciją μ() , kuri parenkama priklausomai nuo norimo tikslumo, sistemos

jautrumo įėjimo signalams ar paprastumo. „Fuzzy“ aibės paviršiai gali būti trikampio, trapecijos, S raidės, varpo ir kitokių formų (8 pav.).

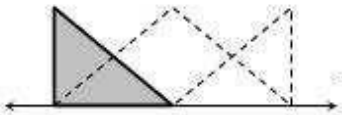
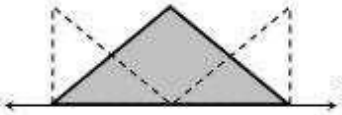
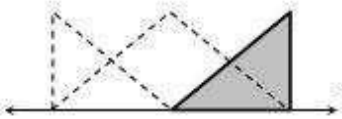


10 pav. Tipinių „fuzzy“ aibės paviršių pavyzdžiai [11].

Dažniausiai yra naudojamas trikampio formos paviršius, dėl skaičiavimo paprastumo. Tokį paviršių apsprendžia trys taškai: kairys pagrindo taškas, centras ir dešinys pagrindo taškas. Įėjimo erdvės intervale (a, b) priklausomybės funkcija įgauna nenulinę vertę.

Trikampio formos priklausomumo funkcijos yra apskaičiuojamos pagal 1 lentelėje pateiktas formules, kur c – trikampio centras, o kraštinių trikampių atveju (žr. kairys, dešinys kraštas) – riba, už kurios priklausomumo funkcija visuomet yra lygi 1, w žymi trikampio pagrindo plotį.

1 lentelė. Trikampio formos priklausomumo funkcijų apskaičiavimo formulės [11]

Kairys kraštas		$\mu(u)=\begin{cases}1 & \text{jeigu } u \le c \\ \max\left\{0,1+\frac{c-u}{0.5w}\right\} & \text{kitu atveju}\end{cases}$
Centrinė dalis		$\mu(u)=\begin{cases}\max\left\{0,1+\frac{u-c}{0.5w}\right\} & \text{jeigu } u \le c \\ \max\left\{0,1+\frac{c-u}{0.5w}\right\} & \text{kitu atveju}\end{cases}$
Dešinys kraštas		$\mu(u)=\begin{cases}\max\left\{0,1+\frac{u-c}{0.5w}\right\} & \text{jeigu } u \le c \\ 1 & \text{kitu atveju}\end{cases}$

„Fuzzy“ aibės turi tik joms būdingas charakteristikas, apsprendžiančias priklausomumo funkcijų vertikalius ir horizontalius matmenis ir formą:

1. Normali „fuzzy“ aibė: tai aibė, kurios priklausomumo funkcija yra lygi vienetui nors viename visų galimų reikšmių intervalo taške ir nors viename ji lygi nuliui. Jeigu priklausomumo

funkcija nei viename visų galimų reikšmių intervalo taške nelygi vienetui, ji yra normuojama, t.y. priklausomumo funkcijos paviršius yra praplečiamas taip, kad funkcija nors viename taške įgytų reikšmę, lygią vienetui.

2. „Fuzzy“ aibės arba priklausomumo funkcijos aukštis: tai „fuzzy“ aibės priklausomumo funkcijos maksimali reikšmė.
3. α riba: tai riba, žemiau kurios visos „fuzzy“ aibės priklausomumo funkcijos reikšmės priskiriamos nuliui, ši riba veikia kaip filtras, pašalinantis labai mažų svorių priklausomumo funkcijų reikšmes:

$$\mu(u) = \begin{cases} \mu(u) & jeigu \mu(u) \geq \alpha \\ 0 & kitu\_atveju \end{cases}, \tag{9}$$

Neraiški logika pasižymi tokiomis unikaliomis savybėmis, kurios suteikia nemažai privalumų sistemų valdyme:

- Naudojant neraiškią logiką, įėjimo signalai gali būti netikslūs, triukšmingi. Valdymo signalas - tolygi (angl. smooth) valdymo funkcija, nors ir naudojamos įvairių lingvistinių geriau zymeti ,negu numeruoti kintamųjų variacijos.
- Neraiškios logikos reguliatoriai manipuliuoja vartotojo apibrėžtomis taisyklėmis, sistemų valdyme. Toks valdymo būdas leidžia nesunkiai modifikuoti ir konfigūruoti patį sistemos valdymą, tokiu būdu gerinant sistemos valdymo kokybę. Nauji daviklių duomenys yra lengvai prijungiami prie sistemos: sukuriant papildomas taisykles.
- Kadangi neraiškios logikos reguliatoriai atlieka veiksmus su apibrėžtomis taisyklėmis, gana nemažas įėjimo signalų skaičius gali būti apdorojimas, o taip pat ir išėjimo signalų (1-4 ir daugiau) gali būti sugeneruota, nors, tokiu atveju, taisyklių bazė greitai tampa sudėtinga. Tada, rekomenduojama skaldyti visą valdymo sistemą į mažesnes dalis ir naudoti kelis mažesnės apimties neraiškios logikos reguliatorius.
- Neraiškios logikos reguliatorių pagalba galima valdyti netiesines, inertiškas sistemas, kurias kitais metodais valdyti yra sunku arba iš vis neįmanoma [10-13].

1.4 Dinaminių sistemų valdymo metodų apibendrintas palyginimas

PID reguliatorius puikiai tinka valdyti sistemoms, kuriose judėjimas yra tiesinis ir nesudėtingas. Šį reguliatorius gana paprasta derinti ir pritaikyti individualių sistemų valdymui, mat reikia derinti tik 3 parametrus.

Tinkamai parinktas greičio profilis ir jo parametrai galima užtikrini tolydesnį judėjimą, sumažinti dėvėjimąsi ir pagerinti plataus spektro sistemų operacijų laiką.

Neraiškiosios logikos kontrolerių pranašumas yra tas, kad jais galima valdyti sistemas, kuriose judėjimas yra netiesinis. Tačiau sėkmingam jų veikimai reikia suderinti labai daug parametrų ir apibrėžti daug logikos taisyklių

Tam, kad būtų paprasčiau išsirinkti valdymo metodą, PID reguliatorius, greičio profiliai ir neraiškiosios logikos reguliatorius palyginami pliusų ir minusų lentelėje(2 lentelė).

2 lentelė. Valdymo metodų vertinimas

	Derinimas	Tikslumas	Sudėtingumas
PID reguliatorius	+	-	-
Greičio profiliai	+	+	+
Neraiškioji logika	-	+	-

Kaip pavaizduota lentelėje, PID reguliatoriaus derinimas yra palyginus paprastas, nes yra tik 3 derinami parametrai, tačiau visiškam tikslumui pasiekti reikalingas ilgas ir sudėtingas derinimo procesas. Greičio profiliai yra lengvai derinami ir gana tikslūs, o jų parametrų parinkimas nesudėtingas. Neraiškiosios logikos reguliatoriaus derinimo procesas yra ilgas, nes reikia įvertinti daug parametrų. Ir nors šio valdiklio dėka galima pasiekti didelį valdomos sistemos tikslumą, reikia daug pastangų, kad jis veiktų tinkamai. Taigi apibendrinus šiuos aspektus nutarta tolimesniam elektrinio traukinio greičio kontrolės sistemos kūrimui naudoti greičio profilius.

1.5 Įtolinto valdymo sistemų analizė

Pastaraisiais metais pasaulyje populiarėja įtolinto valdymo sistemos grįstos bevielio ryšio technologijomis. Tokios sistemos lengvai integruojamos į jau esančias sistemas ir suteikia atnaujintai sistemai lankstumo [14]. Visa tai padidina sistemos efektyvumą. Šios sistemos vis dažniau yra

taikomos pramonėje, kur sudėtingos darbo sąlygos – elektrinis triukšmas, mechaninės vibracijos ar didelis greitis.

Mokslininkai Ramazan Bayindir ir Yucel Cetinceviz nagrinėjo problemas susijusias su sistemos valdymu kai pagrindinis valdomas objektas yra nutolęs. Jie nagrinėjo bevielio ryšio panaudojimo galimybę naudojant programuojamus loginius valdiklius, išskirstytus įvedimo/išvedimo ir specifinius bevielio ryšio modulius. Indijos mokslininkai sprendė uždavinius susijusius su „Bluetooth“ ryšio panaudojimu valdant įvairius mobilius objektus realiu laiku. Jie taip pat nagrinėjo problemas, kurias sukelia sudėtinga aplinką: aukšta temperatūra, dulkės, mechaninio pažeidimo pavojus [15].

1.5.1 Įtolinto valdymo sistemų pagrindinė problematika

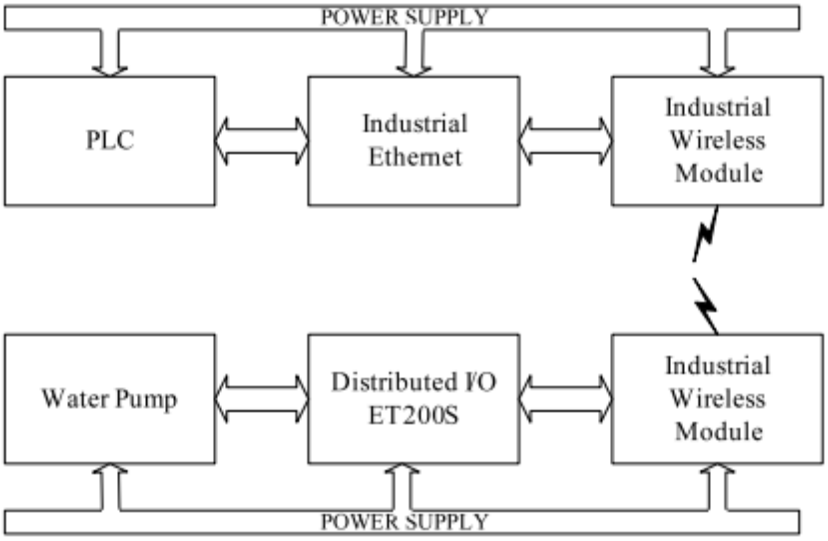
Pagrindinė įtolinto valdymo sistemų problematika yra patikimumo užtikrinimas perduodant duomenis bevieliu ryšiu. Kadangi tokios sistemos veikia sudėtingose aplinkose, kuriuose yra daug triukšmų, įtakojančių patikimą duomenų bei valdymo signalų perdavimą, pagrindinis uždavinys kuriant įtolinto valdymo sistemas yra ryšio technologijos parinkimas, kuri tenkintų sistemai keliamus reikalavimus.

Bevielio ryšio technologijų yra daugybė: GSM, GPS, RFID, WLAN, Bluetooth, skaitmeninė antžeminė technologija (DVB-T), palydovinė televizija, IR ir daugelis kitų [8]. Kiekviena technologija turi savo taikymo ypatumus, apribojimus, panaudojimo sritis, todėl diegiant bevielį ryšį į įtolinto valdymo sistemą, reikia atsižvelgti į duomenų perdavimui keliamus reikalavimus ir parinkti bevielio ryšio technologiją konkrečiai problemai išspręsti.

1.5.2 Bevielis ryšys tarp PLV ir valdomo įrenginio

Mokslininkai Ramazan Bayindir ir Yucel Cetinceviz iš Turkijos aprašo vandens siurblio valdymo sistemą, kuri suprojektuota ir pritaikyta realioms veikiančioms sistemoms, ir sukurtą prototipą, kuris naudojamas eksperimentams laboratorijoje bei demonstracijai. Tokios sistemos veikia potencialiai pavojingose aplinkose, kur chemikalai, vibracijos, judančios dalys gali pažeisti ryšio su valdymo įranga laidus. Ši valdymo sistema naudoja programuojamą loginį valdiklį (PLV) ir pramoninę bevielę vietinio tinklo technologiją (*industrial wireless local area network - IWLAN*). Sistemą sudaro PLV, komunikacijos procesorius (*communication processor - CP*), du IWLAN moduliai, paskirstyti įvedimo/išvedimo moduliai, vandens siurblys ir jutikliai. Šios sistemos komunikacija yra pagrįsta pramoniniu „Ethernet“ ryšiu ir naudoja standartizuotą TCP/IP protokolą parametrų stebėjimui, konfigūracijai ir diagnostikai.

Visa ši sistema susideda iš 2 pagrindinių modulių (11 pav.). Pirmasis yra – valdantysis (*angl. master*) modulis, kuris susideda iš 3 dalių: PLV, komunikacijos procesoriaus ir bevielio tinklo modulio. Trečiasis yra pavaldusis (*angl. slave*) modulis, kuris jungiamas prie valdomo įrenginio. Šiuo atveju vandens siurblio[16].



11 pav. Vandens siurblio valdymo sistemos blokinė diagrama [16]

Šioje sistemoje naudojamas *Siemens S7 313C Compact* programuojamas loginis valdiklis, kuris turi 24 skaitmeninius įėjimus, 16 skaitmeninių išėjimų, 5 analoginius įėjimus ir 2 analoginius išėjimus. Jis, pagal jame įrašytą programos algoritmą, valdo vandens siurblį. Jutikliai perduoda sistemos būseną valdikliui, kuris atitinkamai vykdo užprogramuotas komandas. Išskirstyti įvedimo/išvedimo moduliai leidžia „surinkti“ signalus iš išskirstytų jutiklių ir per tinklą perduoti juos valdikliui. Variklis turi apsaugą nuo perkaitimo, srovės ir įtampos daviklius.

Labiausiai dominantis modulis šioje analizuojamoje sistemoje yra *Ethernet* modulis. Tai yra komunikacijos procesorius skirtas *Siemens S7* valdikliams. Jis išplečia PLV galimybes leisdamas jungti valdiklį į tinklą. Šis modulis reikalingas tam, kad bevielio ryšio modulis (IWLAN – industrial wireless local area network) galėtų „bendrauti“ su PLV per TCP/IP protokolą. Šioje sistemoje IWLAN tinklas buvo sukurtas naudojant Siemens SCALANCE W kartos įrenginius. Prieigos taškas buvo įrengtas naudojant SCALANCE W788-1 PRO įrenginį, o kliento modulis – W744-1 PRO, kuris leidžia prisijungti prie sukurto prieigos taško. Bevielio tinklo įranga išsprendžia problemas, kurios yra būdingos laidų tiesimui: mažėja diegimo išlaidos, padidina lankstumą atnaujinant sistemas ir pagreitina tinklo įrengimą [16].

Šį bevielio ryšio taikymo metodą galima naudoti stacionariems objektams, bet ne judantiems, nes dalis įrangos yra jungiama prie valdomo objekto. Įranga užima nemažai vietos, todėl nepatogu, jei objektas yra nedidelis. Mano projektuojamoje sistemoje greitojo traukinio modelis juda bėgiais. Prie jo neįmanoma prijungti tokios įrangos, kokia pajungta šioje vandens siurblio valdymo sistemoje, todėl šį bevielį ryšį reikia realizuoti naudojant kuo mažesnių matmenų papildomą įrangą, pvz. *Bluetooth*.

1.5.3 „Bluetooth“ ryšio panaudojimo analizė

Indijos mokslininkai Harshit Gulati, Shriyansh Vaishya ir Sreehari Veeramachaneni savo darbe pristato realiu laiku per *Bluetooth* ryšį valdomą robotą [17]. Jo paskirtis ieškoti žmonių griuvėsiuose po katastrofos. Robotas turi vaizdo kamerą ir vykdantįjį įrenginį – „ranką“. Suradęs žmogų jis išsiunčia buvimo vietos duomenis į serverį. Sudėtingiausia užduotis ieškant išgyvenusių yra jų buvimo vietos nustatymas, nes „aklai“ naršant po griuvėsius galima sukelti pavojų išgyvenusiems ir prispaustiems žmonėms. Robotas atlieka šią užduotį – suranda išgyvenusius ir parodo jų buvimo vietą bei kelią kaip juos pasiekti.

Kadangi robotas turi veikti sudėtingomis sąlygomis, jis turi būti atsparus aplinkos poveikiui. Elektrinės grandinės turi būti apsaugotos karščiui atspariomis ir tvirtomis dėžutėmis taip, kad kokiam nors objektui užkritus ant roboto, jo elektroninės dalys liktų nepažeistos ir jis galėtų toliau funkcionuoti. Roboto galimybes itin išplečia jutikliai, kurie padeda atpažinti aplinką. Nors ir aprašomas robotas valdomas rankiniu būdu, jis turi galimybę pats išvengti kliūčių, kurios atsiranda jo kelyje. Infraraudonųjų spindulių atstumo jutikliai siunčia signalą į mikrovaldiklį, kuris pagal užprogramuotą algoritmą įvertiną situaciją ir atlieką atitinkamus veiksmus, pvz. sustabdo robotą ir atsitraukia staigiai atgal, jei būtina. Visa informacija apie netikėtas kliūtis ir roboto veiksmus iš karto perduodama operatoriui [17].

Kiekvienas robotas turi mikrovaldiklį, kuris vykdo visas operacijas. Šiame gelbėjimo robote naudojamas „*Arduino Duemilanove*“ mikrovaldiklis (12 pav.), kuris buvo parinktas, nes nesunkai perprantamas programavimas, lengvai prieinamas bei populiarus – yra plati dokumentacija, didelė vartotojų bendruomenė, iš kurių galima sulaukti pagalbos susidūrus su įvairiomis programavimo problemomis. Taip pat *Arduino* teikia atskirus modulius su integruotais *Bluetooth* arba *Ethernet* moduliais, kurie leidžia tiesiogiai keistis duomenimis tarp mikrovaldiklio ir bevielio ryšio modulių.

12 pav. Mikrovaldiklis „Arduino Duemilanove [17]

Aktualiausias modulis šiame gelbėjimo robote – *Bluetooth* modulis (13 pav.). Galimas abipusis ryšys. Tokio ryšio sistemos privalumai:

- Mažos energijos sąnaudos. Tai aktualu prietaisams, kurie naudoja baterijas, nes tai padeda prailginti įrenginio darbo laiką.
- Kintančio dažnio technologija. Ši technologija skaldo perduodamus duomenis ir mažus paketus išsiunčia 79 skirtingomis dažnio juostomis (po 1 MHz kiekviena) nuo 2400 iki 2483,5 MHz. Kintantis dažnis neleidžia *Bluetooth* signalams susilieti su kitais 2,4 GHz dažnio radijo signalais.
- Dvikryptis duomenų perdavimas. *Bluetooth* modulis leidžia ir siųsti, ir gauti duomenis. Dvipusis ryšys vienas dažniausių sistemos reikalavimų, o tokį ryšį su Bluetooth moduliais sukurti labai paprasta.
- Iš to pačio modulio galima išsiųsti duomenis į kelis kitus *Bluetooth* modulius. Moduliai yra identiški. Padavus signalą vienas nustatomas kaip valdantysis (*angl. Master*), o kiti – kaip pavaldusis (*angl. Slave*). Taip pat padavus reikiamus signalus moduliai suderinami taip, kad atitinkamas signalas pasiektų atitinkamą modulį.
- Prieinamumas. Šiais laikais beveik kiekvienas mobilusis bei išmanusis telefonas, taip pat nešiojamieji kompiuteriai turi integruotus *Bluetooth* ryšio modulius. Tai leidžia valdyti įrenginį tiesiai iš nešiojamo kompiuterio ar telefono – nebereikia pirkti papildomos valdymo įrangos.
- Kai naudojami 1.2 klasės *Bluetooth* moduliai, ryšio veikimo atstumas siekia 10 m, 1.3 klasės – 100 m. Naudojamus modulius galima pasirinkti pagal keliamus reikalavimus.

13 pav. Mikrovaldiklio „Arduino“ *Bluetooth* modulis [17]

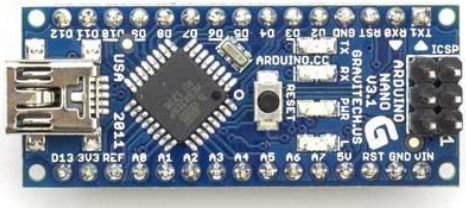
Visi išvardinti *Bluetooth* ryšio privalumai skatina naudoti būtent tokią technologiją mano projektuojamoje įtolintoje greičio kontrolės sistemoje. Aktualiausias parametras yra *Bluetooth* modulio dydis – jis pakankamai mažas, kad tilptų į greitojo traukinio modelį. Ryšio dengiamas plotas yra pakankamas, nes planuojamas naudoti 2 klasės *Bluetooth* ryšys veikia iki 10 m atstumu, o stendo išmatavimai mažesni.

2 AUTOMATINĖS ELEKTRINIO TRAUKINIO GREIČIO KONTROLĖS SISTEMOS KŪRIMAS

2.1 Techninė ir programinė įranga

2.1.1 Arduino Nano valdiklis

Automatinės elektrinio traukinio greičio kontrolės sistemos kūrimui pasirinktas Arduino Nano valdiklis. Tokį pasirinkimą įtakojo jo galimybės ir dydis. Jo ilgis tik 45 milimetrai, plotis 18 milimetrų ir jis sveria tik 5 gramus.



14 pav. Arduino Nano [18]

Jo funkcionalumo galimybės yra ne mažesnės nei Arduino Duemilanove, kurį savo robotų kūrime naudojo Indijos mokslininkai Harshit Gulati, Shriyansh Vaishya ir Sreehari Veeramachaneni, kurių darbo rezultatai aptarti ankstesniame skyriuje. Šio mikrovaldiklio detalesni techniniai duomenys įvardijami 3 lentelėje.

3 lentelė. Techniniai arduino Nano duomenys

Mikrokontroleris	Atmel ATmega168 or ATmega328
Darbinė įtampa (loginis lygis)	5 V
Įėjimo įtampa (rekomenduojama)	7-12 V
Įėjimo įtampa (ribos)	6-20 V
Skaitmeninės I / O jungtys	14 (6 iš jų teikia PWM išėjimus)
Analogijio įėjimo jungtys	8
Flash atmintis	32 KB iš kurių 2 KB naudoja bootloader
SRAM	2 KB
EEPROM	1 KB
Taktinis dažnis	16 MHz

Arduino Nano gali būti maitinamas tiek per Mini-B USB jungtį, tiek per 6-20V nereguliuojamą išorinį maitinimo šaltinį (pin 30) arba 5V reguliuojamą išorinį maitinimo šaltinį (pin 27). Energijos

šaltinis yra automatiškai parenkamas pirmenybę suteikiant aukščiausios įtampos šaltiniui. Kiekviena iš 14 jungčių ant Arduino Nano plokštės gali buti naudojama kaip įejimo arba išėjimo jungtys, programiniu būdu nurodant jų paskirtį tokiomis komandomis kaip pinMode(), digitalWrite() ardigitalRead(). Jos veikia 5 voltų įtampoje ir gali suteikti arba priimti daugiausiai 40 miliamperų srovę. Taip pat verta paminėti, jog kai kurios jungtys turi specializuotas funkcijas:

Išoriniai pertraukimai: 2 ir 3 jungtys. Šios jungtys gali būti konfigūruojamos taip, kad sukeltų pertraukimus dėl mažos vertės, reikšmei pradėjus didėti arba mažėti arba dėl bet kokių vertės pokyčių.

PWM: 3, 5, 6, 9, 10, 11 jungtys. Suteikia 8 bitų PWM išėjimą su analogWrite() funkcija.

LED: 13 jungtis. Ant mikrovaldiklio yra pritaisytas LED šviesos diodas ir jis yra sujungtas su 13 jungtimi. Kai ši jungtis turi reikšmę HIGH, LED šviesos diodas šviečia, o kai šios jungties reikšmė LOW, jis nešviečia.

Arduino Nano turi 8 analoginius įėjimus, kurių kiekvienas gali suteikti 10 bitų raišką (t.y. 1024 skirtingas reikšmes). Pagal nutylėjimą jos matuoja nuo įžeminimo (*angl. ground*) iki 5 voltų, nors viršutinę reikšmę dar galima pakeisti naudojant analogReference() funkciją.

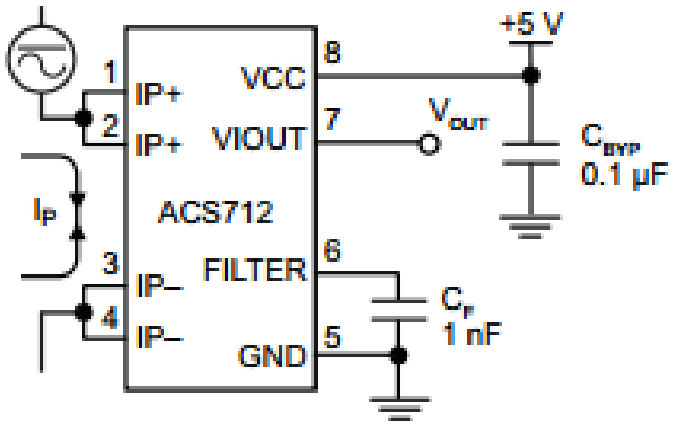
Taigi apžvelgus šio mažo, bet geromis techninėmis savybėmis pasižyminčio mikrovaldiklio galimybes nutarta nebeieškoti kitų variantų, nes šio valdiklio galimybės yra pakankamos suprojektuoti elektrinio greitojo traukinio modelio greičio valdymo sistemai. Tai yra vienas mažiausių rinkoje esančių mikrovaldiklių, o kompaktiškas dydis greitojo traukinio modelio kūrime yra labai svarbus, nes čia vietos turi atsirasti ne tik valdikliui, bet ir varikliui, matavimo ir bevielio ryšio įrangai [18].

2.1.2 Srovės matuoklis ASC712

15 pav. Srovės matuoklis ASC712 [18]

Srovės matuoklis Allegro ACS712 suteikia ekonominius ir tikslius sprendimus AC ir DC srovės stebėjimui pramonės, komercijos ir komunikacijų sistemose. Šio prietaiso įdiegimas ir naudojimas sistemoje yra paprastas ir nereikalaujantis specifinių žinių. Įprastai šis prietaisas naudojamas variklių

apkrovų stebėjime ir valdyme ir impulsinių maitinimo šaltinių viršsrovio apsaugai. Prietaisas susideda iš tikslios, mažo kompensavimo linijinės Hall integrinės grandinės su variniais kontaktiniais takeliais, esančiais plokštelės paviršiuje. Srovei tekant per šiuos varinius takelius yra generuojamas magnetinis laukas, kurį Hall integrinė grandinė konvertuoja į proporcingą įtampą. Prietaiso išėjimo signalas turi teigiamą nuolydį, kai didėjanti srovė teka per pirminį varinį laidumo taką (iš 1 ir 2 kontakto į 3 ir 4 kontaktus) kuris yra naudojamas srovės palyginimui (16 pav.).

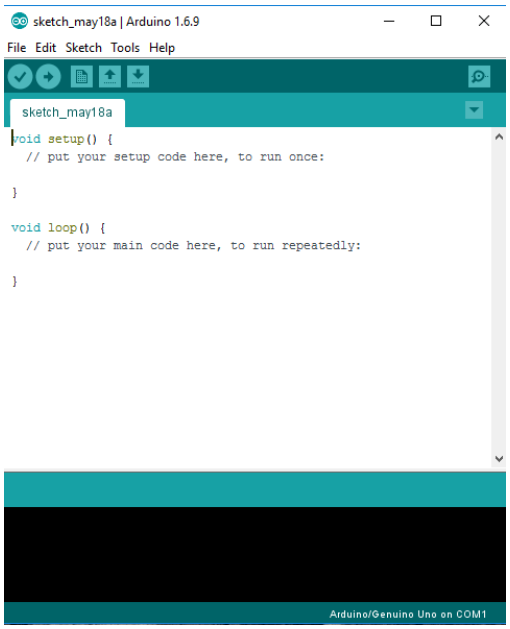


16 pav. ASC712 integrinės grandinės schema [18]

Vidinė šio tako varža yra 1.2 mΩ, dėl to elektros energijos nuostoliai yra labai maži. Varinio laidininko storis leidžia prietaisui atlaikyti penkis kartus didesnį viršsrovį nei rekomenduojama jo techninėse charakteristikose. Laidininko tako gnybtai yra elektriškai izoliuoti nuo signalo gnybtų (jungtys nuo 5 iki 8) (16 pav.). Tai leidžia tam tikrose situacijose naudoti šį prietaisą be papildomų izoliatorių ir specialių izoliavimo technikų.

2.1.3 Arduino 1.6.9

Arduino Nano programavimui naudojama Arduino 1.6.9 programa (17 pav.). Tai specialiai Arduino mikrovaldikliams programuoti skirta atvirojo kodo programa. Šia programa galima ne tik suvesti reikalingą programinį kodą, bet ir įkelti jį į mikrovaldiklį.



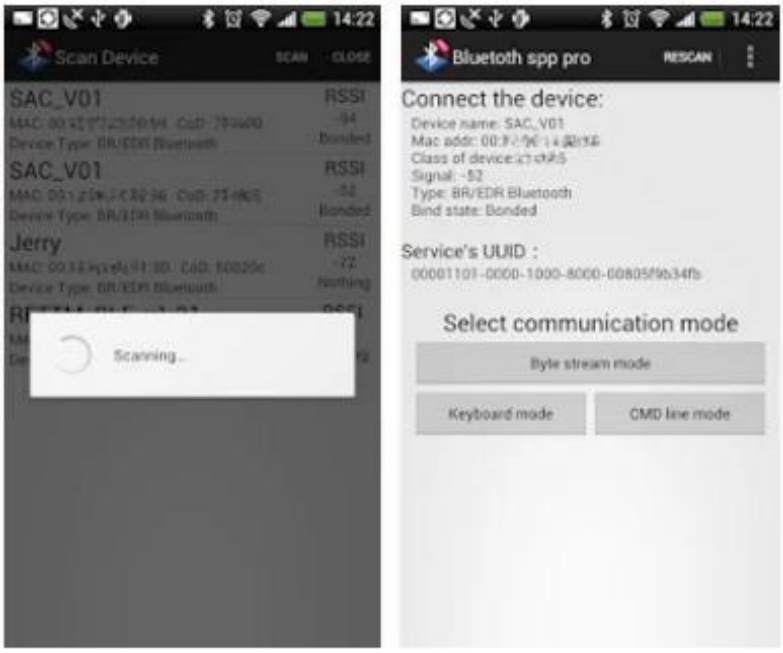
17 pav. Arduino 1.6.9.

Ši programa gali veikti ne tik Windows, bet ir Mac OS X ar Linux operacinių sistemų aplinkoje. Rašyti programinius kodus šios programos aplinkoje gana paprasta. Arduino programavimo kalbos pagrindą sudaro C++ programavimo kalba. Yra pateikiama daug pavyzdžių pradedantiesiems. Taip pat darbą palengvina specialios bibliotekos, kuriose pateikiamas jau suvestas programos kodo pagrindas, kurį belieka pritaikyti savo individualiam projektui.

Šios programos aplinkoje bus kuriamas automatinės elektrinio traukinio modelio greičio kontrolės sistemos, mažinančios energijos sąnaudas programinis kodas. Taip pat viena iš šios programos funkcijų yra rinkti vykdomos programos duomenis. Tuo pasinaudojus bus stebimas sistemos suvartojamos energijos kiekis kintant variklio greičiui.

2.1.4 Bluetooth spp tools pro

Kadangi kuriama sistema yra integruojama į judančią sistemą, jos valdymui yra būtinas bevielis ryšys. Tam bus panaudota Bluetooth spp tools pro programa. Ši programa skirta Bluetooth bevielio ryšio įgalinimui tarp valdiklio ir valdymo prietaiso. Jos vartotojo sąsaja pavaizduota 18 paveiksle.

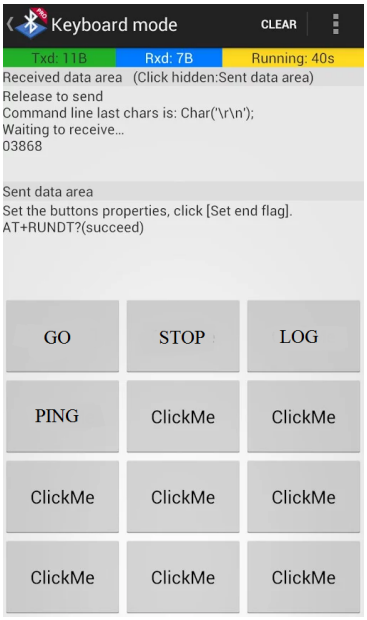


18 pav. Bluetooth spp tools pro vartotojo sąsaja

Programos ypatybės:

- Gali ieškoti Bluetooth prietaisų ir rodyti signalo stiprumą;
- Gali siųsti ir priimti duomenis;
- Galimas ASCII ir HEX įėjimo ir išėjimo režimas;
- Gauti duomenys gali būti išsaugoti SD kortelėje;

Įdiegus programą, belieka įrenginyje įjungti Bluetooth ryšį ir nuskenavus netoliese esančius įrenginius pasirinkti norimą valdyti valdiklį su integruotu Bluetooth moduliu. Prisijungus belieka ekrane atsiradusioje klaviatūroje pervardyti mygtukus pagal savo sistemą (19 pav.).

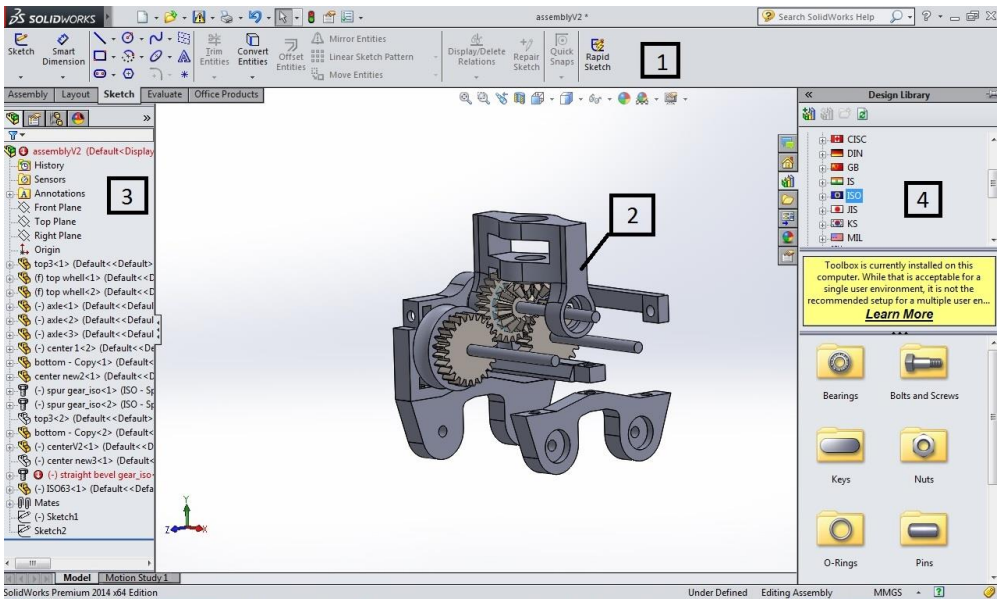


19 pav. Bluetooth spp tools pro pritaikymas sistemoje

Mygtukai turi būti pavadinti taip, kaip nurodyta programiniame kode, kuris rašomas ankščiau minėtos Arduino programos aplinkoje. Suvedus visas reikiamas funkcijas galima teigti, kad įtolintas valdymas yra integruotas elektrinio traukinio modelio greičio kontrolės sistemoje.

2.1.5 SolidWorks programinis paketas

Elektrinio traukinio modelio greičio kontrolės sistemos projektavimo procese reikia atsižvelgti ir į saugumą. Tam, kad bėgių konstrukcijoms nebūtų padaryta žala, jų tvirtumas tikrinamas kompiuterinės simuliacijos metu. Simuliacijos atliekamos naudojant SolidWorks programinį paketą. Ši priemonė plačiai taikoma tokiuose pramonės sektoriuose kaip aviacija, transportas, baldų gamyba, energetika, statyba ir elektronika [19]. Jos aplinkoje projektuojamoms detalėms simuliacijų metu gali būti atliekama inžinerinė analizė, taip surandant jų silpnąsias konstrukcijos vietas. 20 paveiksle pavaizduotas pagrindinis SolidWorks darbo langas.

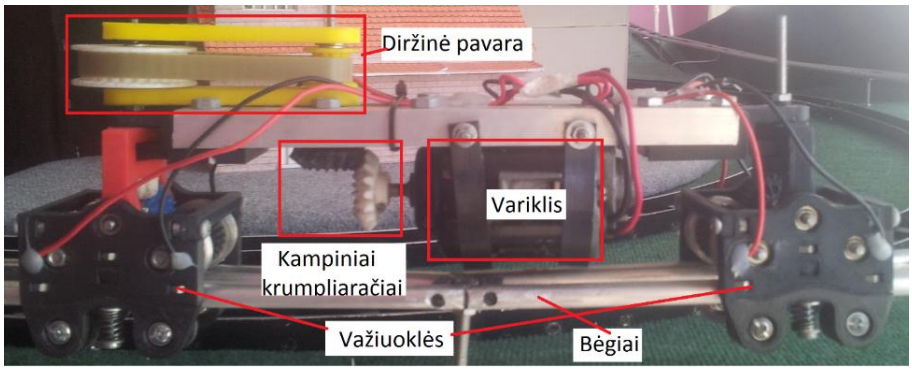


20 pav. SolidWorks darbo aplinka.

20 paveiksle numeriu 1 pažymėta pagrindinė SolidWorks funkcijų juosta. Ji gali būti sudaryta pagal individualius poreikius. Pagrindinės funkcijų grupės tokios kaip *Sketch* ir *Features* yra būtinos pradėti darbą su SolidWorks. Jos naudojamos brėžti plokštuminiams brėžiniams ir paversti juos erdviniais. Taip pat šis programinis paketas turi daug kitų naudingų funkcijų grupių, tokių kaip lakštinių metalų, paviršinių brėžinių, metalų suvirinimų imitavimo ar simuliacijų grupių. Pastaroji grupė plačiai naudojama šiame darbe, nes atveria plačias inžinerinių tyrimų galimybes, tokias kaip detalių atsparumo ar jų aptakumo tyrimo simuliacijas. 2 numeriu pažymėtas projektavimo laukas, kuriame vyksta visas virtualaus modelio kūrimas, pradedant nuo plokštuminių brėžinių ir atskirų detalių braižymo, baigiant detalių ir konstrukcijų testavimo, medžiagos ir spalvos parinkimu. 3 numeriu pažymėtas funkcijų tvarkyklės medis, kuriame galima prieiti prie kiekvienos modelyje panaudotos funkcijos. Šio medžio dėka galima keisti funkcijų parametrus, jas paslėpti, ištrinti, pervardyti ir grupuoti tam, kad būtų kuo paprasčiau surasti reikiamą atliktą funkciją ir taip sutaupant laiko padidinti darbo našumą. 4 numeriu pažymėta sritis – SolidWorks resursų biblioteka. Ji skirta tam, vartotojui nereikėtų projektuoti įvairių tvirtinimo elementų, metalo profilių, guolių ar dantračių. Patogu tai, kad ši biblioteka suskirstyta į tarptautinių standartų grupes, tokias kaip ANSI Inch, ANSI Metric, BSI, ISO ir daug kitų, todėl vartotojui visada patogu pasirinkti reikiamo elemento matmenis, parengtus pagal standarto nuostatas. Šiame darbe naudojami elementai atitinka vakarų Europoje naudojamus ISO standartus.

2.2 Greičio kontrolės sistemos projektavimas

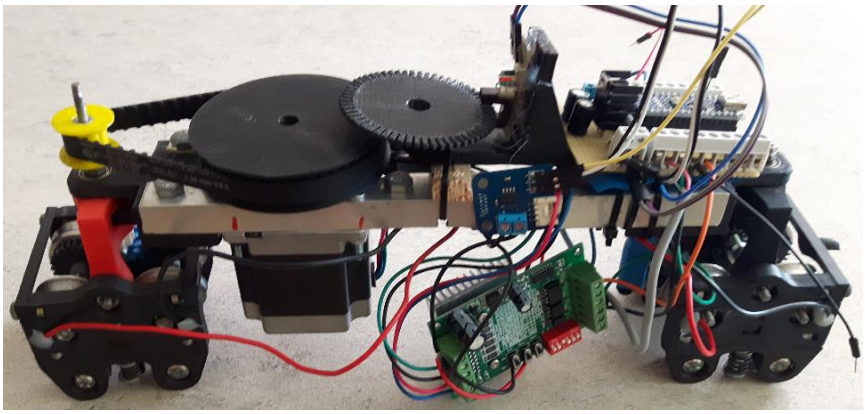
21 pav. pavaizduota pilnai surinkto funkcionalaus greitojo traukinio prototipo ankstesnė versija.



21 pav. Greitojo traukinio prototipas.

Paveiksle pavaizduoti traukinio bėgiai, kuriais traukiniui perduodama elektros energija. Nuo bėgių iki variklio elektros energija perduodama aliumininiais ratais. Nuo ratų ašių elektros energija laidais perduodama į variklį. Variklio sukimo energija kampiniais krumpliaračiais perduodama į diržinę pavarą, kuri suka važiuoklės krumpliaračių sistemą, kuri savo ruožtu suka traukinio ratus [24].

Tam, kad į šį elektrinio traukinio modelį būtų galima įdiegti naują greičio kontrolės sistemą, reikėjo atlikti keletą pakeitimų konstrukcijoje ir padaryti vietos valdymo bei matavimo įrangai. Perdarytas elektrinio traukinio modelis pavaizduotas 22 paveiksle.



22 pav. Perdarytas elektrinio traukinio modelis

Kaip galima matyti iš 21 ir 22 paveikslų, traukinio modelis gavo naują variklį, taip pat atsisakyta kampinių krumpliaračių kurie perduoda variklio sukimo energiją diržinei pavarai. Toks sprendimas priimtas norint palikti daugiau vietos valdymo įrangai, o taip pat jų atsisakius sumažėjo variklio energijos praradimas. Nuo šiol variklis tiesiogiai suka diržinės pavaros būgną. Kaip matome, prie šio būgno pritvirtintas enkoderis, kuris matuoja variklio sukimosi greitį, šalia jo pritvirtintas Arduino

Nano valdiklis, bei srovės matuoklis ASC712. Po korpuso apačia bus pritvirtintas variklio valdiklis. Visi šie pakeitimai leis įdiegti naują greičio kontrolės sistemą ir atlikti tyrimus susijusius su ja.

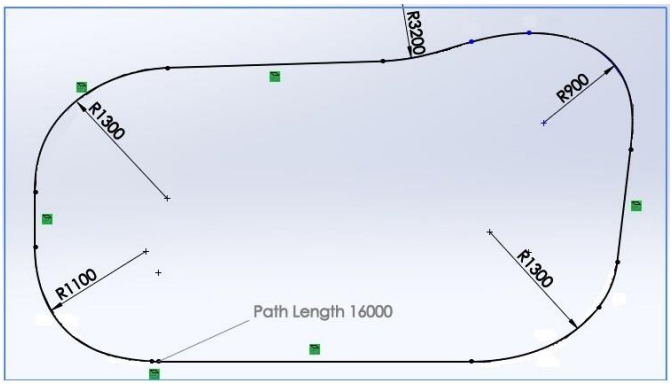
2.2.1 Greičio apribojimų bėgių ruožuose skaičiavimas

Kuriant traukinio greičio kontrolės sistemą pagrindinis kriterijus yra saugumas. Greičio kontrolės sistema kuriama taip, kad traukinys įveiktų posūkius kuo įmanoma didesniu greičiau, tačiau nesukeldamas avarinės situacijos ir minimaliai prailgindamas kelionės laiką. Šiam tikslui pasiekti į pagalbą pasitelkiama SolidWorks programinė įranga. Ji padės simuliacijos proceso metu nustatyti, kaip traukinio įcentrinės jėgos atsirandančios posūkiuose veikia atskirus bėgių konstrukcijos elementus (23 pav.).

23 pav. Bėgių laikiklis.

23 pav. pavaizduotas traukinio bėgių laikiklis, kuris sujungia vamzdžius, iš kurių pagaminti traukinio bėgiai, ir laiko juos ant metalinio strypo, kuris imituoja koloną. Laikikliai esantys bėgių posūkių ruožuose pravažiuojant traukiniui yra deformuojami dėl atsirandančios traukinio įcentrinės jėgos.

24 pav. pateiktas elektrinio greitojo traukinio bėgių žemėlapis.



24 pav. Greitojo traukinio bėgių žemėlapis.

Žemėlapyje pateikti greitojo traukinio bėgių posūkių spinduliai. Posūkio spindulio reikšmė reikalinga apskaičiuoti greitojo traukinio įcentrinės jėgos dydžiui važiuojant posūkiu. Įcentrinė jėga apskaičiuojama pagal formulę (10):

$$F_c = \frac{mv^2}{r} \tag{10}$$

Čia m – kūno (traukinio) masė, v – kūno greitis ir r – posūkio spindulys. Šiame darbe naudojamo traukinio svoris – 3 kilogramai, o jo didžiausias išvystomas greitis 2,3 metro per sekundę.

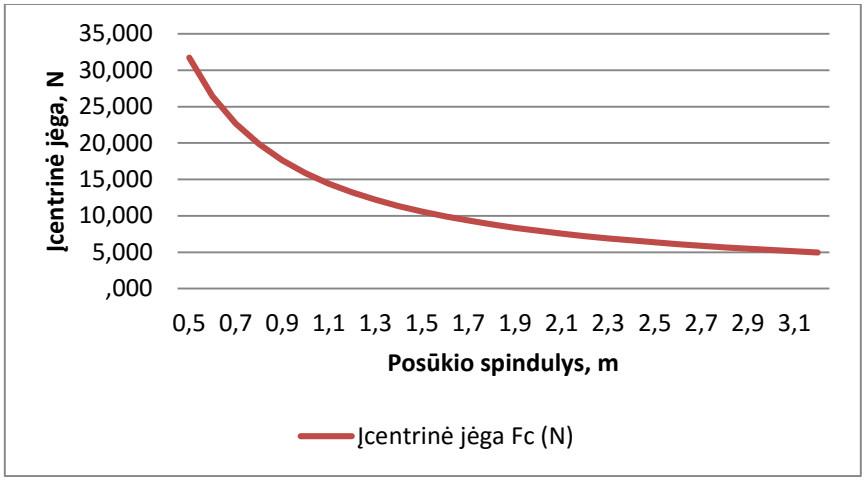
4 lentelėje pateikti įcentrinės jėgos dydžiai pagal posūkio spindulio dydį kai traukinys visuose bėgių ruožuose važiuoja maksimaliu išvystomu greičiu.

4 lentelė. Įcentrinės jėgos dydžio priklausomybė nuo posūkio spindulio.

Posūkio spindulys R (m)	Įcentrinė jėga Fc (N)
0,5	31,74
0,6	26,45
0,7	22,67
0,8	19,84
0,9	**17,63**
1	15,87
1,1	**14,43**
1,2	13,23
1,3	**12,21**
1,4	11,34
1,5	10,58
1,6	9,92
1,7	9,34
1,8	8,82
2	7,94
2,5	6,35
3	5,29
3,2	**4,96**

Lentelėje paryškintos reikšmės atitinka posūkių, kuriais važiuoja greitasis traukinys, dydžius.

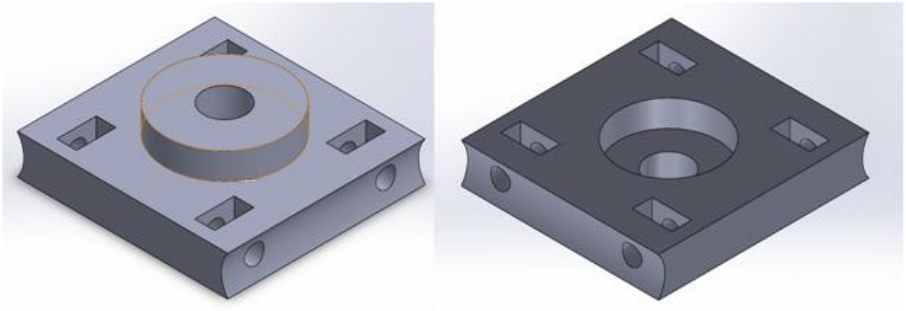
Taip pat pateikiamas ir įcentrinės jėgos priklausomybės nuo posūkio kampo grafikas (25 pav.), iš kurio matoma kaip didėjant posūkio spinduliui mažėja įcentrinės jėgos dydis.



25 pav. Įcentrinės jėgos priklausomybės nuo posūkio spindulio grafikas.

2.2.2 Bėgių laikiklių testavimas simuliuojant veikiančias jėgas

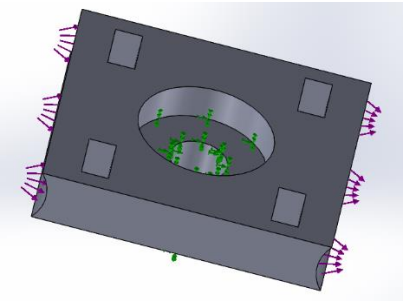
Atsižvelgiant į šią kreivę projektuojama greičio kontrolės sistema paremta traukinio greičio reguliavimu įvažiuojant į posūkį ir išvažiuojant iš jo į tiesų bėgių ruožą. Stebimos traukinio įcentrinės jėgos sukeliamos bėgių laikiklių deformacijos, galinčios padaryti šiam konstrukcijos elementui negrįžtamą deformaciją. Bėgių laikiklis pavaizduotas 26 pav.



26 pav. Bėgių laikiklis virtualioje aplinkoje

Simuliacijos metu testuojama, kaip pro šalį važiuojantis traukinys veikia bėgių laikiklius skirtingo spindulio posūkiuose. Atitinkamai nuo to kaip stipriai detalė transformuojama, mažinamas traukinio greitis iki tokios ribos, kurios neviršijant būtų užkirstas saugumas darant kaip įmanoma mažesnius nuostolius traukinio kelionės laikui.

Simuliacijos atlikimui būtina nurodyti tvirtinimo taškus ir jėgų veikimo kryptis. Jie pavaizduoti 27 pav.

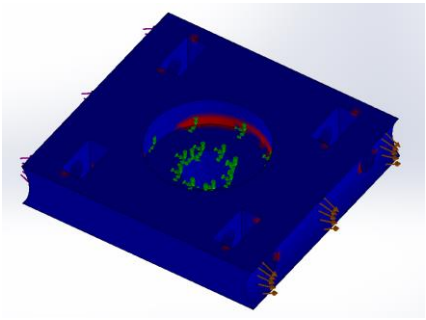


27 pav. Tvirtinimo taškai ir jėgų veikimo kryptys.

Paveiksle žaliomis rodyklėmis parodoma, kuriose vietose yra detalės tvirtinimo taškai. Violetinės rodyklės vaizduoja jėgų veikimo kryptis.

Simuliacijos pagalba tikrinamas detalės atsparumas imituojant ją veikiančias jėgas. Iš pradžių detalė testuojama imituojant didžiausias jėgas, kokios ją veikia greitojo traukinio mažiausio spindulio posūkyje. Tai posūkis kurio spindulys lygus 0,9 m. Jame traukiniui važiuojant maksimaliu greičiu susidaro 17,63 N dydžio įcentrinė jėga.

Žemiau 28 pav. pavaizduota, kaip 17,63 N dydžio jėga veikia detalę. Raudonai pažymėtos detalės zonos rodo kuriai detalės vietai tenka didžiausia apkrova. Tose vietose veikiant jėgoms dėl deformacijos detalė gali neatlaikyti ir sulūžti [24].



28 pav. Jėgų veikiama detalė.

Tam, kad būtų nustatyta kokio dydžio jėgą gali atlaikyti detalė, simuliacijų metu surandama optimalus įcentrinės jėgos dydis, kuriam veikiant detalei nedaroma negrįžtama deformacija. Atlikus simuliaciją nustatoma, kad didžiausia jėga, kuriai veikiant detalei nedaroma žala yra 9 N, todėl kuriant greičio kontrolės sistemą būtina keisti greitį taip, kad bet kuriame bėgių posūkyje traukinį veiktų ne didesnė nei 9 N jėga. Atsižvelgiant į 4 lentelėje pateiktus duomenis matoma, kad traukinys nesudarydamas žalos bėgių konstrukcijoms maksimaliu greičiu gali važiuoti didesnio, nei 1,8 metro spindulio posūkiais.

Tam, kad būtų apskaičiuotas optimalus traukinio greitis kiekviename posūkyje, naudojama formulė (11):

$$v = \sqrt{\frac{F_c \times r}{m}} \quad (11)$$

Pagal formulę išskaičiuojamas maksimalus galimas greitis pirmame posūkyje, kurio spindulys 1,1 metro.

$$v = \sqrt{\frac{F_c \times r}{m}} = \sqrt{\frac{9 \times 1{,}1}{3}} = 1{,}81659 \approx 1{,}82\ m/s \quad (12)$$

Gautas atsakymas, kad didžiausias galimas greitis šiame posūkyje yra 1,82 m/s. Greitis kituose posūkiuose apskaičiuojamas analogiškai. Gauti rezultatai rodo, kad 0,9 metro spindulio posūkiuose maksimalus greitis lygus 1,64 m/s, o 1,3 metro spindulio posūkiuose leistinas greitis yra 1,97.

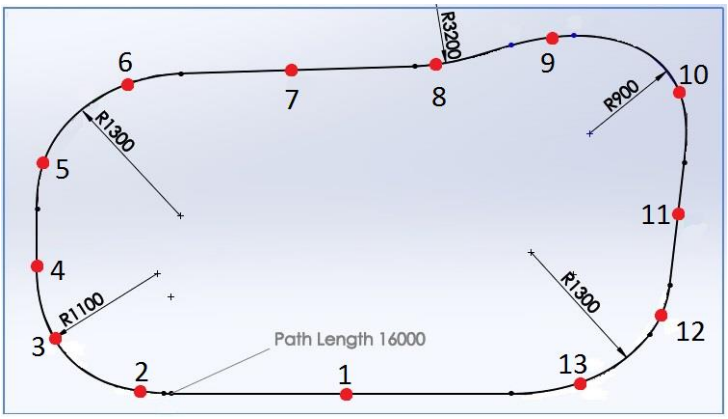
Remiantis kompiuterinės simuliacijos duomenimis nustatyta kokio dydžio įcentrinė jėga gali veikti traukinį, kad šis posūkyje nesulaužytų bėgių konstrukcijos elementų. Sudaryta lentelė, kurioje parodoma greičio priklausomybė nuo posūkio spindulio dydžio (5 lentelė).

5 lentelė. Greičio priklausomybė nuo posūkio spindulio dydžio.

Posūkio spindulys R (m)	Greitis (m/s)
0,5	1,22
0,6	1,34
0,7	1,45
0,8	1,55
0,9	1,64
1	1,73
1,1	1,82
1,2	1,90
1,3	1,97
1,4	2,05
1,5	2,12
1,6	2,19
1,7	2,26
1,8	2,32
1,9	2,39
2	2,45

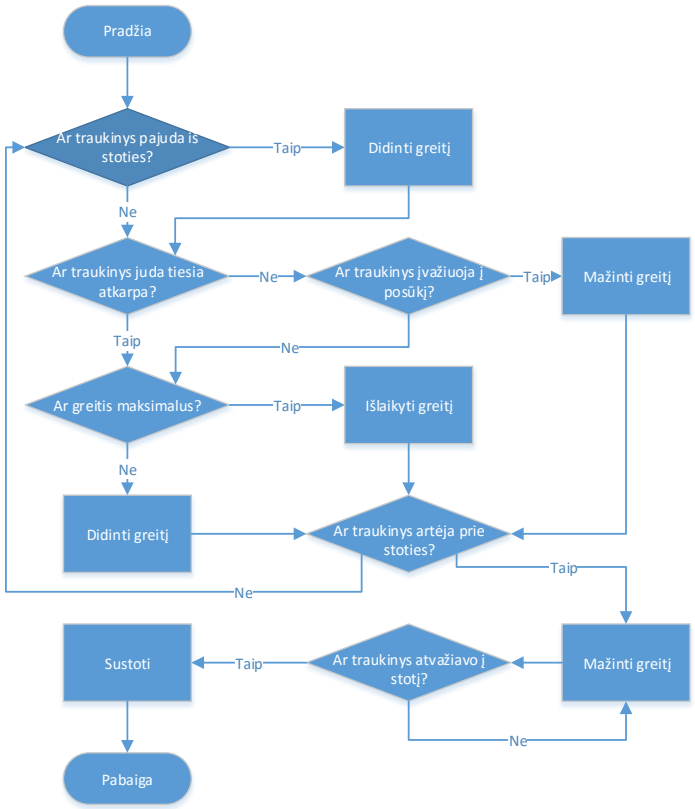
2.2.3 Traukinio greičio kontrolės sistemos parametrų skaičiavimas

Žemiau pateiktame paveikslėlyje pavaizduotas greitojo traukinio bėgių žemėlapis ir greičio kontrolės taškai, kuriuose keičiamos traukinio varikio apsukos ir mažinamas greitis (29 pav.).



29 pav. Greitojo traukinio bėgių žemėlapis ir greičio kontrolės taškai

Pažymėta 13 greičio kontrolės taškų, kuriuose atsižvelgiant į valdymo sistemos algoritmą, pavaizduotą 30 pav., priimami sprendimai dėl traukinio greičio keitimo arba jo išlaikymo.



30 pav. Greičio reguliavimo sistemos algoritmas.

29 pav. pavaizduotame greičio kontrolės taškų žemėlapyje 1 numeriu pažymėta greitojo traukinio stotis, iš kur įprastai pajuda greitasis traukinys. Pagal valdymo sistemos algoritmą patikrinus ar traukinys pajuda nuo stoties priimamas spendimas didinti greitį. Greitis didinamas iki maksimumo, arba tol, kol pasiekiamas kitas kontrolės punktas, šiuo atveju pažymėtas 2 numeriu. Šiame kontrolės punkte greitis turi būti mažinamas todėl, kad traukinys įvažiuoja į posūkį. Sumažinus greitį ir pasiekus 3 kontrolės punktą patikrinama sąlyga, ar traukinys neartėja prie stoties. Patikrinus ir gavus neigiamą atsakymą tikrinama sąlyga ar traukinys pajuda iš stoties ir ar jis juda tiesia atkarpa, tačiau ir čia gaunamas neigiamas atsakymas veda prie kitos algoritmo sąlygos, kur iškeliamas klausimas ar traukinys įvažiuoja į posūkį. Taip pat gautas neigiamas atsakymas leidžia daryti išvadą, kad traukinys važiuoja ne į posūkį, o iš jo, todėl tikrinama sąlyga ar traukinys važiuoja maksimaliu greičiu. Šiuo atveju traukinys juda posūkyje, vadinasi praeitame kontrolės punkte jo greitis buvo sumažintas. Tai reiškia, kad jis juda ne maksimaliu greičiu, todėl greitis didinamas iki maksimalaus arba tol, kol pasiekiamas kitas kontrolės punktas.

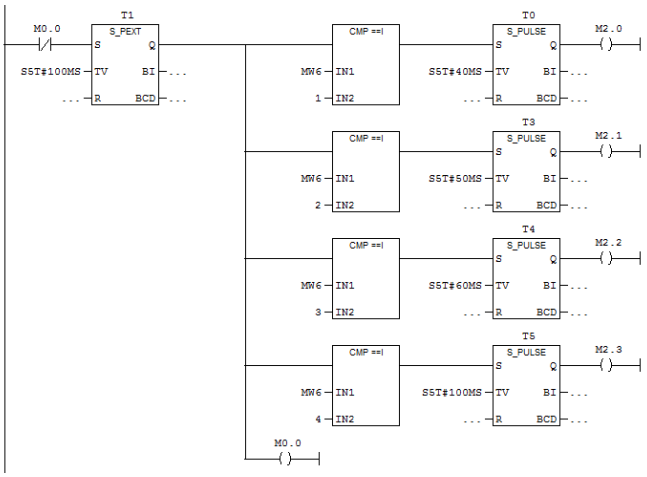
Pasiekus 4 greičio kontrolės punktą vadovaujamasi ta pačia operacijų seka. Tačiau čia priimti sprendimai skirsis nuo anksčiau aprašytų tuo, kad traukinys 4 numeriu pažymėtame greičio kontrolės punkte juda ne į posūkį ar iš jo, bet tiesia linija. Todėl tikrinant sąlygą ar traukinys juda tiesia linija gaunamas teigiamas atsakymas veda prie klausimo ar greitis maksimalus, kur patikrinus sąlygą priimamas sprendimas greitį didinti arba jei jis jau pasiekęs maksimalią reikšmę, jį išlaikyti.

Pasiekus 5 greičio kontrolės punktą elgiamasi analogiškai taip pat kaip ir 2 greičio kontrolės punkte, o po jo sekančiuose 6 ir 7 punktuose atitinkamai taip pat kaip 3 ir 4 punktuose. Toliau sekant žemėlapį matome bėgių vingį, po kurio yra ne tiesus ruožas, o vingis į kitą posūkį. Šioje situacijoje vadovaujamasi analogiškai taip, kaip ankščiau aprašyta įvažiuojant į posūkį.

Po to sekančiuose punktuose pažymėtuose 10, 11 ir 12 numeriais elgiamasi analogiškai taip pat, kaip aprašyta 3 – 4 – 5 punktuose, kur aprašomas judėjimas iš posūkio į tiesiąja ir įvažiuojama į posūkį. Privažiavus 13 kontrolės punktą visos operacijos vykdomos taip pat kaip ir išvažiuojant iš posūkio, tačiau čia tikrinant sąlygą ar traukinys artėja prie stoties gaunamas teigiamas atsakymas, todėl greitis mažinamas tol, kol pasiekiama stotis ir traukinys visiškai sustoja [24].

3 GREIČIO KONTROLĖS MAŽINANČIOS ENERGIJOS SĄNAUDAS VERIFIKACIJA

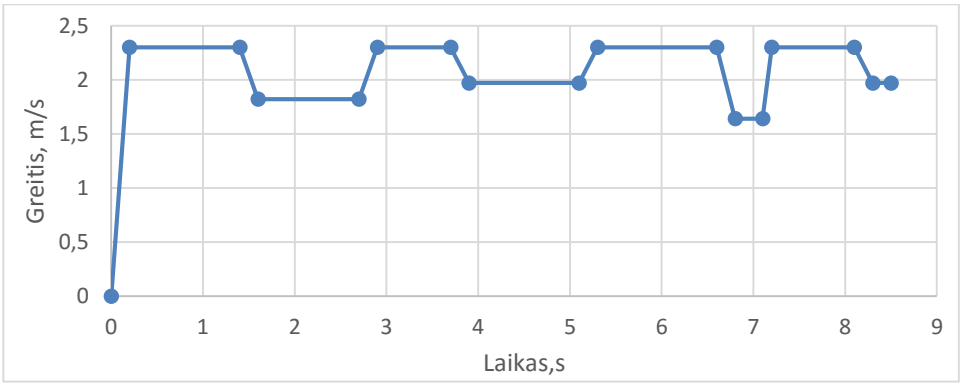
Testavimo tikslais *Ladder* programavimo kalba buvo sukurta traukinio greičio kontrolės sistemos programa. Ji buvo atsakinga tik už greičio apribojimus skirtinguose bėgių ruožuose.



31 pav. PWM moduliacijos ladder programavimo kalbos išraiška.

31 pav. pavaizduotas programos fragmentas, kuriame PWM moduliacijų pagalba nustatomi visi reikalingi variklio sukimosi greičiai.

Tyrimo metu nustatinėjama traukinio greičio priklausomybė nuo laiko. Tam, kad būtų pasiektas kuo didesnis matavimo tikslumas, valdiklis užprogramuotas taip, kad registruotų greičio pasikeitimus laiko atžvilgiu. Gauti rezultatai leido sekti traukinio važiavimą tiesiosiomis ir posūkių ruožais. Gautas tyrimo rezultatai pavaizduotas žemiau esančiame paveiksle (32 pav.). Pažvelgus į grafiką, galima nustatyti, kada traukinys įvažiuoja į posūkį ar išvažiuoja iš jo ar kiek maršrute yra tiesių bėgių ruožų.



32 pav. Traukinio greičio priklausomybė nuo laiko.

Važiuojant posūkiais sistema priklausomai nuo posūkio spindulio dydžio sumažina variklio apsisukimų skaičių nuo 100 % iki 57 %, 48 % ar 39 %. Traukinio važiavimo greitis atitinkamai šioms procentinėms reikšmėms kinta nuo 2,3 m/s iki 1,97 m/s, 1,82 m/s ir 1,64 m/s. Tačiau kaip matome, ši už greičio kontrolę atsakinga sistema greitį keičia naudodama trapecinį greičio profilį, todėl varikliui tenka didelė apkrova, ratai ima suktis greičiau, nei juda pats traukinio modelis, o dėl to išauga energijos sąnaudos. Atsižvelgiant į tai sukurta nauja sistema atsakinga ne tik už variklio greičio keitimą, bet ir už traukinio judėjimo tolydumą, kad būtų išvengta bereikalingų variklio apkrovų ir dėl per greitai pasikeitusių apsukų atsiradusio ratų praslydimo.

Kaip jau minėta, naujos sistemos kūrimui pasirinkti greičio profiliai. Kadangi senoji sistemos versija savo veikimu panaši į trapecinius greičio profilius, nauja sistema kuriama pasirinkus S-kreivės tipo greičio profilius. Žemiau pavaizduotas programinio kodo fragmentas, vaizduojantis S-kreivės profilį aprašytą Arduino programos darbo aplinkoje.

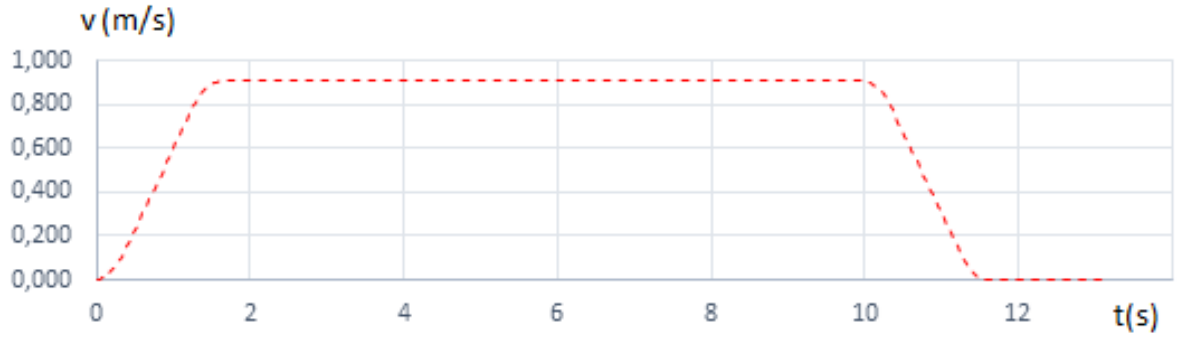
```
   String inputString = "";
   boolean stringComplete = false;
   //Motor1 = Y axis
    Motor2 = Z axis
      f        f
           f           f          f
       int        int      boolean               int           int

   //Motor [xyz] (name, invertAxisDir, en, dir, step, stopCW, stopCCW, maxSpeed,
acceleration, deceleration, diameter, gearRadio, stepSize, trosoKoef, useTimer3)
   Motor y ("Y" , false, 4, 5, 9, 0.5, 2, 26.0, 0.22, 2); // CCW positive
```

Šiame kodo fragmente nurodomi tokie parametrai kaip variklio sukimosi kryptis, užduodamas norimas maksimalus greitis ir laikas, per kurį variklis turi pasiekti didžiausią greitį ir laikas, per kurį nuo didžiausio greičio variklis visiškai sustotų, taip pat nurodomas laikas po kurio pradedama lėtėti. Programos kodas su tokiais užduotais parametrais įkeliamas į valdiklį. Norint patikrinti kaip veikia sistema, bevieliu Bluetooth ryšiu išsiunčiama komanda „go“, kuri programos kode aprašoma kaip sistemos vykdymo komanda. Taip pat kartu su šia komanda išsiunčiama ir komanda „log“, kuri programos kode reiškia sistemos duomenų rinkimą. Sistemai įvykdžius darbą, gaunami tokie duomenys (33 pav.).

```
2538 0.000 0.000 0.000 0.000 0.000 -0.08
2578 0.075 0.002 0.000 0.000 0.002 1.189
2618 0.150 0.007 0.000 0.043 0.004 0.632
2658 0.225 0.016 0.001 0.107 0.010 1.110
2698 0.300 0.027 0.002 0.107 0.012 0.659
2738 0.375 0.041 0.003 0.085 0.014 0.871
2778 0.450 0.059 0.005 0.064 0.019 1.136
2818 0.525 0.079 0.008 0.150 0.025 1.136
2859 0.602 0.103 0.012 0.064 0.027 1.269
2899 0.677 0.129 0.017 0.064 0.031 0.871
2939 0.752 0.158 0.023 0.101 0.035 0.898
2979 0.752 0.188 0.030 0.107 0.040 1.136
3019 0.752 0.218 0.039 0.107 0.044 1.083
3059 0.752 0.242 0.046 0.085 0.047 0.898
3099 0.752 0.273 0.057 0.085 0.051 0.818
3139 0.752 0.303 0.069 0.107 0.058 1.189
3179 0.752 0.333 0.082 0.150 0.064 0.579
3219 0.752 0.364 0.097 0.171 0.072 0.791
3259 0.752 0.394 0.112 0.171 0.078 1.428
3299 0.752 0.424 0.129 0.171 0.085 1.030
3339 0.752 0.455 0.147 0.192 0.095 0.871
```

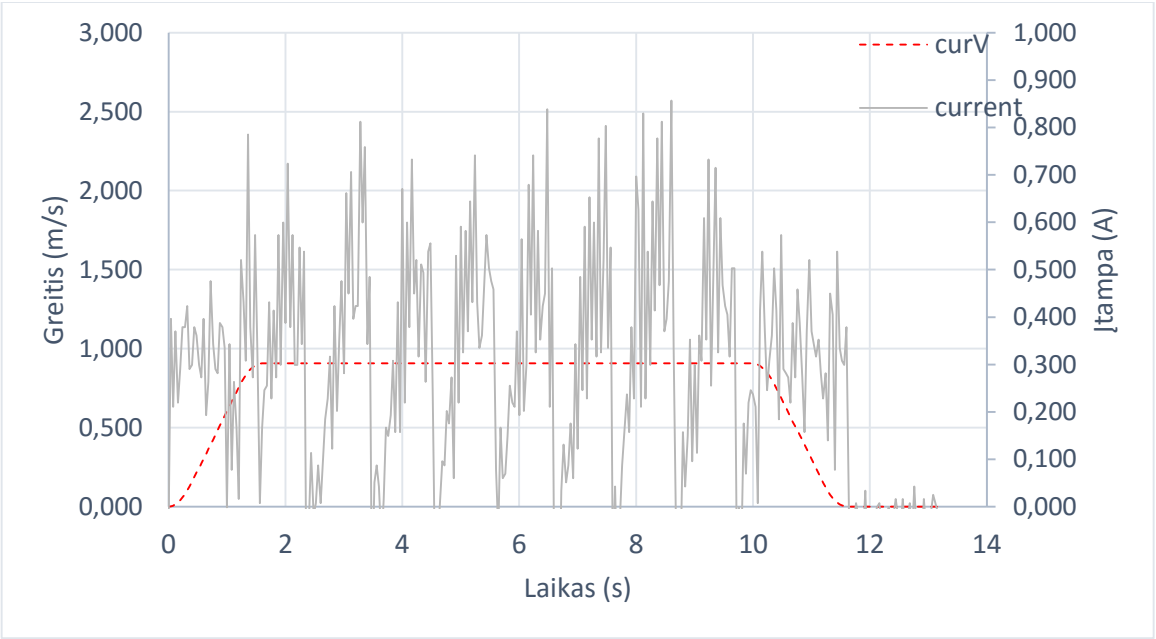
33 pav. Sistemos duomenų fragmentas

33 paveiksle pavaizduota dalis surinktų sistemos duomenų. Kiekvienas stulpelis reiškia skirtingus sistemos duomenis, kurių rinkimas aprašytas programos kode, Arduino programos aplinkoje. Kad pamatytume kaip atrodo traukinio greičio priklausomybės nuo laiko grafikas, reikalingi pirmas ir trečias stulpeliai. Pirmame stulpelyje nurodomas sistemos veikimo laikas, o trečiame – greitis.



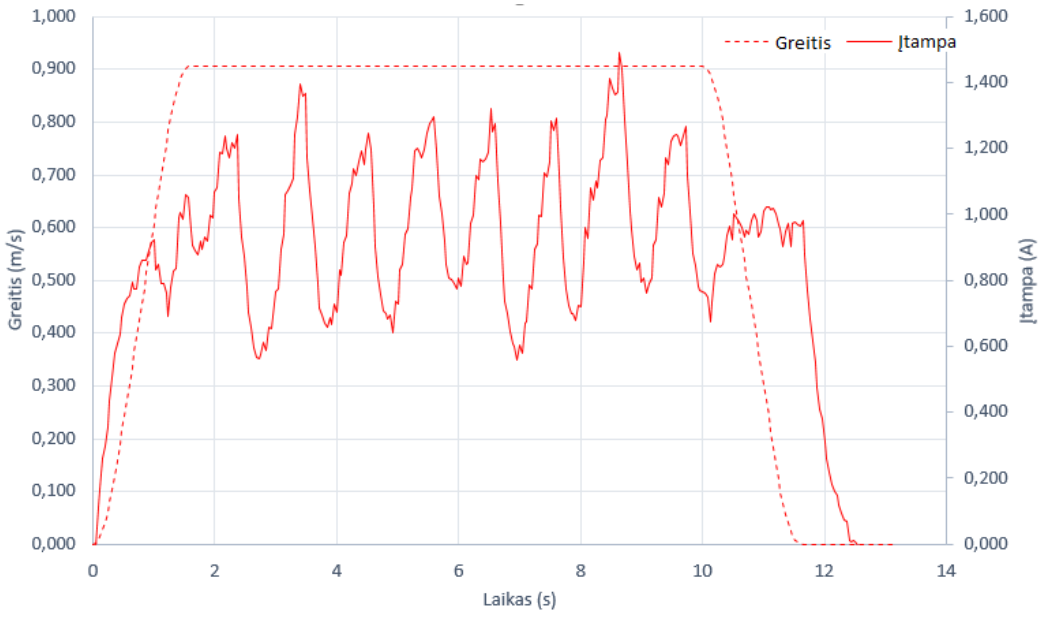
34 pav. Greičio priklausomybė nuo laiko

Taigi, sudarius grafiką (34 pav.) galima matyti, kad elektrinio traukinio modelis judėjo pagal S-kreivės greičio profilį. Kadangi pagrindinis darbo tikslas yra elektros energijos suvartojimas, panaudojus surinktus sistemos veikimo duomenis, nubraižomas ir energijos suvartojimo grafikas (34 pav.)



35 pav. Sistemos sunaudojamos srovės grafikas

Kaip matyti iš grafiko, srovės stipris labai nepastovus. Kad būtų paprasčiau vertinti gautus duomenis, srovės matavimo duomenys išfiltruojami, taip panaikinant grafiko šuolius atsirandančius dėl įvairių sistemos triukšmų. Išfiltruotas signalas pavaizduotas 36 paveiksle.

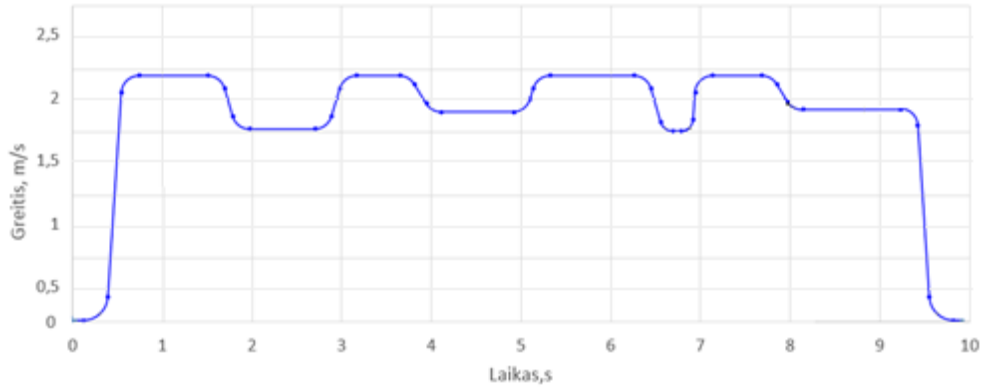


36 pav. Greičio ir naudojamos srovės grafikas

Kaip galima matyti iš išfiltruotų duomenų, sistema mažiausiai energijos suvartoja įgreitėjimo metu. Pastoviam greičio palaikymui suvartojamas didžiausias energijos kiekis. Lėtinant, kitaip nei būtų galima pamanyti, energijos suvartojama panašiai kaip ir greitėjimo metu. Taip yra dėl PWM moduliacijų. Jeigu srovės tiekimas būtų paprasčiausiai nutraukiamas, variklis staiga sustotų, dėl mažo

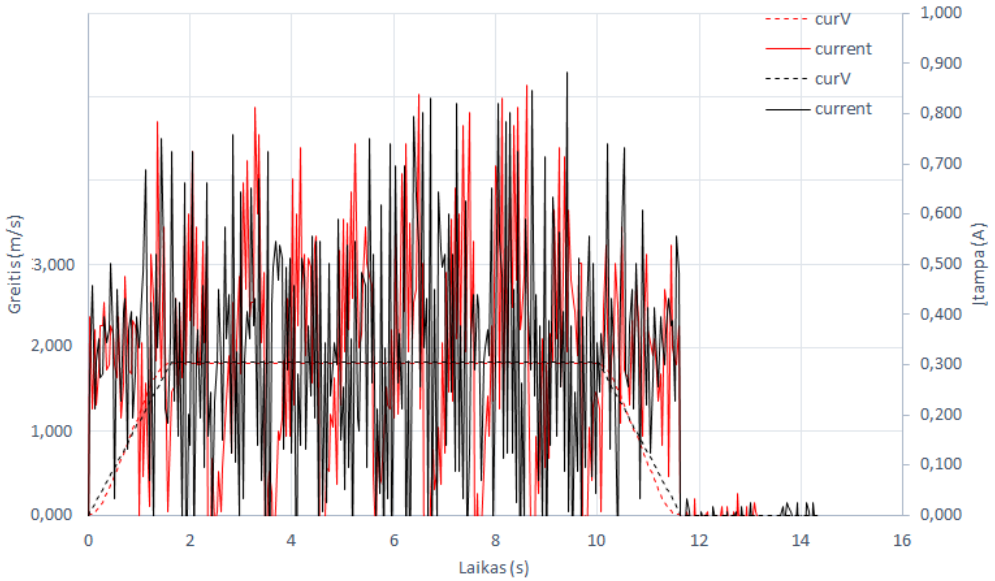
traukinio modelio svorio. O tam, kad variklio sukimasis lėtėjimo metu nenutrūktų, reikalinga elektros srovė.

Tam, kad būtų galima palyginti, kaip elektros energija vartojama naujoje sistemoje palyginus su senąja, reikalinga suprogramuoti ir pritaikyti naują energijos sąnaudas mažinančią sistemą visam bėgių žemėlapiui. Naujos sistemos greičio priklausomybės nuo laiko grafikas pavaizduotas 37 paveiksle.



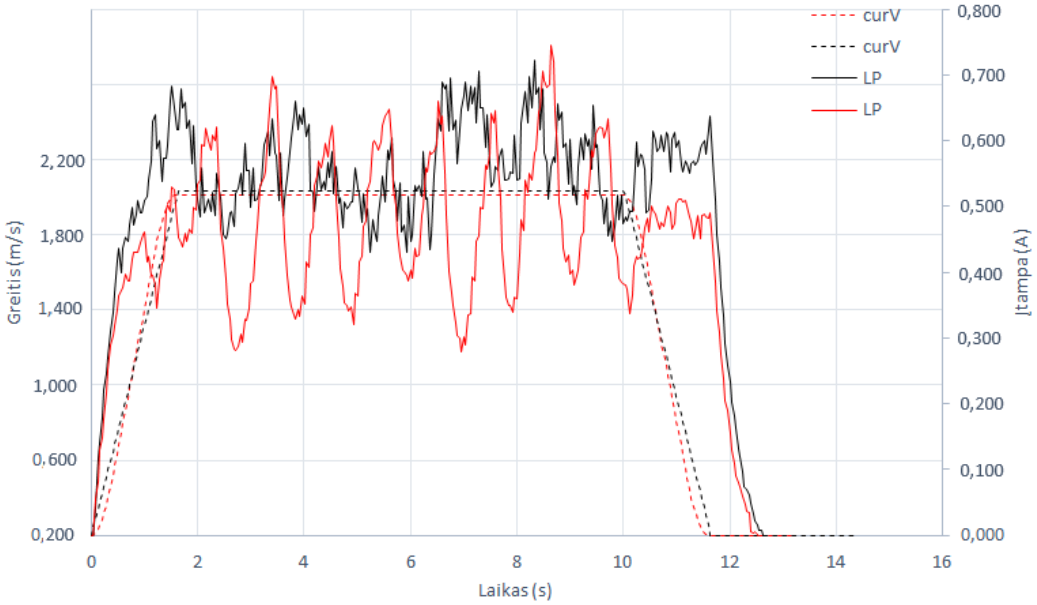
37 pav. Sistemos su S-kreivės profiliais greičio grafikas

Sistemos efektyvumo patikrinimui paimamas vienas greičio ruožas. Vieno važiavimo metu jis užprogramuojamas veikti trapecinio greičio profilio pagrindu, o kitu važiavimu S-kreivės tipo pagrindu. Abiejais važiavimais renkama sistemos informacija. 38 paveiksle pateikiami iš gautų duomenų sudaryti grafikai.



38 pav. Skirtingų profilių palyginimas

Kadangi gautus duomenis tokiame grafike sudėtinga vertinti, pritaikomas duomenų filtras (39 pav.).



39 pav. Išfiltruoti tyrimo rezultatai

39 paveiksle raudonai pavaizduoti duomenys susiję su S-kreivės greičio profiliu, o juodai pavaizduoti trapecinio greičio profilio duomenys. Galima matyti, kad S-tipo profiliu valdomam traukinio modeliui įgreitėjiant reikia iki 0,5 A mažiau, o lėtėjant iki 0,3 A mažiau elektros energijos. O štai pasiektam maksimaliam greičio išlaikymui reikalingas energijos kiekis, abiejais profilias valdomam elektrinio traukinio modeliui, yra panašus.

Atlikus šį tyrima galima teigti, kad sistema veikia kaip ir tikėtasi. Elektrinio traukinio modelis valdomas S-kreivės tipo profiliu važiuoja tolydžiau, greitėjant ir lėtėjant apkrovos tenkančios varikliui yra mažesnės, o dėl to sumažėja elektros energijos sąnaudos.

IŠVADOS

1. PID reguliatoriaus derinimas yra palyginus paprastas, nes yra tik 3 derinami parametrai, tačiau visiškam tikslumui pasiekti reikalingas ilgas ir sudėtingas derinimo procesas. Greičio profiliai yra lengvai derinami ir gana tikslūs, o jų parametrų parinkimas nesudėtingas. Neraiškiosios logikos reguliatoriaus derinimo procesas yra ilgas, nes reikia įvertinti daug parametrų. Ir nors šio valdiklio dėka galima pasiekti didelį valdomos sistemos tikslumą, reikia daug pastangų, kad jis veiktų tinkamai. Taigi apibendrinus šiuos aspektus nutarta tolimesniam elektrinio traukinio greičio kontrolės sistemos kūrimui naudoti greičio profilius.
2. Išanalizavus įtolinto valdymo sistemas taikomas automatiniame valdyme, buvo surasti jų pagrindiniai privalumai ir trūkumai. Atsižvelgiant į šiuos pastebėjimus buvo pasirinktas *bluetooth* bevielis duomenų perdavimo ryšys. Tokį pasirinkimą įtakojo šios technologijos paprastumas, prieinamumas bei techninės galimybės, dėl kurių ši technologija gali būti pritaikoma projektuojamoje sistemoje.
3. Sukurta greitojo traukinio modelio greičio valdymo sistema. Ji buvo kuriama jau esančios greičio kontrolės sistemos pagrindu atsižvelgiant į greičio apribojimus ir valdymo algoritmus, tačiau keičiant greičio pokyčius įgreitėjimo ir sulėtėjimo metu taip, kad būtų užtikrintas traukinio judėjimo tolydumas ir mažinamos energijos sąnaudos.
4. Atliekant greičio kontrolės patikrą eksperimentiniu būdu prie elektrinį greitąjį traukinį valdančio Arduino valdikio buvo prijungtas srovės matuoklis ASC712, kuriuo buvo matuojama traukinio judėjimui reikalinga suvartojama elektros srovė. Atlikus matavimus nustatyta, jog nauja greičio valdymo sistema suvartoja mažiau energijos dėl tolydaus judėjimo. Palyginimui išmatuota kaip elektrinis traukinys vartoja elektros energiją su senąja valdymo sistema, kurioje įgreitėjimai ir sulėtėjimai įvykdavo labai greitai, be jokio tolydaus perėjimo iš vieno greičio į kitą. Nustatyta, kad su nauja valdymo sistema elektrinis traukinys suvartoja mažiau energijos dėl mažesnių apkrovimų, kurios sumažėja panaikinus staigius greičio pokyčius. S-tipo profiliu valdomam traukinio modeliui įgreitėjiant reikia iki 0,5 A mažiau, o lėtėjant iki 0,3 A mažiau elektros energijos. O štai pasiektam maksimaliam greičio išlaikymui reikalingas energijos kiekis, abiejais profilias valdomam elektrinio traukinio modeliui, yra panašus.

LITERATŪRA

1. M.K Hassan, N.A.M. Azubir, Nizam H.M.I, S.F Toha ir B.S.K.K Ibrahim. A power-saving control strategy for reducing the total pressure applied by the primary air fan of a coal-fired power plant. Addison Wesley, 2016.
2. K. M. Passino, S. Yurkovich. *Fuzzy Control*. (ICSET), pp. 1-6, 2009. ISBN 0-201-18074-X.
3. R. Liutkevicius, V. Barzdaitis. *Netiesinio laboratorinio proceso „fuzzy“ reguliatorius*. Informacinė visuomenė ir universitetinės studijos. 2001 konferencijos pranešimu medžiaga.
4. J Abonyi. *Fuzzy Model Identification for Control*. Boston, 2003. ISBN 0-8176-4238-2, ISBN 3-7543-4232-2.
5. http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html. Fuzzy Logic Tutorial, 2008. Paskutinė lankymosi data: Gegužės 15, 2016. Informacijos mokslai [Tinkle]
6. http://internet.ktu.lt/lt/mokslas/zurnalai/elektr/z64/Valiulio%20str.pdf. Skysčio lygio valdymas taikant „fuzzy“ logika, 2008. Paskutinė lankymosi data: Gegužės 15, 2016. Informacijos mokslai [Tinkle]
7. http://ifveikla.vdu.lt/uploads/File/Data/FuzzyLogikaSistemuValdymui.exe. Fuzzy Logika Sistemų Valdymui, 2008. Paskutinė lankymosi data: Gegužės 15, 2016. Informacijos mokslai [Tinkle]
8. Z. Kovacic, S. Bogdan. *Fuzzy Controller Design Theory and Applications*. CRC Press, 2006.
9. P. Albertos, A. Sala. *Multivariable Control Systems: An Engineering Approach (Advanced Textbooks in Control and Signal Processing).* Springer-Verlag, 2010. London.
10. G. Chen, T.T. Pham. *Introduction to Fuzzy Sets, Fuzzy Logic and Fuzzy Control Systems*. CRC Press, USA, 2001R. Sam, K. Arrifin, N. Buniyamin „Simulation of Pick and Place Robotics System Using Solidworks Softmotion“ *International Conference on System Engineering and Technology* (ICSET), pp. 1 – 6, 2012.
11. Yang Yu „The Parametric design and inteligent assembly system based on the secondary development of solidworks“ *IEEE Conference Publications on Computer Engineering and Technology (ICCET),*pp. 602-605, 2010.
12. L. Zhao, S. Zhao, „Virtual Design of Bike Based on SolidWorks“ *International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)*t. 1, pp. 129 -132, 2011.
13. Xia Dayong, Wu Zhen, Chen Kai, He Xiaoying, Huang Lei, Wang Yiduo „Virtual Protorype modeling of a Cable crane and Simulation to its Passing Ability“*IEEE Conference Publications on Mechanic Automation and Control Engineering (MACE),* pp. 507-510, 2010.

14. R. Sam, K. Arrifin, N. Buniyamin „Simulation of Pick and Place Robotics System Using Solidworks Softmotion“ *International Conference on System Engineering and Technology* (ICSET), pp. 1-6, 2012.

15. Yang Yu „The Parametric design and inteligent assembly system based on the secondary development of solidworks“ *IEEE Conference Publications on Computer Engineering and Technology (ICCET),* pp. 602-605, 2010.

16. C.-L. Lai. „Design the Remote Control System With the Time-Delay Estimator and the Adaptive Smith Predictor“. *IEEE transactions on industrial informatics,* t. 6, nr. 1, pp. 73-80, 2010.

17. R. Bayindir and Y. Cetinceviz. „A water pumping control system with a programmable logic controller (PLC) and industrial wireless modules for industrial plants — An experimental setup“. *ISA Transactions 50,* no. 50, pp. 321-328, 2011.

18. H. Gulati, S. Vaishya ir S. Veeramachaneni. „Bluetooth and Wi-Fi Controlled Rescue Robots“. *India Conference (INDICON),* Hyderabad, 2011.

19. X. C. I. Frances Cleveland. „Uses of the New Types of Wireless Technologies for Distribution and Substation Automation“. 30p., 2007. [interaktyvus]. [žiūrėta 2012 05 04] http://xanthus-consulting.com/Publications/documents/Wireless_in_DA_and_SA.pdf.

20. L. Zhao, S. Zhao, „Virtual Design of Bike Based on SolidWorks“ *International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)* T. 1, pp. 129-13, 2011.

21. Ning Xiao-bo, Jiang Quan-sheng. „A digital design method of geometric model for centrifugal fan impeller based on SolidWorks and VB“. *Electronic and Mechanical Engineering and Information Technology (EMEIT),* 2011 *International Conference on China*: 2011, p. 40234026.

22. Xin-wu Du, Shi-Tong Jia; Jiang-tao Ji, Xiu-Fang Yu; Zhi-tao He; Zhi-hua Zheng; Ya-kai He;

23. R. Sam, K. Arrifin, N. Buniyamin „Simulation of Pick and Place Robotics System Using Solidworks Softmotion“ *International Conference on System Engineering and Technology* (ICSET), pp. 1 – 6, 2012.

24. Bertašius Bronius. *Traukinio greičio reguliavimo sistemos kūrimas panaudojant simuliacijos duomenis.* Klaipėda, Klaipėdos Universitetas, 2014,

PRIEDAI

1 Priedas

Programinis kodas

```
#include <TimerOne.h>
#include <stdlib.h>

#define ENC_Y_KOEF  1.068 // 1 imp [mm]
#define DUTY        512
#define STEPSREV    200
#define DT          10     // ms
#define DTLOG       40     // ms
#define DTSPEED     50
#define epsilon ((float)0.000001)

class Motor {
    typedef enum { CW, CCW } Direction;
    String name;
    boolean useTimer3 = false;

    int pinEn, pinStep, pinDir; // pins

    float d; // būgno skersmuo mm
    int stepSize = 2; // motor driver step size Step size: 1 / [1, 2,
8, 16]
    float gearRatio;
    float systemKoef;



    Direction curDir;
    boolean invertAxis;

    float maxV;
    float targetV = 0.0;
    float stopV = 0.0;
    int jerkState = 0;
    unsigned long curDT = 0;
    unsigned long previousMillis;
```

```
    float s = 0.5;
    float t_accel = 0.0;
    float t_idle = 0.0;
    float a_vid = 0.0;
    float a_max = 0.0;
    float j = 0.0;

    float t_st1 = 0.0;
    float t_st2 = 0.0;
    float t_st3 = 0.0;

    boolean trapezoidal = true;

    unsigned long time_st_changed = 0;

  public:
    typedef enum { HOLD, J1, J2, J3, RUN, J5, J6, J7 } State;
    State curState;
    float curJ = 0.0;
    float curV = 0.0;
    float curA = 0.0;
    float curX = 0.0;
    float stopDist = 0.0;
    boolean flagMove = false;

//Motor y ("Y" , true, 4, 5, 9, 0.16, 2.4, 26.0, 0.22, 2);
    Motor(String name, boolean invertDir, int en, int dir, int step,
float maxSpeed, float duration_accel, float diameter, float gearRadio,
int stepSize) {
      this->name = name;
      this->invertAxis = invertDir;
      this->pinEn = en;
      this->pinDir = dir;
      this->pinStep = step;
```

```
      pinMode(pinEn, OUTPUT);
      pinMode(pinDir, OUTPUT);
      pinMode(pinStep, OUTPUT);

      this->maxV = maxSpeed;
      this->t_accel = duration_accel;
      this->d = diameter;
      this->gearRatio = gearRadio;
      this->stepSize = stepSize;

      systemKoef = (15708 * diameter) / (5 * STEPSREV * stepSize *
gearRatio);

      a_vid = maxV / t_accel;
      a_max = a_vid / (1 - s * 0.5);
      j = 2 * a_max / (t_accel * s);

      setTrapezoidalMode(trapezoidal);

      this->previousMillis = millis();
      setDir((invertDir) ? CCW : CW);
      hold();
    }

    /** CHANGE STATE
**************************************************************************
**************************************************************************
**/
    void hold() {
      setState(HOLD);
      setJerk(0.0);
      Timer1.disablePwm(pinStep);
      //      Serial.println("PWM DISABLED");
    }

    void startCW() {
        setDir(CW);
```

```
        previousMillis = millis();
        log(previousMillis);
        setState(J1);
    }

    void startCCW() {
        setDir(CCW);
        previousMillis = millis();
        log(previousMillis);
        setState(J1);
    }

    void run() {
      setState(RUN);
      setJerk(0.0);
      updatePWM();
    }

    void stop() {
      if (getState() != HOLD) {
        //        Serial.println("MOTOR STOP CMD");
        previousMillis = millis();
        setState(J5);
      }
    }

    /** UPDATE
**************************************************************************
*****************************************************/

    void update() {
      unsigned long currentMillis = millis();

      curDT = currentMillis - previousMillis;

      switch (getState()) {
```

```
      case HOLD:
        if (curDT >= DT) {
          setJerk(0.0);
          previousMillis = currentMillis;
        }
        break;
      case J1:
        if (curDT >= DT) {
          setJerk(j);
          updatePWM();

          if (currentMillis - time_st_changed >= t_st1) {
            setState(J2);
            //              Serial.println("J1 END ");
          }

          previousMillis = currentMillis;
        }
        break;
      case J2:
        if (curDT >= DT) {
          setJerk(0.0);
          updatePWM();

          if (currentMillis - time_st_changed >= t_st2) {
            setState(J3);
            //              Serial.println("J2 END ");
          }

          previousMillis = currentMillis;
        }
        break;
      case J3:
        if (curDT >= DT) {
          setJerk(-j);
```

```
      updatePWM();

      if (currentMillis - time_st_changed >= t_st3) {
        //              Serial.println("J3 END ");
        run();
      }

      previousMillis = currentMillis;
    }
    break;
  case RUN:
    if (curDT >= DT) {
      setJerk(0.0);
      previousMillis = currentMillis;
    }
    break;
  case J5:
    if (curDT >= DT) {
      setJerk(-j);
      updatePWM();

      if (currentMillis - time_st_changed >= t_st1) {
        //              Serial.println("J5 END ");
        setState(J6);
      }

      previousMillis = currentMillis;
    }
    break;
  case J6:
    if (curDT >= DT) {
      setJerk(0.0);
      updatePWM();

      if (currentMillis - time_st_changed >= t_st2) {
```

```
            //              Serial.println("J6 END ");
            setState(J7);
          }

          previousMillis = currentMillis;
        }
        break;
      case J7:
        if (curDT >= DT) {
          setJerk(j);
          updatePWM();

          if (currentMillis - time_st_changed >= t_st3) {
            //              Serial.println("J7 END ");
            hold();
          }

          previousMillis = currentMillis;
        }
        break;
    }
  }


 private:
   void setJerk(float jerk) {
     if ((getDir() == CCW && !invertAxis) || (getDir() == CW &&
invertAxis)) jerk = -jerk;


     float dt = curDT / 1000.0;
     curJ = jerk;
     if (this->trapezoidal) { // trapecinis
       curA = a_vid;
       if ((getDir() == CCW && !invertAxis) || (getDir() == CW &&
invertAxis)) curA = -curA;
       if (getState() == J6) curA = -curA;
     } else { // s-shaped
```

```
        curA += curJ * dt;
      }

      if (getState() == HOLD || getState() == RUN) {
        curJ = 0.0;
        curA = 0.0;
        if (getState() == HOLD) curV = 0.0;
      }
      curV += curA * dt + curJ * dt * dt / 2;
      curX += curV * dt + curA * dt * dt / 2 + curJ * dt * dt * dt /
6;

      //      float jerk_stop = ((getDir() == CCW && !invertAxis) ||
(getDir() == CW && invertAxis)) ? -maxJ : maxJ;
      //      float t_stop = -curA / jerk_stop;
      //      stopV =  abs(curA * t_stop + jerk_stop * t_stop * t_stop
/ 2);

      //      float decel = ((getDir() == CCW && !invertAxis) ||
(getDir() == CW && invertAxis)) ? -maxDecel : maxDecel;
      //      float t = -curV / decel;
      //      stopDist = abs(curV * t + decel * t * t / 2);
      stopDist = 0.0;
      //      previousMillis = currentMillis;
      //log(millis());
    }

    void updatePWM() {
      Timer1.pwm(pinStep, DUTY, getPeriod(curV));
    }

    unsigned int getPeriod(float v) {
      int period = int(systemKoef / abs(v));
      if (v != 0.0) return period;
      else return 0;
    }
```

```
/*************************************************************************
*************************************************************************
****/
  public:
    void setState( State newState) {
      this->curState = newState;
      time_st_changed = millis();
    }
    void setDir( Direction newDir) {
      this->curDir = newDir;
      (newDir == CW) ? digitalWrite(pinDir, LOW) :
digitalWrite(pinDir, HIGH);
    }
    State getState() {
      return this->curState;
    }
    Direction getDir() {
      return this->curDir;
    }




    void setTrapezoidalMode(boolean mode) {
      this->trapezoidal = mode;
      if (this->trapezoidal) { // trapecinis
        t_st1 = 0;
        t_st2 = t_accel * 1000;
        t_st3 = t_st1;
      } else { // s-shaped
        t_st1 = t_accel * s / 2 * 1000;
        t_st2 = t_accel * (1 - s) * 1000;
        t_st3 = t_st1;
      }
    }
```

```
}; /*** End of Motor class ******************************************/



/*** GLOBAL
*********************************************************************
**/


/*** ENCODER *******/

const int pinEncY = 2;


volatile int impY = 0;              // for speed calculation

volatile long posY = 0;             // imps

volatile boolean PastYB = 0;

volatile float realPosY = 0.0;  // m

volatile float speedY = 0.0;      // m/s


/*** TIMERS *******/

unsigned long lastLog, lastSpeedSensor = 0;


unsigned long startTime = 0;

unsigned long endTime = 0;

boolean moving = false;


boolean sendLog = false;


String inputString = "";

boolean stringComplete = false;

//Motor1 = Y axis        Motor2 = Z axis
f          f             f           f          f        int
int      boolean               int           int

//Motor [xyz] (name, invertAxisDir, en, dir, step, stopCW, stopCCW,
maxSpeed, acceleration, deceleration, diameter, gearRadio, stepSize,
trosoKoef, useTimer3)

Motor y ("Y" , false, 4, 5, 9, 2.3, 0.2, 26.0, 0.22, 2); // CCW
positive


/*** FUNCTIONS
*********************************************************************
**/
```

```
/*** ENCODER ******************/
void intYA() {
  posY++;
  impY++;
  realPosY = posY * ENC_Y_KOEF / 1000.0;
}
void speedSensor(unsigned long now) {
  unsigned long diff = now - lastSpeedSensor;
  if (diff >= DTSPEED) {
    speedY = impY * ENC_Y_KOEF / diff;
    impY = 0;
    lastSpeedSensor = now;
  }
}

/*** LOG ******************/
void log(unsigned long now) {
  if ((now - lastLog) >= DTLOG && sendLog) {
    Serial.print(now);
    Serial.print(" ");
    Serial.print(y.curA, 3);
    Serial.print(" ");
    Serial.print(y.curV, 3);
    Serial.print(" ");
    Serial.print(y.curX, 3);
    Serial.print(" ");
    Serial.print(speedY, 3);
    Serial.print(" ");
    Serial.print(realPosY, 3);
//    Serial.print(" ");
//    Serial.print(y.curJ, 3);
    Serial.print(" ");
    Serial.print(analogRead(0)*1.0, 3);
    Serial.print(" ");
```

```
    Serial.print(((analogRead(0)+77)*(5.02/1023.0)-2.5)/0.185/10.0,
3);
    Serial.println();
    Serial.flush();
    lastLog = now;
  }
}

void toggleLog(boolean newState = false) {
  if (newState)
    sendLog = newState;
  else
    sendLog = !sendLog;

  if (sendLog) {
    y.curX = 0.0;
    y.curV = 0.0;
    y.curA = 0.0;
    y.curJ = 0.0;
    posY = 0;
    realPosY = 0.0;
  }
}

void SerialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar != '\r' && inChar != '\n') inputString += inChar;
    if (inChar == '\n') stringComplete = true;
  }
}

bool isEqual(float x, float y) {
  return (abs(y - x) < epsilon);
}
```

```
boolean trapezoidal = false;


/*** START
HERE**********************************************************************/
void setup() {


  Timer1.initialize(2000);
  Serial.begin(115200);


  /* ENCODER Y*/
  pinMode(pinEncY, INPUT);
  digitalWrite(pinEncY, HIGH);
  attachInterrupt(digitalPinToInterrupt(pinEncY), intYA, RISING);


  inputString.reserve(16);
}


void loop() {
  unsigned long now = millis();
  SerialEvent();
  if (stringComplete) {
    if (inputString == "go") {
      toggleLog(true);
      moving = true;
      startTime = now;
      y.startCW();
//      Serial.println("go, OK");
    } else if (inputString == "stop") {
      y.stop();
//      Serial.println("stop, OK");
    } else if (inputString == "kill") {
      y.hold();
    } else if (inputString == "log") {
      toggleLog();
    } else if (inputString == "ping") {
      Serial.println("Go AWAY");
```

```
    } else if (inputString == "help") {
      Serial.println("go, stop");
    } else {
      Serial.println("WTF?");
    }
    inputString = "";
    stringComplete = false;
  }
  if(moving && (now - startTime >= 10*1000)){
    y.stop();
    moving = false;
  }
  y.update();
  speedSensor(now);
  log(now);
}
```

1. priedas

Kompaktinis diskas su baigiamojo darbo elektronine versija

2. Kompaktinis diskas, kuriame pateikiami:
3. Darbo katalogas: „Bronius_Bertašius/..“.
4. Baigiamasis darbas: „../Bronius_Bertašius_baigiamasis_darbas.pdf“.