Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Trijų laisvės laipsnių judesio platformos prototipo sukūrimas ir tyrimas

Baigiamasis magistro projektas

Martynas Loveikis

Projekto autorius

Doc. dr. Leonas Balaševičius

Vadovas

Kaunas, 2022

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Trijų laisvės laipsnių judesio platformos prototipo sukūrimas ir tyrimas

Baigiamasis magistro projektas

Valdymo technologijos (6211EX014)

Martynas Loveikis

Projekto autorius

Doc. dr. Leonas Balaševičius

Vadovas

Doc. dr. Tomas Tekorius

Recenzentas

Kaunas, 2022

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Martynas Loveikis

Trijų laisvės laipsnių judesio platformos prototipo sukūrimas ir tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Martynas Loveikis

Patvirtinta elektroniniu būdu

Loveikis, Martynas. Trijų laisvės laipsnių judesio platformos prototipo sukūrimas ir tyrimas. Magistro baigiamasis projektas / vadovas doc. dr. Leonas Balaševičius; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studiju krypčių grupė): Elektronikos inžinerija, inžinerijos mokslai.

Reikšminiai žodžiai: RRS, lygiagretieji manipuliatoriai, trijų laisvės laipsnių, SimTools, simuliacija.

Kaunas, 2022. 51 p., priedų 11 p.

Santrauka

Baigiamojo magistro projekto tema – „Trijų laisvės laipsnių judesio platformos prototipo sukūrimas ir tyrimas“. Teorinės dalies apžvalgoje aptariami straipsniai susiję su magistro baigiamojo projekto tematika. Detaliai nagrinėjama, kas yra manipuliatoriai, kokios yra jų rūšys ir kokie galimi privalumai naudojant skirtingas rūšis. Minimi vieni iš pirmųjų mechanizmų, kurie panašūs į lygiagrečiąsias platformas istorijoje. Sprendžiama kokia lygiagretaus manipuliatoriaus architektūra yra tinkamiausia trijų laisvės laipsnių simuliacijos atkūrimui. Projektinėje dalyje pateikiamos suprojektuoto prototipo mechanizmo dalys, panaudoti komponentai ir kaip jie sujungiami tarpusavyje pagal suprojektuotas elektros schemas bei *SolidWorks* modelį. Taip pat pateikiama mikrovaldiklio *Arduino* prototipo valdymo etapų programos aplinkos kodas bei jo paaiškinimas. Paaiškinama kaip judesio platforma imituoja simuliacijos modelio judesius. Kokie nustatymai reikalingi atvirojo kodo programoje *SimTools*, jog prototipo judesiai būtų teisingai interpretuojami pagal generuojamus simuliacijos modelio nustatymus. Tiriamojoje dalyje atliekami teoriniai bandymai, kuriais nustatoma prototipo platformos kinematinių grandžių judėjimo amplitudė ir judėjimo greitaveikos bandymai. Teoriniams bandymams panaudota *Simulink* aplinkos *Simscape* modelis. Taip pat atlikti fiziniai greitaveikos, platformos posvyrių bei garso tyrimai. Toliau gauti bandymų rezultatai palyginti su rinkoje esančiomis judesio platformomis.

Loveikis, Martynas. Three degrees of freedom motion platform prototype development and testing. Master's final project / supervisor Assoc. Prof. Dr. Leonas Balaševičius. Kaunas University of Technology, Faculty of Electrical and Electronics Engineering.

Study field and area (study field group): Electronics Engineering, Engineering Science.

Keywords: RRS, parallel manipulators, three degrees of freedom, SimTools, simulation.

Kaunas, 2022. 51 p., extra 11 p.

Summary

The topic of the Final master's thesis – „Three degrees of freedom motion platform prototype development and testing". The review of the theoretical part examines the articles related to the topic of the master's thesis. Explanation of a manipulator and the possible perks of using various types are scrutinized in detail. Early mentioning of mechanisms related to parallel platforms. Determination of which parallel manipulator architecture is the most applicable for mimicking three degrees of freedom simulation. The design part shows the modelled parts of the prototype mechanism and how they join each other according to the *SolidWorks* model. As well as how the components hook up in between according to the designed wiring diagrams. Control stages of the prototype in the *Arduino* coding environment is explored in detail. Provided explanations of how the motion platform shifts according to the simulation model. Settings which are required for the open source *SimTools* program to correctly translate the prototype movements delivered from the simulation model are also given. In the research part theoretical tests were done to determine the amplitude of motion of the kinematic chains present in the prototype platform and motion speed. A developed *Simscape* model of the *Simulink* environment was used to accomplish theoretical tests. Real platform tests of speed, inclination size, sound level were also performed. The following test results are compared with the motion platforms available on the market.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

LM – lygiagretusis manipuliatorius;

NM – nuoseklusis manipuliatorius;

PWM (Pulse Width Modulation) – impulso pločio moduliacija;

UART (Universal Asynchronous Receiver – Transmitter) – universalus asinchroninis imtuvas – siųstuvas;

ASCII (American Standart Code for Information Interchange) – amerikietiškas informacijos mainų koduotės standartas;

PLA (Polylactic Acid) – polilakto rūgštis;

CAD (Computer Aided Design) – dizainas padarytas kompiuterio pagalba;

XML (Extensible Markup Language) – išplėstinė žymėjimo kalba;

RMS (Root Mean Square) – kvadratų vidurkio kvadratinė šaknis;

RRS (Revolute – Revolute – Spherical) – šarnyrinė – šarnyrinė – sferinė;

RPS (Revolute – Prismatic – Spherical) – šarnyrinė – prizmatinė – sferinė;

PRS (Prismatic – Revolute – Spherical) – prizmatinė – šarnyrinė – sferinė;

PPS (Prismatic – Prismatic – Spherical) – prizmatinė – prizmatinė – sferinė;

UPU (Universal – Prismatic – Universal) – universalioji – prizmatinė – universalioji;

UPS (Universal – Prismatic – Spherical) – universalioji – prizmatinė – sferinė;

1T2R (One Translational Two Rotational) – viena slenkančioji, dvi sukamosios.

Terminai:

Surge – slenkantysis judėjimas lygiagrečiai su O_x ašimi trimatėje erdvėje.

Sway – slenkantysis judėjimas lygiagrečiai su O_y ašimi trimatėje erdvėje.

Heave – slenkantysis judėjimas lygiagrečiai su O_z ašimi trimatėje erdvėje.

Roll – posvyris pagal O_x ašį trimatėje erdvėje.

Pitch – posvyris pagal O_y ašį trimatėje erdvėje.

Yaw – posvyris pagal O_z ašį trimatėje erdvėje.

Įvadas

Sparčiai augant pramogų industrijos didžiausiam žaidimų sektoriui, pastarųjų metų technologijų erdvės pažanga atkreipė kasdienių žmonių dėmesį į šią tendenciją ir jos kylantį potencialą. Siekiant vartotojui sukurti jutiminę patirtį įskaitant regėjimą, klausą, lytėjimą ir vestibulinį jausmą, simuliacinė erdvė suteikia galimybę patekti į dirbtinį pasaulį, kurį galima visiškai valdyti. Kompiuterinės programinės įrangos modeliavimas leidžia asmeniui sąveikauti su dirbtine trimate vaizdine ir kita jutimo aplinka, kuri šiandien kelia pramonės perversmą. Į virtualų pasaulį galima pakliūti su tokiais atradimais kaip virtualios realybės akiniais ir kostiumais, judesio perteikimo vairasvirtėmis, ir platformomis bei begale kitų įrenginių. Lygiagretieji manipuliatorių mechanizmai dažnai naudojami minėtosioms judesio platformoms. Jos gali tiksliau pozicionuoti, turi didesnę greitaveiką, standžiau valdomos ir mažiau sveria, lyginant su nuosekliaisiais manipuliatoriais. Nepaisant to, jog lygiagrečiosios platformos turi ribotą darbo zoną ir yra sudėtingesnės konstrukcijos, jos yra vienos iš populiariausių tarp judesių platformų. Dėl įvarių konstrukcijos rūšių galimybių bei todėl, kad norint patirti kuo realistiškesnį simuliuojamos erdvės pojūtį, reikia imituoti judesius kuo įmanoma tiksliau.

Šiame darbe yra lyginamos manipuliatorių grupės, nagrinėjama specifinė trijų laisvės laipsnių lygiagrečios platformos *RRS* rūšis bei palyginamas skirtumas tarp Stiuarto platformos. Toliau atliekamas prototipo *SolidWorks* modelio ir elektros schemos projektavimas. Analizuojamas prototipo valdymo algoritmas *Arduino* programavimo aplinkoje. Galiausiai atliekamas prototipo teorinis tyrimas *Matlab Simulink* aplinkoje naudojant *Simscape* modelį ir palyginama su gautais fiziniais prototipo bandymais manipuliuojant su atvirojo kodo simuliacijos programa *SimTools*.

Darbo tikslas – sukurti ir ištirti trijų laisvės laipsnių judesio platformos prototipą.

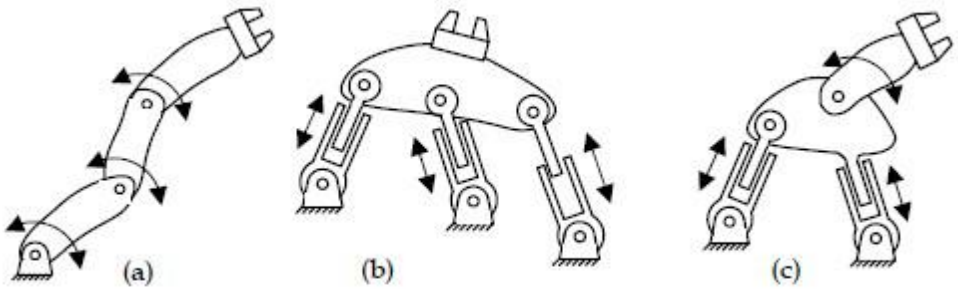
Darbo uždaviniai:

1. Atlikti literatūros analizę ir pagrįsti projektuojamo prototipo tipo pasirinkimą.
2. Suprojektuoti ir realizuoti judesio platformos prototipą.
3. Apžvelgti prototipo charakteristikų tyrimus ir palyginti juos su rinkoje esančių analogų.

1. Literatūros apžvalga

1.1. Manipuliatorių rūšys

Manipuliatoriai gali būti skirstomi į tris skirtingas grupes, priklausomai nuo to, kaip kinematinė struktūra juda. Pirmąją grupę galima apibendrinti kaip nuosekliuosius manipuliatorius, kurie susideda iš atvirų kinematinių grandžių. Prie antros grupės priskiriami uždaros kinematinės grandies lygiagretieji manipuliatoriai, kuriuose yra bent dvi jungtys kiekvienoje valdymo šakoje. O trečioji grupė vadinama hibridiniais manipuliatoriais, kadangi juose yra atvirų ir uždarų kinematinių grandies elementų [23]. Šios trys grupės pavaizduotos žemiau, pirmame paveikslėlyje.



1 pav. a) nuosekli, b) lygiagreti, c) hibridinė manipuliatoriaus rūšis [23]

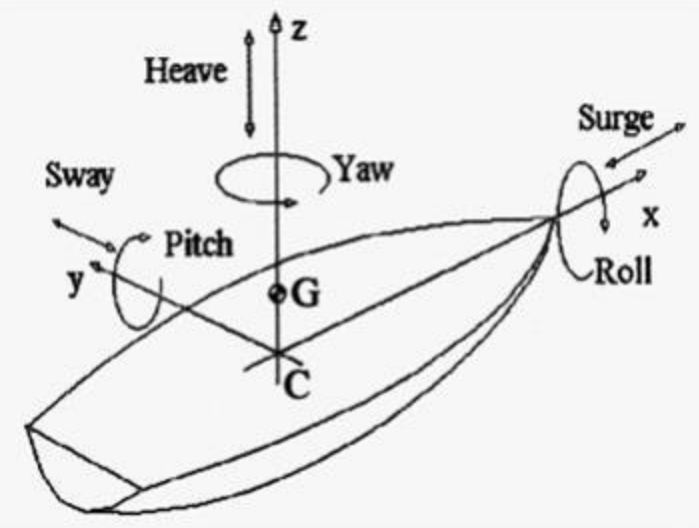
LM taip pat galima skirstyti į atskiras grupes, priklausomai nuo to, kokios jungtys naudojamos atskirose atšakų sąnariuose. Jeigu naudojama UPU, tada gaunamas tipinis *DELTA* robotas. Vadinamoji **U**– (angl. *universal*) universalioji sąnario jungtis, galinti suktis dviem atskiromis ašimis, naudojama prie platformos bei pagrindo. Vidurinis sąnarys yra **P**– (angl. *prismatic)* slenkantysis sąnarys kiekvienoje platformos atšakoje. Laisvės ašių simuliacijai dažnai naudojami PRS, RRS ir PPS lygiagretieji manipuliatoriai, [3] jeigu **R**– (angl. *revolute)* tai šarnyrinė jungtis, o **S**– (angl. *spherical)* yra sferinė jungtis.

Lyginant LM ir NM manipuliatorius, pastebimi keli LM pranašumai. Vienas iš pagrindinių pranašumų yra LM keliamoji galia. Palyginus keliamosios galios ir masės santykį tarp šešių laisvės laipsnių nuosekliojo ir lygiagretaus roboto galime gauti iki 66,66 kartų didesnę keliamąją galią, kadangi svoris pasiskirsto tolygiai kiekvienai platformos atšakai [6]. Taip pat LM turi dar šiuos privalumus lyginant su NM [6],[9],[23]:

1. tiksliau pozicionuoja;
2. turi geresnę standumo charakteristiką;
3. pasiekia didesnį greitį bei pagreitį;
4. mažesnio svorio.

Tačiau reikia atsižvelgti ir į tai, jog LM palyginus su NM turi tokius trūkumus kaip mažesnę darbo zoną ir singuliarumus toje pačioje darbo zonoje, o ne judėjimo zonos apribojimus kaip NM [23].

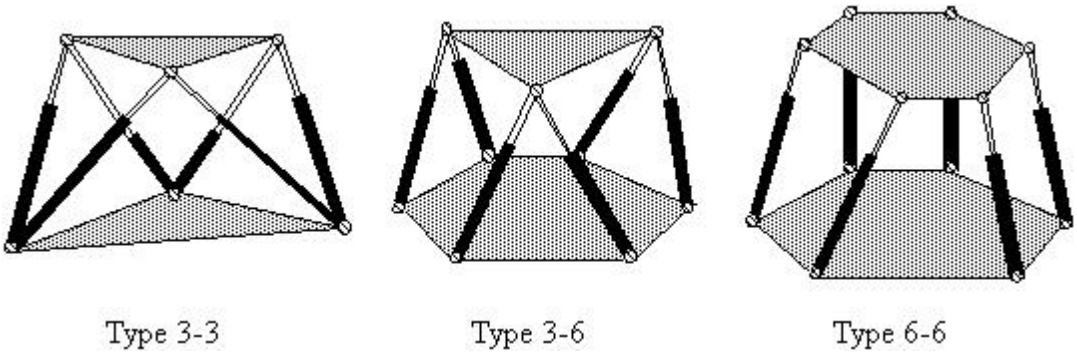
Judesių simuliacijai trijų dimensijų koordinačių plokštumoje galimi tik šeši laisvės laipsniai. Trys atskiros judėjimo kryptys lygiagrečiai *xyz* ašimis (angl. *Surge*, *Sway*, *Heave*) ir trys atskiri posvyriai aplink *xyz* ašis (angl. *Roll*, *Pitch*, *Yaw*) [25]. Kuo daugiau laisvės laipsnių, tuo mechanizmo konstrukcija tampa sudėtingesnė. Visi galimi laisvės laipsniai pavaizduoti antrame paveikslėlyje.



2 pav. Judesio simuliacijos ašys trijų dimensijų erdvėje [25]

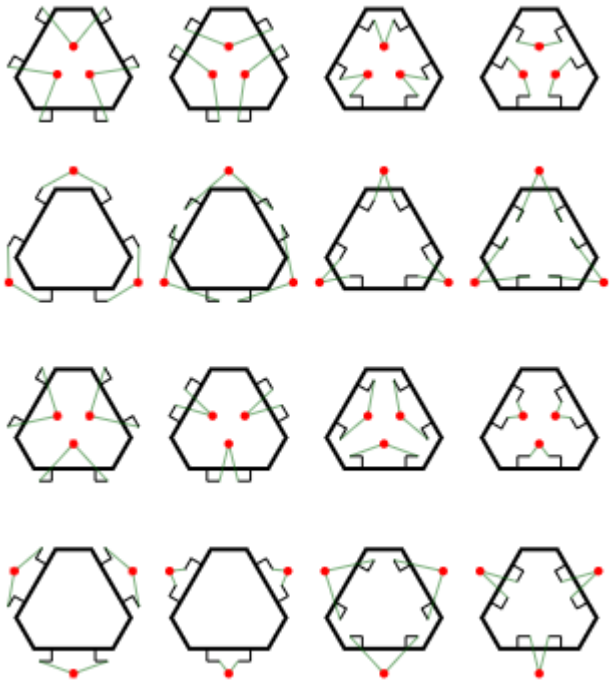
Judesio platforma, kuri gali atlikti simuliaciją šešiomis laisvės ašimis, paprastai yra Stiuarto platforma. Tokiai platformai judesį suteikia šeši vykdikliai atskirose platformos kojose. Praktikoje šie vykdikliai dažniausiai būna hidraulinės linijinės, elektromechaninės sraigtinės ir servo pavaros dėl aukštos keliamosios jėgos, tačiau ieškant pigesnių komponentų galima naudoti ir įprastus variklius[18].

Šio manipuliatoriaus sąnariams paprastai naudojamos UPS ir RPS grandys, priklausomai ar kojų vykdikliai yra linijinės arba rotacinės pavaros [18],[3]. Jeigu naudojamos UPS grandys, galima apibrėžti tris populiariausias geometrines konfigūracijas su linijinėmis pavaromis : 3–3, 3–6 ir 6–6 kaip parodyta trečiame paveikslėlyje žemiau [18].



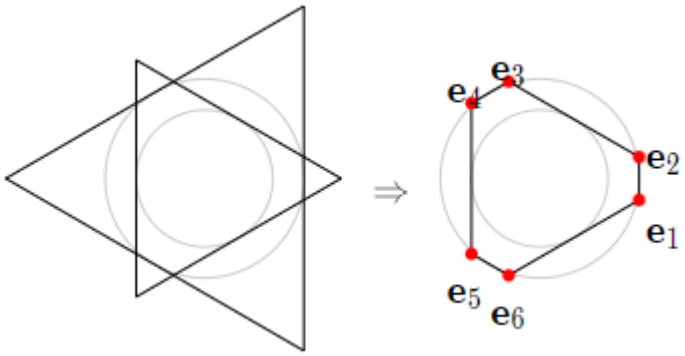
3 pav. Stiuarto platformų geometrinės konfigūracijos naudojant UPS grandis [18]

Taip pat naudojant RRS grandis, galima įsivaizduoti net šešiolika skirtingų geometrinių konfigūracijų kai naudojamos rotacinės pavaros. Ketvirtame paveikslėlyje vaizduojama kaip ant šešiakampio pagrindo įtaisyti servo rotaciniai vykdikliai ir žaliomis linijomis atvaizduojamos kojos sujungiančios platformą su pagrindu, o raudoni taškai – tai tvirtinimo taškai platformai. Atitinkamai platforma gali būti mažesnė nei pagrindas arba didesnė [8].



4 pav. Stiuarto platformų geometrinės konfigūracijos naudojant RRS grandis [8]

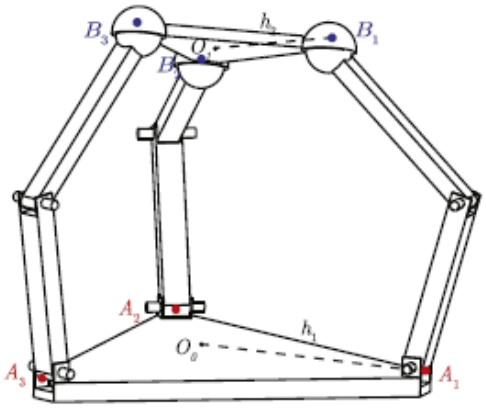
Žiūrint iš apačios, tokios formos šešiakampis pagrindas gautas apibrėžiant lygiakraščiais trikampiais viduryje esančius apskritimus iš kurių vieno spindulys yra dvigubai didesnis (žr. **5 pav.**). O žiūrint iš viršaus (žr. **4 pav.**) pačios platformos inkaravimo taškai yra įrengti kas 120 laipsnių [8].



5 pav. Šešiakampės platformos išdėstymas [8]

1.2. Prototipo manipuliatoriaus architektūra

Sprendžiant pagal prototipo galimybes ir konstrukcijos sudėtingumą ir kainą, trijų laisvės laipsnių RRS platforma yra viena iš paprasčiausių [3] ir gali turėti panašų efektą kaip ir Stiuarto platforma su dvigubai mažesniu kiekiu vykdiklių (žr. **6 pav.**). Laikantis tokio principo šio tipo platformos tampa žymiai prieinamesnės ir populiaresnės. Tokio grandies tipo platformos kitaip vadinamos 1T2R (angl. **T**–*translation*, **R**– *rotation*) turi vieną laisvės ašį, kuria galima platformą judinti koordinačių plokštumoje, šiuo atveju judėti aukštyn ir žemyn. Kitus du laisvės laipsnius turime platformos posvyriui atitinkamomis atskiromis ašimis – posvyriai į kairį arba dešinį šoną ir posvyris į priekį arba atgal (žr. **2 pav.**). 1T2R LM platformos paprastai naudojamos tokių įrenginių kaip simuliacijos platformos ir koordinačių matavimų prietaisų pozicionavimui [23].

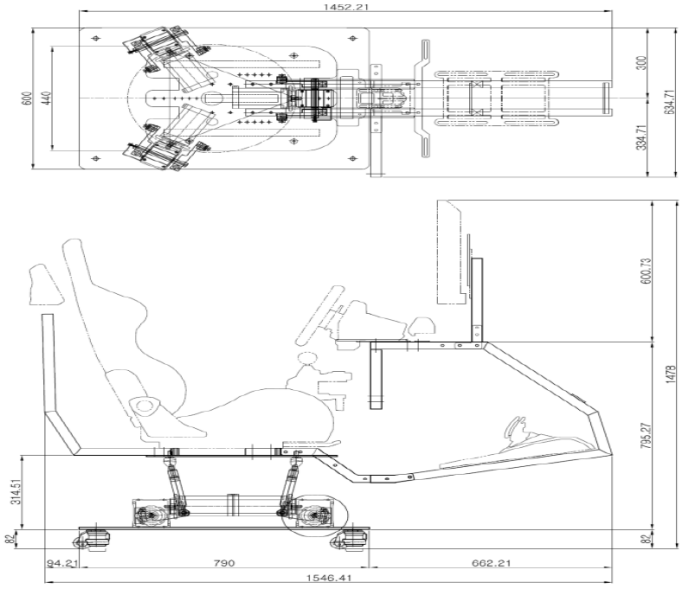


6 pav. Trijų RRS kinematinių grandžių LM architektūra [3]

Šios konstrukcijos tipo platformos bei pagrindo inkaravimosi taškai išdėstyti panašiai kaip Stiuarto platformos – lygiakraščio trikampio principu. Tačiau jos jau nebegalime vadinti Stiuarto platforma, kadangi trijų RRS kinematinių grandžių platforma yra tik tris laisvės laipsnius turinti konstrukcija, o ne visus šešis [18],[8].

Pagrindinis prototipo tikslas yra kuo artimiau imituoti simuliuojamos mašinos modelio judesius. Jeigu būtų konstruojamas realaus dydžio prototipas [15], reikia atkreipti dėmesį į tai, jog prototipas turi atspindėti tikrą sėdimą vietą mašinos modelyje. Tokiam tikslui prototipo platforma turėtų sėdimą vietą žmogui ir reikalingus įrenginius mašinos valdymui . Tai reiškia, jog turi būti įtaisytas sėdynės mechanizmas, pedalai su vairu bei ekranas kuriame būtų matoma pati simuliacija [22] kaip pavaizduota septintame paveikslėlyje [7].

Pačios platformos judėjimui svarbu turėti stiprios keliamosios jėgos variklius, o tam tikslui puikiai tinka, draugiški pradedantiesiems, servo varikliai, kurie veikia kaip žingsniniai varikliai naudojami šiame baigiamajame darbe. Servo varikliai gali turėti žymiai didesnį sukibimo momentą lyginant su žingsniniais varikliais, o tai ypač aktualu, kadangi reikalinga pakelti sunkią masę, palaikant pakankamą greitaveiką. Taip pat šie varikliai turi turėti vidinį grįžtamąjį ryšį, (angl. *encoder*) jog nebūtų praleidžiami žingsniai, kurie sukelia pozicionavimo klaidas [20].

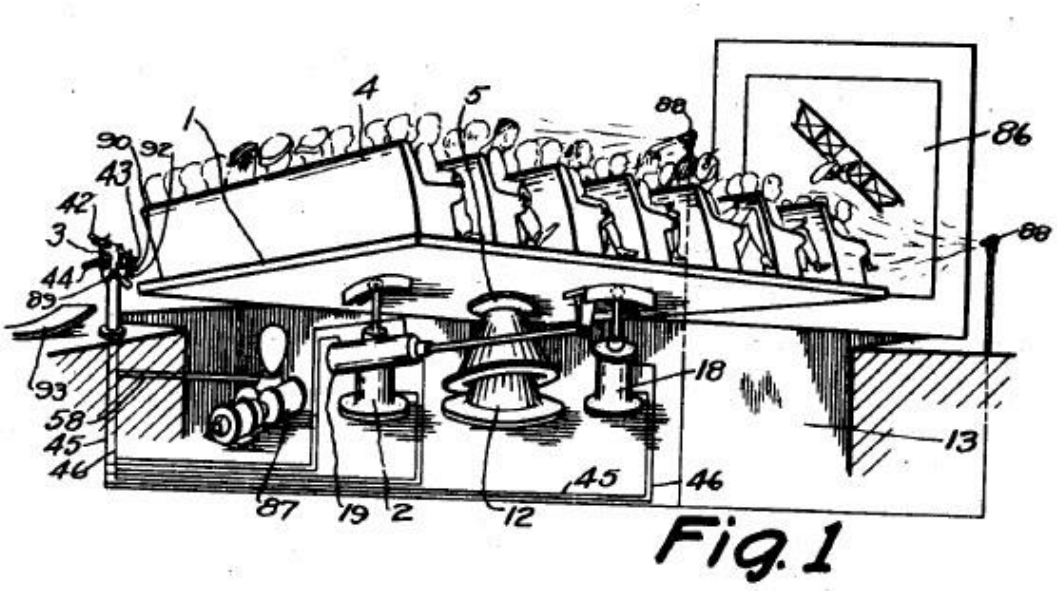


7 pav. Realaus prototipo naudotinos įrangos schema [7]

1.3. Lygiagrečiųjų manipuliatorių pirmtakai

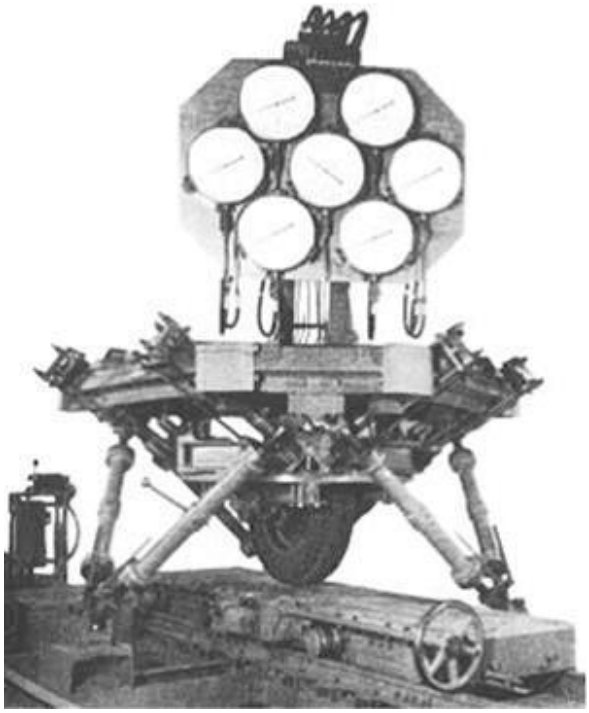
Žodis *Robotas* buvo pirmą kartą paminėtas dar 1920 metais, kada Čekų rašytojas Karel‘is Čapek‘as knygoje *Rossumovi univerzální roboti*, apibūdino masiškai gaminamus darbininkus, kurie tarnauja žmogui. Šis terminas kilęs iš čekų kalbos veiksmažodžio *robotovat,* kuris gali būti verčiamas kaip *vergauti* arba *dirbti*. Šiais laikais robotu taip pat galime vadinti tokį įrenginį, kuris gali atlikti užduotis vietoje žmogaus. Tokias užduotis, kurios dažnai kartojasi ir yra monotoniškos arba gali sukelti pavojų žmogui. Pagal iš anksto nustatytą valdymo algoritmą, valdant kito žmogaus nuotoliniu būdu arba autonomiškai [6]. Pagal Amerikos Robotikos Institutą, robotas yra perprogramuojamas plačios paskirties manipuliatorius, kuris gali perkelti medžiagas, įrenginių dalis, įrankius ar specializuotus įrenginius naudojant įvairius suprogramuotus judesius įvairioms užduotims atlikti. Įvairūs mechaniniai manipuliatoriai, skaitmeniniu būdu valdomos mašinos ir humanoidinės konstrukcijos įeina į šį apibrėžimą[23].

Pačius pirmuosius tyrimus susijusius su lygiagrečiaisiais manipuliatoriais galime atsekti tarp 17a. pabaigos ir 18a. pradžios, kada seras Christopher‘is Wren‘as, vienas iš labiausiai pripažintų anglų architektų istorijoje, paminėjo lygiagrečiosios struktūros mechanizmus savo moksliniuose tyrimuose. Toliau po jo, mokslininkas Augustin‘as–Louis‘as Cauchy‘is 18–19a. bei mokslininkai Henri‘s Lebesgue‘as ir Raoul‘is Bricard‘as 19–20a. dirbo su lygiagrečiaisiais mechanizmais [23]. Jau 1931 metais buvo pritaikytas pirmas praktinis panaudojimas pramogų industrijoje, kada James‘o Gwinnett‘o patentas skirtas kino salės žiūrovų platformos judėjimui, norint simuliuoti tikrą fizinį efektą, buvo patvirtintas. Iš septintojo paveikslėlio matoma, jog prie platformos yra pritvirtinti keli ašių valdymo mechanizmai, kurie gali judinti platformą įvairiomis kryptimis [6],[9],[23].



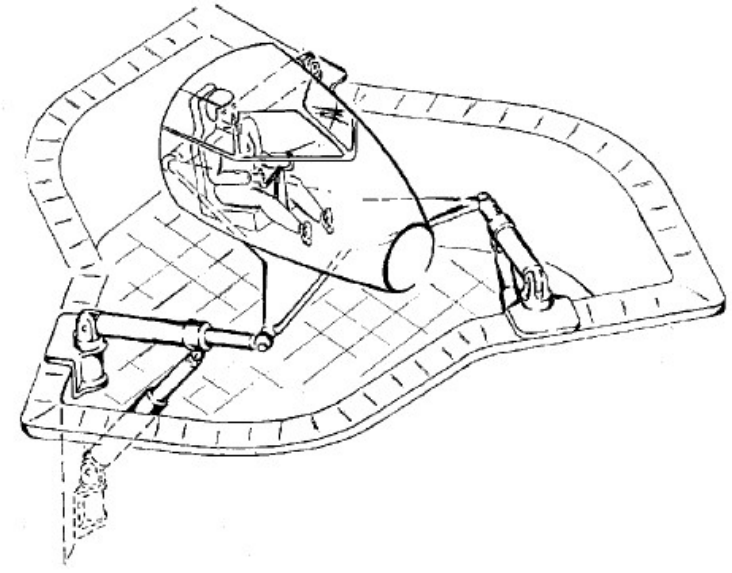
8 pav. Pramogų salė valdoma lygiagrečiuoju manipuliatoriumi [23]

Lėktuvo padangų tyrimams 1947 metais, Eric‘as Gough‘as sukonstravo pirmąją šešių laisvės laipsnių LM architektūros platformą su UPS kinematine grandimi. Tai buvo revoliucinis atradimas robotikos industrijoje. Šiuo robotu buvo naudojamasi iki pat 2000 metų [23].



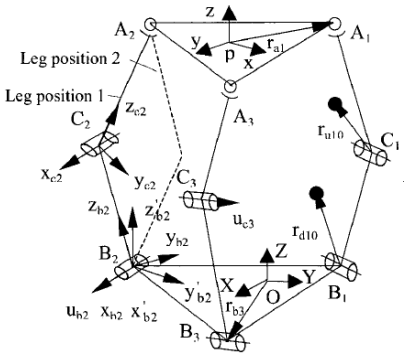
9 pav. Lėktuvo padangų bandymo stendas [23]

Jau 1962 m. Klaus‘as Cappel‘is buvo pasiūlęs panašią platformą į jau minėtą padangų bandymų stendą, skirtą imituoti judesius ir 1967 m. jis šį patentą užtvirtino. Tačiau jau 1965m. buvo publikuotas populiarusis Stiuarto straipsnis, kuriame teigiama, jog šešių laisvės laipsnių simuliacijos robotas, pagrįstas tokiu pačiu padangų bandymų stendo architektūra, gali būti panaudotas treniruojant lėktuvų pilotus. Dešimtajame paveikslėlyje pateikta Stiuarto simuliatoriaus idėjos schema [23].



10 pav. Stiuarto lėktuvo simuliatorius [23]

1.4. Trijų RRS kinematinių grandžių lygiagretaus manipuliatoriaus kinematikos lygtys



11 pav. Trijų RRS kinematinių grandžių LM platformos schema [11]

Šio tipo platformai yra šeši padėties ir orientacijos parametrai ir trys iš jų yra nepriklausomi. Kaip ir trijų RPS kinematinių grandžių LM, kiekvienas platformos sferinės jungties centras visada juda plokštuma. $P-xyz$ plokštuma yra visada fiksuota su judančia platforma, tuo tarpu P taškas yra judančiosios platformos centras ir vektorius r_{ai}, tai judančios platformos centrinio taško pozicijos vektorius. O tuo tarpu r_{bi} yra pagrindo centrinio taško pozicijos vektorius $O-XYZ$ plokštumoje. Plokštumos B_i–$\mathrm{x'}_{bi}\mathrm{y'}_{bi}\mathrm{z'}_{bi}$ (i=1,2,3) tai lokalios standžios šarnyrinės jungties koordinačių plokštumos ir identiškos su u_{bi} ašimis. Koordinačių ašys B_i–$\mathrm{x'}_{bi}\mathrm{y'}_{bi}\mathrm{z'}_{bi}$ (i=1,2,3) juda kartu su L_{BCi} vektoriumi, o x_{bi} ir z_{bi} ašys yra identiškos u_{bi} ašims. Toliau plokštumos C_i–$\mathrm{x}_{ci}\mathrm{y}_{ci}\mathrm{z}_{ci}$ (i=1,2,3) lokalios standžios šarnyrinės jungties koordinačių plokštumos juda kartu su L_{CAi} vektoriumi ir identiškos su u_{ci} bei z_{ci} ašimis. Plokštuma C_i–$\mathrm{x}_{ci}\mathrm{y}_{ci}\mathrm{z}_{ci}$ taip pat lygiagreti plokštumai A_i–$\mathrm{x}_{ai}\mathrm{y}_{ai}\mathrm{z}_{ai}$ [11].

L_{BAi} vektorius gaunamas papildomai įvedant $P-xyz$ koordinačių plokštumos orientacijos matricą R_{op} ir P taško pozicijos vektorių r_p [11].

$$L_{BAi} = r_p + R_{Op}r_{Ai} - r_{Ai} \qquad \text{(i=1,2,3)} \qquad (1)$$

Kampas α_i, tai kampas tarp vektorių L_{BAi} (gauto iš (1) formulės) ir L_{BCi} [11].

$$\alpha_i = \arccos\left(\frac{L^2{}_{BCi}+|L_{BAi}|^2-L^2{}_{CAi}}{2L_{BCi}|L_{BAi}|}\right) \qquad \text{(i=1,2,3)} \qquad (2)$$

Kampas β_i, tai kampas tarp vektorių L_{BAi} ir koordinačių ašies y'_{Bi} [11].

$$\beta_i = arccos\left(\frac{L_{BAi}*y'_{Bi}}{L_{BAi}}\right) \qquad \text{(i=1,2,3)} \qquad (3)$$

Toliau atitinkamai pagal (2), (3) formules galime gauti kampą q_{bi} tarp y_{bi} ir y'_{bi} ašių [11].

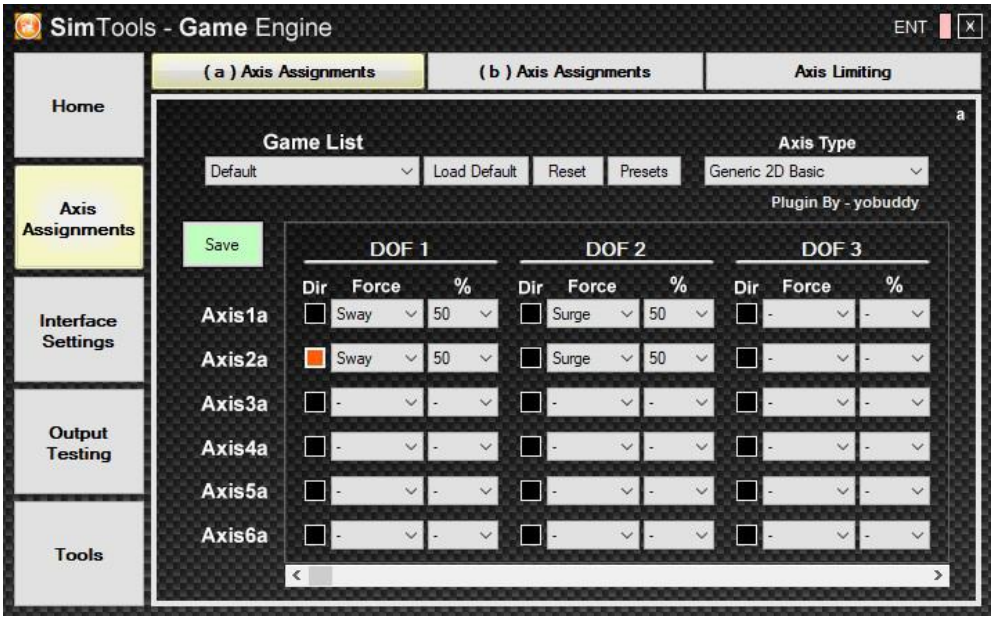
$$q_{bi} = \alpha_i + \beta_i - \frac{\boldsymbol{\pi}}{2} \qquad \text{(i=1,2,3)} \qquad (4)$$

Galiausiai išreiškiama B_i ašies lokalios koordinačių plokštumos orientacijos matrica R_{ubi} O–XYZ plokštumos atžvilgiu [11].

$$R_{ubi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q_{bi}) & -\sin(q_{bi}) \\ 0 & \sin(q_{bi}) & \cos(q_{bi}) \end{bmatrix} \qquad \text{(i=1,2,3)} \qquad (5)$$

1.5. Simuliacijos modelio judesio perteikimo, atvirojo kodo programa *SimTools*

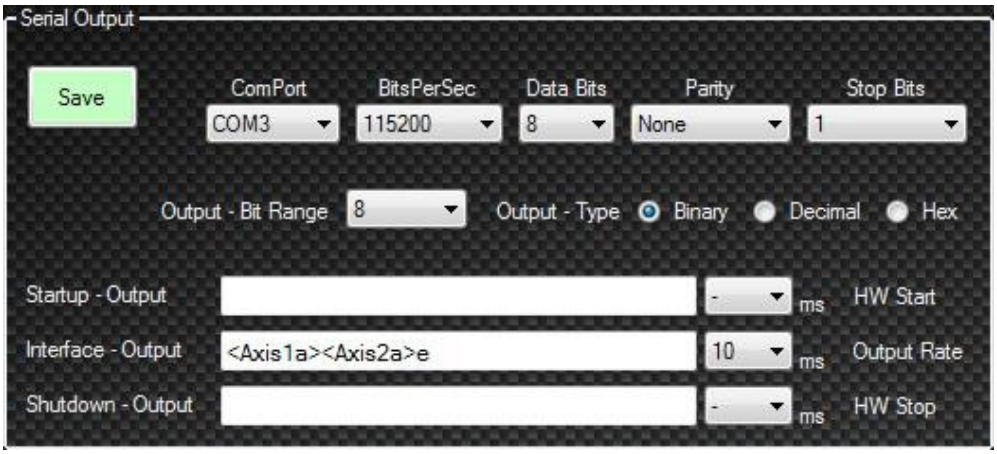
SimTools programa buvo sukurta, jog būtų prieinamas ir paprastai suprantamas įrankis, kuris padėtų žmonėms iš viso pasaulio vystyti įvairius judesių platformų simuliatorius kaip įmanoma greičiau. Šis įrankis leidžia sukonfigūruoti bei lanksčiai redaguoti beveik visas įmanomas judesio platformas nuo vieno iki šešių laisvės laipsnių bei galima pasirinkti iki trijų papildomų judesio ašių. Ši programa yra tarsi tarpininkas tarp simuliuojamos erdvės modelio realių parametrų. Judėjimo greičio, judėjimo krypties, akceleracijos dydžio (ir daugelis kitų parametrų) standartizuotų verčių ištraukimo, kurias galime perduoti judesio platformai, ir komunikacijos duomenų perdavimo.



12 pav. *SimTools* judesio platformos ašių nustatymų langas [25]

Šia programa galime naudotis nemokamai demonstraciniame režime žaidžiant tik vienintelį lenktynių simuliatorių *Live for Speed*, tačiau yra galimybė įsigyti neribotą licenciją su kuria galime bandyti simuliuoti bet kurioje kitoje simuliacinėje erdvėje.

Vienas iš populiariausių metodų perduoti simuliacinės erdvės duomenis yra naudojant serijinę komunikaciją. Populiariausia vien todėl, jog pats patogiausias mikroprocesorius Arduino ir turi *UART* serijinę komunikaciją (žr. **13 pav.**).



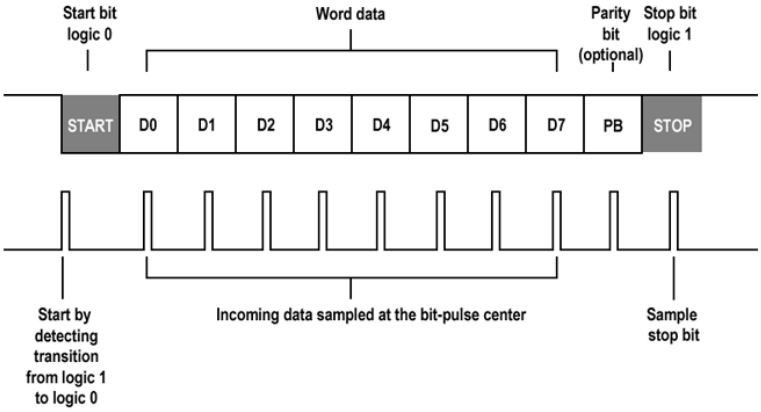
13 pav. *SimTools* serijinės komunikacijos nustatymai [25]

1.6. *UART* komunikacijos perdavimas

UART (angl. *Universal Asynchronous Receiver – Transmitter*) komunikacija buvo naudojama dar 1960 metais. Prieš *USB* atsiradimą, kompiuteriai turėjo serijinės komunikacijos prievadus, tačiau dabar šia komunikacija pagrinde naudojasi mikrovaldiklio tipo įrenginiai. Šia komunikacija duomenys siunčiami abipusiškai dvejais laidais – vienas laidas perdavimui, o kitas laidas duomenų nuskaitymui. Duomenys perduodami po vieną dvejetainį bitą, toliau šie bitai sugrupuojami į reikšmingą grupės formatą vadinamu *kadru* [24].

UART komunikaciją galime laikyti universalia, kadangi tokie parametrai kaip greitis, duomenų dydis ir kiti parametrai nėra fiksuoti ir gali būti konfigūruojami. Tačiau reikia atkreipti dėmesį, jog siunčiamoje pusėje ir gavėjo pusėje turi būti tokie patys parametrai, kadangi duomenų apsikeitimas vyksta asinchroniniu būdu [24].

Kadangi nėra duomenų sinchronizavimo, naudojami *Start* ir *Stop* bitai, kurie atskiria siunčiamus duomenų kadrus, bet vėlgi reikia nustatyti duomenų perdavimo greitį siuntėjo ir gavėjo pusėje. Pariteto bitas siunčiamas tik tuo atveju, jeigu tikslinga tikrinti ar perduodami duomenis yra teisingi ir tik tai su 8 bitų duomenimis [24].



14 pav. *UART* komunikacijos kadro sandara [24]

Serijinėje komunikacijoje duomenys koduojami *ASCII* formatu, kadangi visiems elementams užkoduoti dvejetaine sistema užtenka 7 bitų (žr. **15 pav.)** [24]. Tačiau šiais laikais *ASCII* dažniausiai koduojamas tiesiai į baitą (8 bitų kadrą).

USASCII code chart

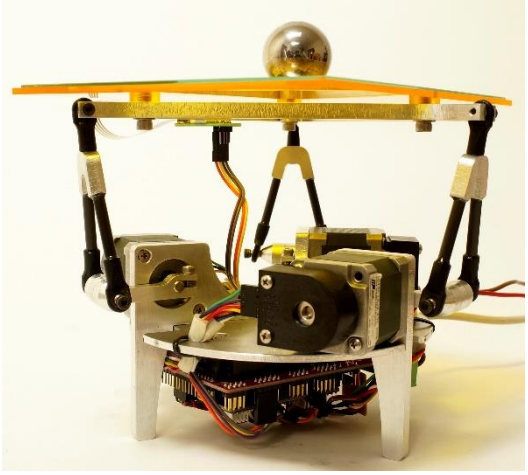
Bits b7 b6 b5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4 b3 b2 b1 / Row ↓ Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

15 pav. *ASCII* simbolių lentelė [24]

2. Projektinė dalis

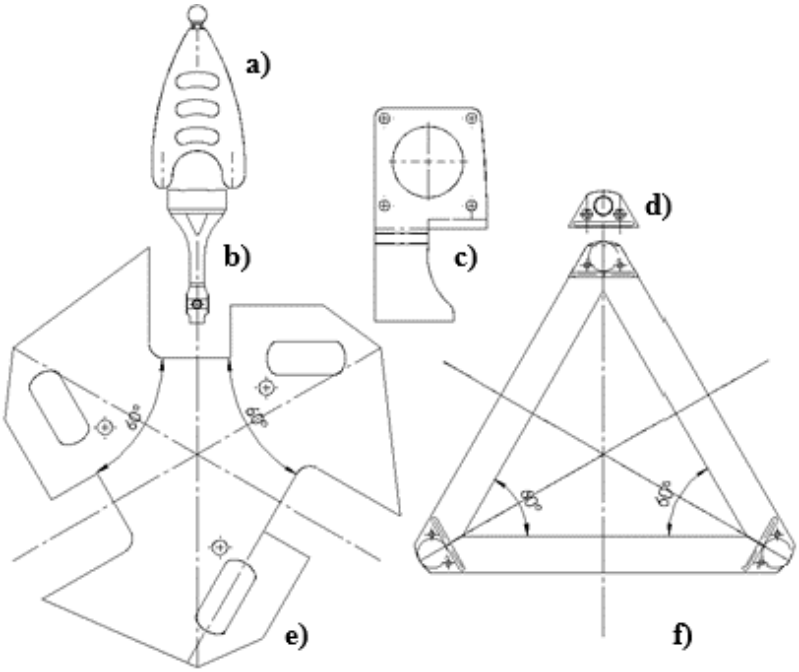
2.1. Prototipo projektavimas

Baigiamajame projekte trijų RRS kinematinių grandžių platforma suprojektuota remiantis pirmuoju literatūros šaltiniu, kuriame sukurtas LM naudojamas kamuolio pozicionavimui. Šios architektūros mechanizmas leidžia platformai judėti trijų laisvės laipsnių ašimis, kurios yra būdingos RRS tipo kinematinės grandies LM judesio simuliatoriams.



16 pav. Kamuolio pozicionavimo sistema [1]

Suprojektuotos šešios detalės pritaikytos pagal užsakytus judesio platformos komponentus , kurie buvo pilnai atspausdinti su *3D* spausdintuvu su *PLA* medžiaga. Pagrindo bei variklio laikiklio detalės suprojektuotos taip, jog būtų įstumiamos viena į kitą be jokių papildomų fiksavimų. Likusiose detalėse įstatytos varžto veržlės į pritaikytas tuščias ertmes ir priveržta srieginiais *M4* varžtais.



17 pav. Prototipo LM platformos suprojektuotos detalės: a) viršutinė koja, b) apatinė koja, c) variklio laikiklis, d) viršutinės kojos fiksatorius, e) platformos pagrindas, f) platforma

Kinematinių grandžių šarnyriniams sąnariams panaudotos šarnyrinės galvutės su sriegiu, (žr. **18 pav.**) o sferinis sąnarys išpildytas naudojant viršutines kojas su sferiniu paviršiumi.

18 pav. *M4* Šarnyrinė galvutė su sriegiu [14]

Platformos judinimui parinkti *NEMA23* žingsniai varikliai su vidiniu grįžtamuoju ryšiu bei mikrožingsnių valdikliais, (žr. **19 pav.**) jog būtų užtikrintas preciziškas atvirojo kontūro valdymas be praleidžiamų žingsnių. Valdiklio pagalba taip pat galima keisti variklio mikrožingsnių skaičių vienam veleno apsisukimui nuo 800 mikrožingsnių iki 40000 mikrožingsnių, taip pasiekiant nuo 0,45 laipsnių iki 0,009 laipsnių veleno pasisukimo tikslumą vienam žingsniui.

19 pav. *RTelligent T60 NEMA23* žingsninis variklis su valdikliu [14]

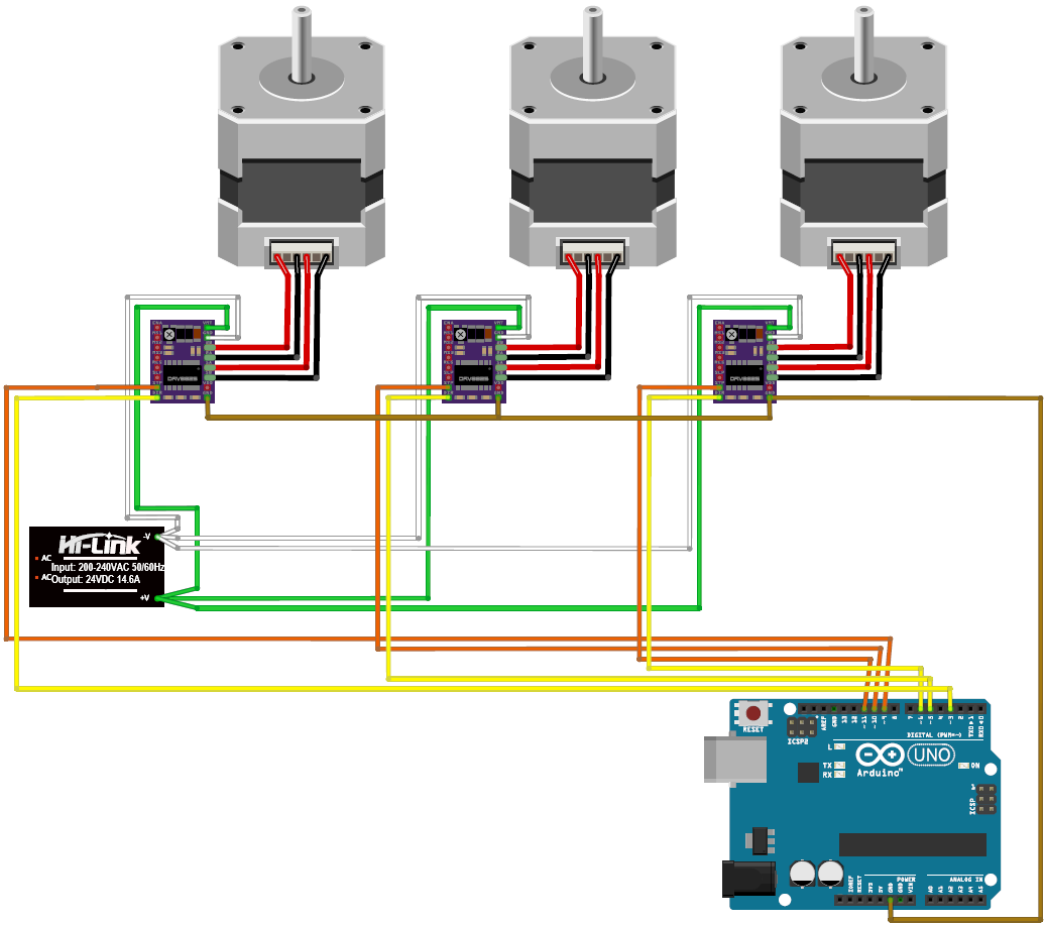
Kadangi šių žingsninių variklių nominali maitinimo įtampa yra 24V ir nominali srovė yra lygi 3,5A, reikalingas papildomas nuolatinės srovės 24V maitinimo šaltinis gebantis atlaikyti tris vienu metu lygiagrečiai veikiančius variklius. Su 30% atsargos koeficientu reikia turėti bent 13,65A srovės sugebantį suteikti 24V nuolatinės srovės šaltinį. Parinktas *MEAN WELL* 24V, 14,6A maitinimo šaltinis (žr. **20 pav.**).

20 pav. *MEAN WELL* nuolatinės srovės maitinimo šaltinis [14]

Žingsninių variklių valdymui išrinktas *Arduino UNO R3* 16MHz mikrovaldiklis su *ATmega328P* procesoriumi bei *UART* komunikacija (žr. **21 pav.**). Žingsninių variklių valdymui panaudoti šeši laisvi diskretiniai išėjimo kontaktai. Trys kontaktai variklio pozicijos valdymui ir likę sukimosi krypčiai keisti.

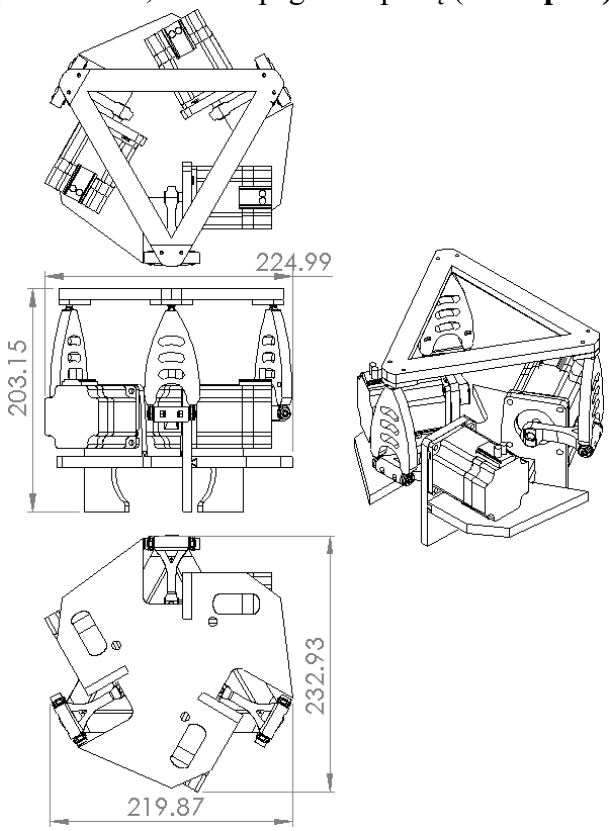
21 pav. *Arduino UNO R3* mikrovaldiklis [14]

Visa projekto prototipo elektros schema susideda iš paskutinių trijų išvardintų elektros komponentų. Kiekvieno žingsninio variklio maitinimo apvijos yra sujungiamos į žingsninių variklių valdiklius raudonais ir juodais laidais. Toliau šie valdikliai lygiagrečiai prijungiami prie maitinimo šaltinio baltais ir žaliais laidais. Valdiklių mikrožingsnių valdymo signalai yra jungiami prie mikrovaldiklio 9,10 ir 11 diskretinių kontaktų jungiami oranžiniais laidais, o krypties valdymo signalai jungiami prie 3, 5 ir 6 diskretinių kontaktų geltonais laidais. Toliau grandinėle prijungiami paskutiniai 0V *Arduino* signalai į valdiklių 0V bendrą signalų kontaktą rudu laidu (žr. **22 pav.**).



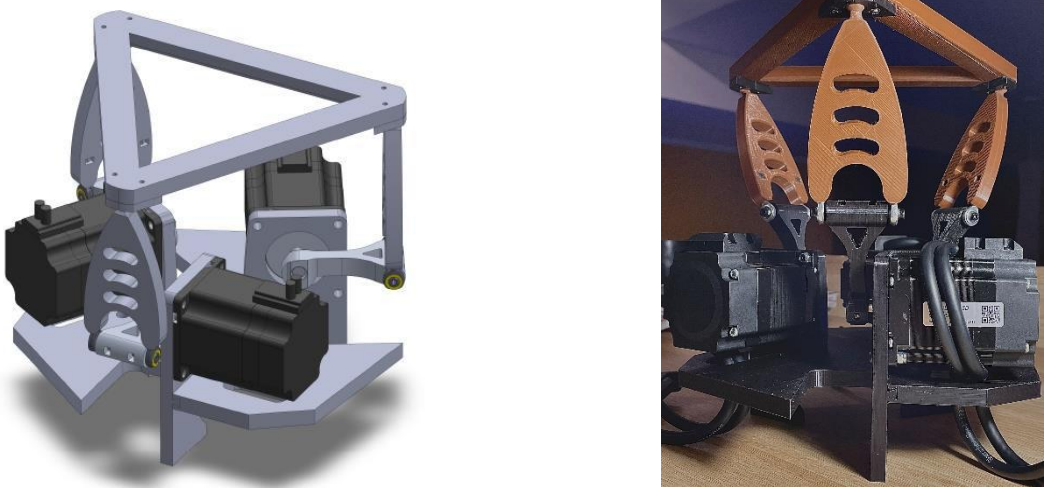
22 pav. Prototipo elektros jungimo schema

Galutinė prototipo projekto konstrukcija susideda iš keturiolikos *3D* atspausdintų detalių, šešių šarnyrinių galvučių, dvidešimt vieno varžto bei dvidešimt septynių veržlių. Prototipas telpa į 225x233x204mm matmenų erdvę, esant neutralioje vidurinėje pozicijoje. Remiantis matmenimis galime spręsti, jog prototipas užima 0,0524m^2 pagrindo plotą (žr. **23 pav.**).



23 pav. Prototipo konstrukcijos pagrindiniai matmenys

Nepaisant to, jog atspausdintos detalės buvo taisomos nuo spausdinimo netobulumų, surinkus visus suprojektuotus platformos elementus, prototipas išliko stabilus ir nei viena detalė neišsprūdo iš sąnarinių jungčių judant šiai platformai būdingomis laisvės ašimis (žr. **24 pav.**).



a) b)

24 pav. Prototipo konstrukcijos palyginimas tarp a) *SolidWorks* modelio ir b) realaus prototipo

2.2. Prototipo savikainos nustatymas

Suprojektavus judesio platformos prototipą, taikoma prielaida, jog *3D* spausdintuvu atspausdintos detalės nieko nekainavo, kadangi panaudoti KTU laboratorijoje esantys *3D* spausdintuvai. Į judesio platformos prototipo gamybos kainą įtraukti elektros įrangos komponentai, sąnarinės jungtys bei tvirtinimo elementai (žr. **1 lentelė**).

1 lentelė. Judesio platformos prototipo komponentų kainos [14]

Komponentas	**Gamintojas**	**Kiekis**	**Vieneto Kaina, €**
Mikroprocesorius	*Arduino*	1	22,00
Žingsniniai varikliai ir jų valdikliai	*Rtelligent*	3	117,53
Šarnyrinė jungtis	*Anodas*	6	1,00
Maitinimo šaltinis	*MEAN WELL*	1	37,40
Laidai	*Lietkabelis*	100	0,32
Varžtai	*Anjesė*	21	0,12
Veržlės	*Anjesė*	27	0,02
Iš viso:			**453,05**

Judesio platformos prototipo bendra kaina siekia – 453,05€. Tačiau realaus dydžio judesio platformai įgyvendinti reikalingas komponentų skaičius išauga. Kadangi reikalinga stipresnė keliamoji jėga, tampa reikalingas reduktorius, dėl keliamosios jėgos reikalavimų [19], [17], [2]. Papildomas maitinimo šaltinis tampa nebereikalingas, kadangi realiai platformai pakelti tinkamiausi sinchroniniai kintamosios srovės servo varikliai [20]. Kinematinių grandžių sąnarinės dalys sudedamos iš atskirų detalių. Tampa svarbios stabdymo varžos skirtos variklių generavimo energijos išsklaidymui, jog neperkaistų varikliai bei valdikliai. Servo variklių valdymui tampa reikalinga serijinės komunikacijos *DB25* jungtis, dėl kitokios valdiklių sąsajos. Taip pat nenumatytoms situacijoms svarbus avarinis mygtukas, skirtas variklių sustabdymui [15].

Norint nustatyti realios judesio platformos savikainą reikalinga daryti prielaidą apie pačios konstrukcijos kainą, kadangi šiame baigiamajame darbe tai nebuvo nagrinėta. Prielaida, jog konstrukcijos kaina neviršija 1000€ sumos. Pagal nustatytus naujus komponentus bei pritaikytą prielaidą, gauta nauja komponentų lentelė (žr. **2 lentelė**).

2 lentelė. Realaus dydžio judesio platformos komponentų kainos [14],[15]

Komponentas	**Gamintojas**	**Kiekis**	**Vieneto Kaina, €**
Mikroprocesorius	*Arduino*	1	22,00
Žingsninis variklis ir valdiklis	*Leadshine*	3	720,58
Šarnyrinė jungtis	*SKF*	6	28,75
Sferinė jungtis	*Igus*	3	39,02
Reduktorius	*Elektros Prekės*	3	120,00
Stabdymo varža	*REOHM*	3	67,00
Laidai	*Lietkabelis*	100	0,32
DB9 komunikacijos kabelis	*Penglin*	3	16,32
Avarinis mygtukas	*Gloglow*	1	24,62
Platformos konstrukcija	–	1	1000,00
Iš viso:			**4139,88**

Pagal antrosios lentelės duomenis gauta, jog realaus dydžio judesio platformos kainos suma 9,14 kartų didesnė nei prototipo platformos – 4139,88€. Tačiau ši suma vis vien yra žemesnė palyginus su kitomis rinkoje esančiomis judesio platformomis [2], [19].

2.3. Mikrovaldiklio programos valdymo algoritmas

Pagal projekto pirmojo priedo valdymo algoritmo schemą, mikrovaldiklio *Arduino C++* programavimo kalbos pagrindo programos aplinkoje galime atskirti tris ciklo etapus:

1. konstantų, kintamųjų bei funkcijų pradinių verčių deklaravimas;
2. žingsninių variklių valdymas;
3. serijinės komunikacijos duomenų apsikeitimas.

Pirmajame etape nustatomas ašių skaičius **kAxisCount**, komunikacijos ašies duomenų atskiriamasis simbolis pavadinimu **kEOL**. Nustatomi ašies simboliai kiekvienai atskirai ašiai duomenų atpažinimui su konstantų masyvu **kAxisName**. Kintamasis **kMaxCharCount** nurodo koks bus maksimalus naudojamas skaitmenų skaičius, nusakantis ašies poziciją. Šiuo atveju nuo 0 iki 255, nes naudojama 8 bitų rezoliucija. Konstantų masyvas **kAxisScale** keičia prototipo fizinius judėjimo apribojimų mastelius. Toliau nurodomi, kurie *Arduino* signalų kontaktai naudojami žingsninių variklių valdymui. Masyvuose **STEP** ir **DIR** nurodomi, kokie kontaktai naudojami žingsniavimo signalams ir kokie naudojami sukimosi krypčiai pakeisti. Pozicijos sekimo masyvuose **Pos** ir **AxisPosition** inicializuojamos nulinės vertės, taikant tokią prielaida, jog prototipas prieš įjungiant visada yra apatinėje, nusileidusioje padėtyje. Kintamasis **valueCharCount**, inicializuojamas su nuline verte, atlieka komunikacijos duomenų baitų nuskaitymo kiekio skaičiuoklio funkciją. Įprastam **digitalWrite** funkcijos impulso trukmės nustatymui naudojamas kintamasis **pulseWidth**. Deklaruojamas kintamasis **currentAxis**, kuris skirtas nuskaitytų ašių sekimui iš *SimTools* programos. Taip pat panaudotas enumeratoriaus kintamasis **TPortState**, kuris turi būsenos reikšmę **psReadxis**, kuri reiškia, jog nuskaitomas komunikacijos ašies pavadinimas ir reikšmė **psReadValue**, kad nuskaitomi ašies pozicijos duomenys. Toliau Inicializuojama ašies nuskaitymo vertė [4].

```
#define MODE1                   // Nustatomas veikimo režimas
//#define MODE2
//#define MODE3
#include <digitalWriteFast.h>  // įtraukiama biblioteka skirta antrajam režimui
#define pinNum1 9              // nustatomi kontaktai antrajam režimui
#define pinNum2 10
#define pinNum3 11
#define pinDir1 3
#define pinDir2 5
#define pinDir3 6
const int   kAxisCount                = 3;        //nustatomas ašių kiekis
const char  kEOL                      = '~';      //duomenų atskyrimo simbolis
const char  kAxisName[kAxisCount]     = {'A','B','C'}; //SimTools ašių simboliai
//nustatomi prototipo fizinio judesio ribų masteliai
const int   kAxisScale[kAxisCount][3] = { {0,13000},{0,13000},{0,13000}};
const int   kMaxCharCount             = 3;   //ašies pozicijos simbolių kiekis
//nustatomi kontaktai pirmajam režimui
const int   STEP[]                    = {9,10,11};//žingsniavimo kontaktai
const int   DIR[]                     = {3,5,6};  //sukimosi krypties kontaktai
//inicializuojamos nulinės reikšmės pozicijos sekimo masyvams
int         Pos[]                     = {0,0,0};  //prototipo pradinės reikšmės
int         AxisPosition[kAxisCount]  = {0,0,0};  //SimTools pradinės reikšmės
//inicializuojama nuskaitytų SimTools komunikacijos duomenų pradinė reikšmė
int         valueCharCount            = 0;
int         pulseWidth                = 10;       //digitalWrite impulso plotis
```

```
int         currentAxis;
enum TPortState { psReadAxis, psReadValue }; //komunikacijos apsikeitimo būsena
TPortState currentState = psReadAxis;        //įrašoma ašies pavadinimo
nuskaitymo būsena
```

Galiausiai deklaruojamos funkcijos, kurios inicializuoja komunikacijos duomenų bitų apsikeitimų spartą **Serial.begin** ir funkcijos priskiriančios *Arduino* kontaktų operacijos veikimo principą **pinMode** [4] bei **pinModeFast**. Visų *Arduino* kontaktai naudojami kaip išvesties signalai, kadangi grįžtamasis ryšys naudojamas tik mikrožingsnių valdikliams. Projekto programoje naudojami trys režimai, kuriuose programa gali veikti. Pirmajame režime programa veikia įprastu metodu naudojant **digitalWrite** impulso generavimo funkciją [4]. Antrajame režime naudojama išorinė **digitalWriteFast.h** biblioteka, kuri yra sukurta paprastam tiesioginiam prievadų valdymui, kuris yra žymiai greitesnis [5],[12]. O trečiasis režimas skirtas įvertinti maksimalų žingsninių variklių sukimosi greitį, kai sukasi nuo vieno iki trijų variklių vienu metu.

```
void setup()       //ši funkcija iškviečiama po kiekvieno įjungimo vieną kartą
{     Serial.begin(500000); //nustatomas komunikacijos bitų apsikeitimo greitis
    #ifdef MODE2                    //apibrėžiamos antrojo režimo operacijos
      pinModeFast(pinNum1,OUTPUT);//nustatomas kontakto išėjimo signalo režimas
      pinModeFast(pinDir1,OUTPUT);
      pinModeFast(pinNum2,OUTPUT);
      pinModeFast(pinDir2,OUTPUT);
      pinModeFast(pinNum3,OUTPUT);
      pinModeFast(pinDir3,OUTPUT);
    #endif
    #ifdef MODE1                    //apibrėžiamos pirmojo režimo operacijos
    #ifdef MODE3                    //apibrėžiamos trečiojo režimo operacijos
      for (int i=0; i < kAxisCount; i++) //ciklas kartojamas tiek, kiek variklių
      {
        pinMode(STEP[i],  OUTPUT);//nustatomas kontakto išėjimo signalo režimas
        pinMode(DIR[i],   OUTPUT);
      }
    #endif
    #endif
}
```

Antrajame etape sukasi ciklas tiek, kiek yra ašių ir tikrinama sąlyga ar skaičiuojamos ašies skirtumas tarp fizinio prototipo pozicijos ir simuliacijos modelio esančios pozicijos. Simuliacijos pozicija siunčiama iš *SimTools* per serijinę komunikaciją ir tikrinama ar ji nelygi nuliui su **Calc** funkcija. Toliau, jeigu sąlyga yra teisinga, tikrinama ar fizinio prototipo pozicijos reikšmė yra didesnė už simuliacijos modelio pozicijos reikšmę. Kai sąlyga teisinga, su **digitalWrite** funkcija nustatoma priešinga ašies sukimosi kryptis. Tada iš fizinio prototipo pozicijos sekimo masyvo **Pos** atimama vieneto reikšmė (prilygstanti vienam žingsnio impulsui) ir atliekamas vienas impulsas, kuris yra siunčiamas žingsninių variklio valdikliui pajudėti per vieną žingsnį. Kai sąlyga neteisinga, su **digitalWrite** funkcija nustatoma įprasta ašies sukimosi kryptis. Prie fizinio prototipo pozicijos sekimo masyvo **Pos** pridedama vieneto reikšmė ir atliekamas toks pats signalas žingsninių variklio valdikliui pajudėti per vieną žingsnį. Atlikus šį ciklą toliau keliaujama prie trečiojo etapo [4]. Antrajame režime programa veikia taip pat, tik vietoj įprastų metodų naudojamas greitesnis **digitalWriteFast** režimas.

```
  #ifdef MODE1 //apibrėžiamos pirmojo režimo operacijos
  for (int i = 0; i < kAxisCount; i++)  //ciklas kartojamas tiek, kiek variklių
  {//tikrinamas skirtumas tarp prototipo ir Simtools pozicijos matricos reikšmės
    if (Calc(axisPosition[i],Pos[i]) !=0),
    {//jeigu skirtumas nelygus nuliui, tikrinamos pozicijos reikšmės
      if (Pos[i] > axisPosition[i])
       //jeigu prototipo pozicija didesnė, tada nustatoma taip:
      {digitalWrite(DIR[i], LOW);       //priešinga sukimosi kryptis
       Pos[i] = Pos[i] - 1;} //iš prototipo pozicijos atimama vieneto reikšmė.
      else //jeigu prototipo pozicija mažesnė, tada nustatoma taip:
```

```
      {digitalWrite(DIR[i], HIGH);//pagrindinė sukimosi kryptis
       Pos[i] = Pos[i] + 1;}//prie prototipo pozicijos pridedama vieneto reikšmė
      monoPulse(STEP[i]);   //atliekame vieną impulso periodą pirmuoju metodu
  } }
  #enddef
```

Patys signalų impulsai žingsniniams varikliams atliekami naudojant **monoPulse** operaciją. Šiai operacijai naudojamas skaitmeninio išėjimo signalo maksimalios ir minimalios vertės su kintamojo **pulseWidth** reikšmės ilgio pauzėmis. Tokiu būdu sukuriamas paprastas impulsų generavimas. Šio principo impulsus galime lengvai suskaičiuoti (tai yra reikalinga, kadangi neegzistuoja grįžtamojo ryšio signalas, ateinantis į *Arduino* mikrovaldiklį) [4].

```
  #ifdef MODE1                    //apibrėžiamos pirmojo režimo operacijos
  #ifdef MODE3                    //apibrėžiamos trečiojo režimo operacijos
void monoPulse(int stepPin)       //funkcija skirta digitalWrite žingsniavimui
{
  digitalWrite(stepPin, HIGH);    //nustatoma žingsniavimo kontakto aukšta vertė
  delayMicroseconds(pulseWidth); //nustatyto impulso aukštos vertės užlaikymas
  digitalWrite(stepPin, LOW);     //nustatoma žingsniavimo kontakto žema vertė
  delayMicroseconds(pulseWidth); //nustatyto impulso žemos vertės užlaikymas
}
  #enddef
  #enddef
```

Pasirinkus antrojo režimo impulso generavimą naudojamas žymiai greitesnis tiesioginis prievadų junginėjimo metodas. Impulso periodą reikalinga pakartoti net 10 kartų, kol galiausiai žingsninio variklio valdiklis gali aptikti signalą kaip tinkamą signalą. **DelayMicroseconds** funkcija čia jau nebenaudojama, kadangi šios operacijos įvyksta greičiau nei mažiausia galima funkcijos vertė.

```
  #ifdef MODE2                   //apibrėžiamos antrojo režimo operacijos
void FastPulse(int stepPin)      //funkcija skirta fastDigitalWrite žingsniavimui
{
  digitalWriteFast(stepPin, HIGH);//nustatoma žingsniavimo kontakto aukšta vertė
  digitalWriteFast(stepPin, LOW); //nustatoma žingsniavimo kontakto žema vertė
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
}
  #enddef
```

Trečiajame etape vyksta duomenų apsikeitimas tarp *Arduino* mikrovaldiklio bei *SimTools* programinės įrangos. Ši programos dalis pradeda veikti tik tada, kada veikia funkcija **serialEvent**. Ji tikrina ar nauji duomenys atkeliauja per *Arduino* komunikaciją ir tik tada atlieka nurodytus veiksmus. Toliau tikrinama kita sąlyga **Serial.available**, kuri parodo ar duomenys yra jau atkeliavę ir yra sukauptas baitas. Kai šios komunikacijos sąlygos išpildomos, prasideda duomenų skaitymas. Jeigu enumeratoriaus **currentState** būsena yra lygi **psReadAxis** būsenai, pradedama nuskaitinėti

siunčiamų duomenų ašies pavadinimą. Duomenų baito nuskaitymui naudojama funkcija **Serial.read** ir įrašoma į simbolio tipo kintamąjį **tmpChar**. Toliau ieškoma su kokiu ašies pavadinimu yra lygus nuskaityto duomens baito simbolio atitikmuo. Sukant šios operacijos ciklą, jeigu randamas atitikmuo, įsimenama ašis, inicializuojama enumeratoriaus reikšmė **psReadValue.** Ji skirta ašies duomenų nuskaitymui bei priskiriamos nulinės reikšmės nuskaitomai ašiai bei nuskaitomos ašies pozicijos duomenų simbolių kiekio skaičiuokliui. O tada išeinama iš ciklo ir keliaujama toliau prie ašies duomenų nuskaitymo [4].

```
void serialEvent() //ši funkcija iškviečiama, kai komunikacija perduoda duomenis
    char  tmpChar; //deklaruojamas kintamasis skirtas ašies simbolių duomenims
    int   tmpValue;//deklaruojamas kintamasis skirtas ašies pozicijos duomenims
    while (Serial.available()) //kai komunikacijoje yra sukauptas duomenų baitas
    { //tikrinama ar dabartinė būsena ašies pavadinimo nuskaitymo būsena
      if (currentState == psReadAxis)
      {
        tmpChar = Serial.read();  //nuskaitomi ašies pavadinimo duomenys
         //ciklas kartojamas tiek, kiek variklių arba kol randamas atitikmuo
        for (int i = 0; i < kAxisCount; i++)
        { // jeigu nuskaitomas simbolis lygus ašies pavadinimo simboliui
          if (tmpChar == kAxisName[i])
          {
            currentAxis = i; //sutapusi ašis įsimenama
            //įrašoma ašies pozicijos duomenų nuskaitymo būsena
            currentState = psReadValue;
            //inicializuojama pradinė nulinė reikšmė komunikacijos nuskaitymui
            axisPosition[currentAxis] = 0;
            //inicializuojama pradinė nulinė reikšmė nuskaitytų simbolių skaičių
            valueCharCount = 0;
            break;                //nutraukiamas ciklas
          }
        }
       }
    }
```

Ašies duomenų nuskaitymo proceso metu tikrinama sąlyga ar duomenų simbolio kiekis **valueCharCount** yra mažesnis, negu maksimalus simbolių kiekis **kMaxCharCount**. Kai sąlyga teisinga, nuskaitomas ašies duomenų sukauptas baitas ir priskiriamas prie sveikojo skaičiaus tipo kintamojo **tmpValue** su **Serial.read** komandos pagalba. Jeigu nuskaitytas simbolis nėra lygus simboliui skirtam ašies duomenų atskyrimui **kEOL**, tai iš nuskaitytos sveikojo skaičiaus reikšmės atimamas skaičius 48. Atimčiai naudojamas toks skaičius, kadangi būtent nuo šio skaičiaus *ASCII* formatu koduojami skaičiai naudojami dešimtainėje sistemoje (t.y. pavieniai skaičiai nuo 0 iki 9). Toliau tikrinama sąlyga – ar gautieji skaičiai yra mažesni už 0 ir arba didesni už 9. Jeigu sąlyga patenkinama, kintamąjį **tmpValue** prilyginame nuliui, kadangi gautieji duomenys nėra dešimtainėje sistemoje ir jog neatsirastų klaidinančių duomenų [4].

```
//tikrinama ar dabartinė būsena ašies pozicijos duomenų nuskaitymo būsena
if (currentState == psReadValue)
   {  //vykdomas ciklas kol nuskaitytų duomenų baitų kiekis pasiekią maksimalų
   while ((valueCharCount < kMaxCharCount))
   {
   tmpValue = Serial.read();    //nuskaitomi ašies pozicijos duomenys
     //tikrinama ar nuskaitomas duomenų baitas yra atskiriamasis simbolis
   if (tmpValue != kEOL)
   {                            //konvertuojama iš ASCII į dešimtainę sistemą
            tmpValue = tmpValue - 48;
      //tikrinama ar konvertuota reikšmė yra tarp 0 ir 9 skaičių reikšmių
            if ((tmpValue < 0) || (tmpValue > 9))
            { tmpValue = 0; }  //jeigu neatitinka tada prilyginama nuliui
```

Jeigu vis dėlto nuskaitomas duomenų baitas yra dešimtainėje sistemoje, tada esamai ašiai įrašoma simuliacijos modelio pozicijos pirmas skaitmuo iš trijų į kintamąjį **axisPosition**. Tada pridedama vieneto reikšmė prie **valueCharCount** kiekio skaičiuoklio. Toks ciklas tęsiasi tol, kol pasiekiama trečia skaitmens reikšmė ir tada nutraukiamas. Paskutinėje tikrinimo sąlygoje stebima ar nuskaitomo duomenų baito *ASCII* reikšmė yra lygi ašies duomenų atskyrimo simboliui **kEOL**. Kita stebima sąlyga ar K**valueCharCount** pasiekė maksimalų simbolių kiekį **kMaxCharCount**. Jeigu bent vieną sąlygą atitinka, tada gautą simuliacijos modelio pozicijos reikšmės mastelį keičiame su **map** komanda. Ši komanda skirta prototipo fizinių judėjimo apribojimui. Toliau priskiriame enumeratoriui **currentState** reikšmę **psReadAxis**, jog sekančio ciklo metu vėl būtų pirmiausia nuskaitomas ašies pavadinimas. Pasibaigus trečiajam etapui *Arduino* programos ciklas vėl pradedamas nuo antrojo etapo pradžios [4].

```
//nuskaityta baito skaitinė reikšmė įrašoma į pozicijos masyvą
axisPosition[currentAxis] = axisPosition[currentAxis] * 10 + tmpValue;
valueCharCount++;//skaičiuojama kiek įrašyta skaitmenų
          }
          else break;
        }   //tikrinama ar nuskaitytas simbolis yra atskiriamasis simbolis
            //arba pasiektas maksimalus trijų simbolių skaičius
        if (tmpValue == kEOL || valueCharCount == kMaxCharCount)
        {   //ašies pozicijos duomenų mastelis keičiamas į fizinio prototipo
            //judesių ribų mastelį
axisPosition[currentAxis] = map(axisPosition[currentAxis], 0, 255,
kAxisScale[currentAxis][0], kAxisScale[currentAxis][1]);
currentState = psReadAxis; //inicializuojama ašies pavadinimo nuskaitymo būsena
        }
```

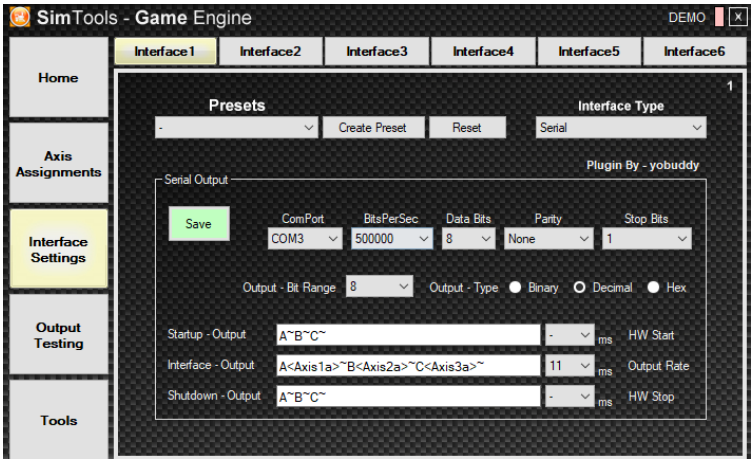
Pilna programos valdymo algoritmo schema bei pilnas programos kodo išdėstymas *Arduino* programavimo aplinkoje yra pateiktas pirmajame ir antrajame priede.

2.4. Simuliacijos modelio duomenų perdavimo programos *SimTools* nustatymai

Kadangi prototipo platforma gali judėti tik trimis laisvės laipsnių ašimis *Roll*, *Pitch* ir *Heave*, pirmiausiai reikia šias ašis sukonfigūruoti taip, jog *Heave* laisvės ašimi visi vykdikliai judėtų ta pačia kryptimi. *Roll* ir *Pitch* laisvės laipsnių ašys turi du vykdiklius, kurie juda viena kryptimi, o likęs vykdiklis priešinga kryptimi. Atitinkamai kiekvienam vykdikliui priskiriame 33% ašies pasiskirstymą, jog veikiant visoms ašims vienu metu sumoje būtų 100% ašių pasiskirstymas (žr. **25 pav.**).

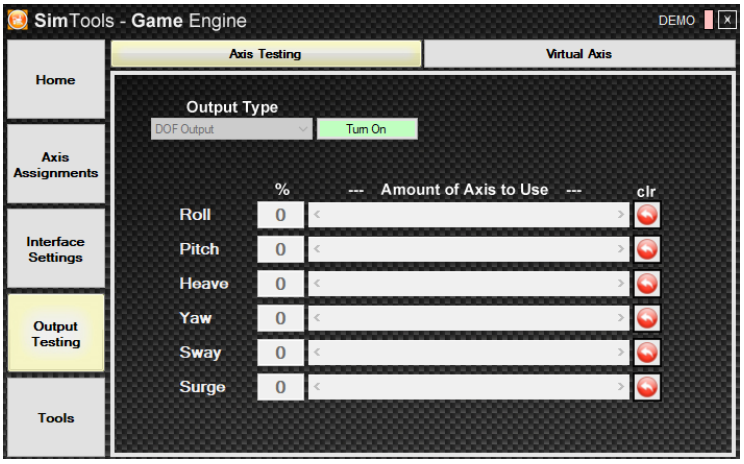
25 pav. Prototipo ašių konfigūracijos langas *SimTools* programoje

Eilutės pavadinimu **Axis1a** žymimam **A** simboliu komunikacijos perdavime priskiriami invertuotos *Roll*, *Pitch* ir *Heave* laisvės ašys. Pagal visų šių laisvės ašių įvestus parametrus nustatoma standartizuota reikšmė, kuri yra perduodama per komunikacija būtent **A** simboliu pažymėtam lokalios prototipo ašies vykdiklio valdymui. Tokiu pat principu perduodama ir **Axis2a** bei **Axis3a** duomenų informacija (žr. **26 pav.**).



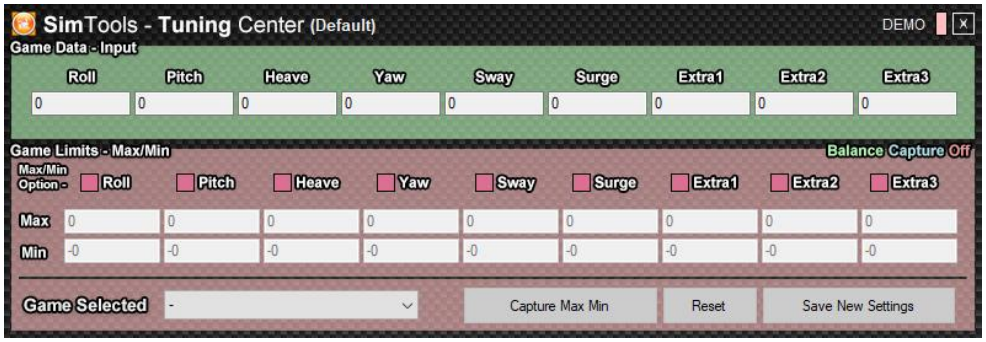
26 pav. Komunikacijos konfigūracijos langas *SimTools* programoje

Prototipo platformos judinimui galima naudoti laisvės laipsnių ašių tiesioginio manipuliavimo *SimTools Game Engine* sąsaja (žr. **27 pav.**) arba naudojant *SimTools Game Manager* realaus laiko simuliacinės erdvės modelio sąsaja (žr. **28 pav.**).



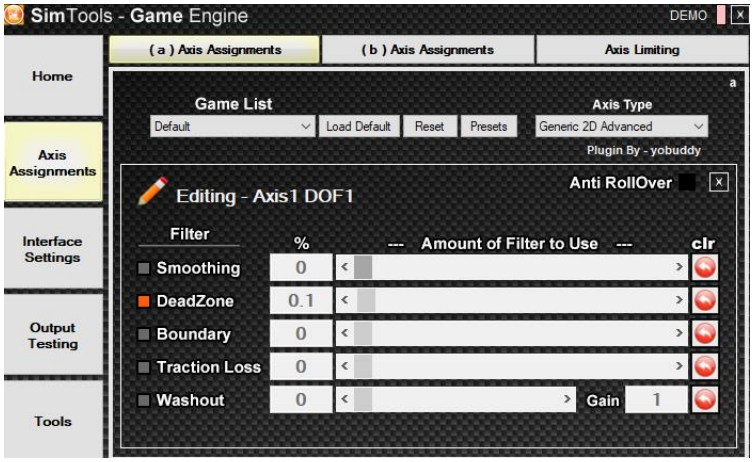
27 pav. Prototipo tiesioginių ašių manipuliavimo su *SimTools Game Engine sąsaja*

Kai realiu laiku testuojama naudojant simuliacinį modelį, galima matyti kintančias laisvės laipsnių skaitines reikšmes ir atitinkamai nustatyti, kokias ribines reikšmes gali būti siunčiamos tolimesniam duomenų apdorojimui ir siuntimui į *Arduino* mikrovaldiklį.



28 pav. Simuliacijos laisvės laipsnių ašių konfigūravimas su *SimTools Game Manager sąsaja*

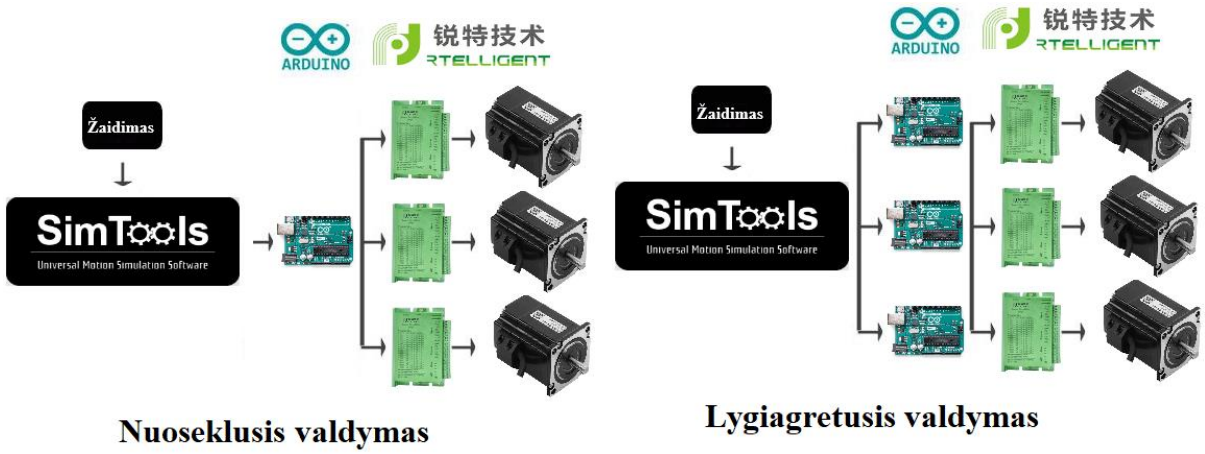
Jeigu platformos judėjimas vis tiek dar nėra tinkamas ir manevruoja per šiurkščiai, galima naudoti papildomas judesio filtravimo parametrus kaip *DeadZone*, kuris didina platformos judesio akląją zoną, arba *Smoothing*, kuris švelnina platformos judesį (žr. **29 pav.**).



29 pav. Simuliacijos modelio judesio filtravimo nustatymai

Apibendrinant, visas prototipo judėjimo procesas prasideda žaidimo simuliuojamo modelio parametrų generavimu, kurie yra perduodami *SimTools* programinei įrangai. Gauti duomenys standartizuojami programos pagalba pagal nustatytus konfigūravimo parametrus ir toliau perduodami *UART* serijine komunikacija į *Arduino* mikrovaldiklį. Per komunikaciją gauti duomenys apdorojami programos pagalba ir toliau nuosekliai siunčiami atskiroms ašių valdymo žingsninių variklių valdikliams, kurie valdo pačius variklius įmontuotus prototipo platformoje (žr. **30 pav.**).

Bandymų metu pastebėta, jog valdant aukščiausia 40000 mikrožingsnių rezoliucija, varikliai nesugeba pasiekti reikiamos greitaveikos ir yra jaučiamas platformos judesio uždelsimas. Vienas iš sprendimų tokiai problemai išspręsti būtų naudoti lygiagretųjį valdymą, su atskiru mikroprocesoriumi kiekvienam varikliui valdyti. Atskirais *SimTools* programos komunikacijos kanalais kiekvienai ašiai (žr. **30 pav.**).



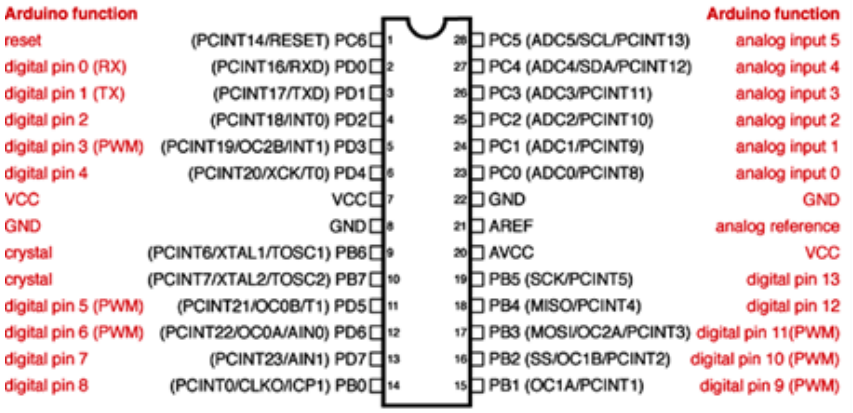
a) b)

30 pav. Prototipo valdymo architektūra a) nuoseklusis valdymas b) lygiagretusis valdymas

Kitas sprendimas būtų keisti projekte naudojama *Arduino Uno Rev 3* mikroprocesorių su *Arduino Due*, kuris yra 75% brangesnis, tačiau turi 5.25 kartų greitesnį procesorių [10].

2.5. Greitesnis platformos valdymas naudojant tiesioginį prievadų junginėjimą

Įmanomas kitas būdas, kuris nereikalauja papildomai nieko pirkti, tai tiesioginis prievadų manipuliavimo metodas. Vietoje naudojamos **digitalWrite** funkcijos galima tiesiogiai valdyti *Arduino* kontaktų prievadus (žr. **31 pav.**).

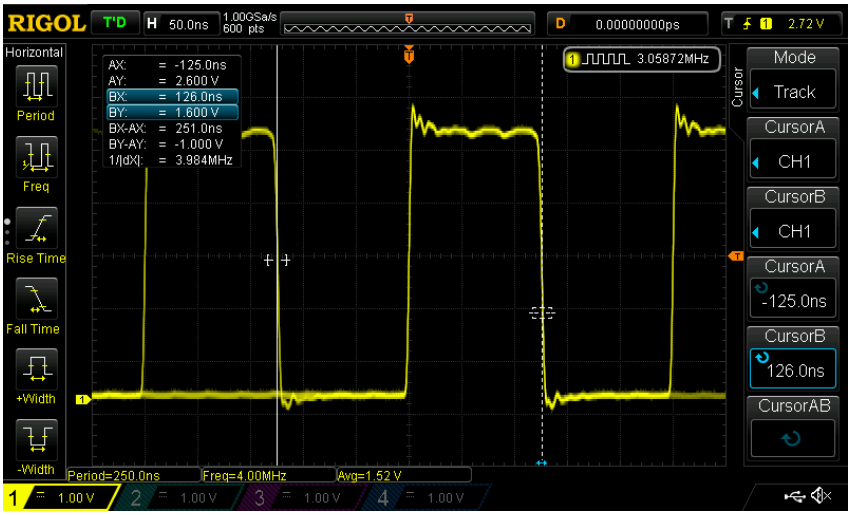


31 pav. *Arduino UNO Rev 3* procesoriaus kontaktų išdėstymas [5]

Norint pasiekti reikalingus kontaktų prievadus, reikia žinoti kur šie kontaktai yra. Žingsninių variklių žingsniavimo signalai gaunami iš 9, 10 ir 11 skaitmeninių kontaktų, kurie pagal schemą atitinkamai yra priskiriami prie **B** prievado. Tiesioginiam prievado atidarymui tiesiog **PORTB** kintamajam priskiriame baitą, kurio dešiniausias bitas šiuo atveju yra **PB0** aštuntasis *Arduino* kontaktas, o kairiausias yra **PB7** procesoriaus kvarcinio generatoriaus reikšmė. Taigi, jeigu priskiriamas baitas 00001110 prie kintamojo **PORTB**, tada 9, 10 ir 11 skaitmeninio išėjimo kontaktai yra įjungiami[5].

Lyginant su paprastu **digitalWrite** metodu, gautas beveik 18 kartų greitesnis junginėjimas, naudojant vieną kontaktą. Paprasta **digitalWrite** operacija užtrunka lėčiau, kadangi turi patikrinti bei įvykdyti kelias sąlygas kaip [5]:

- patikrinti ar egzistuoja naudojamas *Arduino* kontaktas;
- patikrinti ar naudojamame *Arduino* kontakte nėra veikiančio *PWM*, jeigu taip, jis sustabdomas;
- rasti bitų maskavimą pasirinktam *Arduino* kontaktui;
- rasti prievadą, kuris yra susijęs su pasirinktu *Arduino* kontaktu;
- patikrinti prievado būseną ir atitinkamai ją pakeisti.

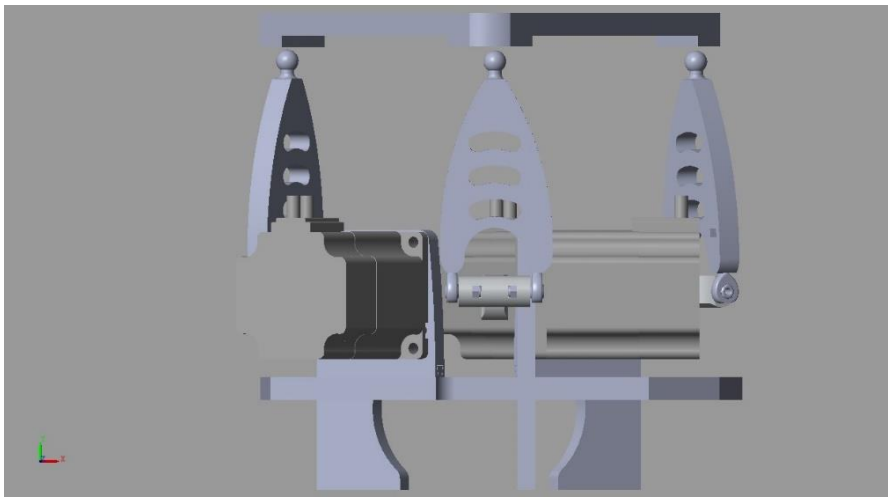


32 pav. **digitalWriteFast** tiesioginio prievado junginėjimo metodo impulso trukmė [12]

Patogesniam valdymui projekte panaudota **digitalWriteFast.h** *Arduino* biblioteka, kurioje yra paruoštos kontaktų konfigūravimo bei kontaktų junginėjimo komandos naudojant tiesioginį prievado junginėjimą. Lyginant su projekte naudojama **digitalWrite** komanda, kurioje apribotas stabilus veikimas iki apie 14 mikrosekundžių vienam impulso periodui [5], su **digitalWriteFast** šis periodas sumažėja iki apie 2.5 mikrosekundžių, kas yra 5.6 kartų greičiau (žr. **32 pav.**). Impulso trukmė skaičiuojama 10 kartų ilgiau, negu aukščiau pavaizduotame paveikslėlyje, kadangi *Arduino* programa turi spėti praeiti ciklą. Praktinių bandymu metu, iškviečiant **digitalWriteFast** komandą būtent 10 kartų, žingsninių variklių valdikliai pradeda valdyti prototipą. Projekto prototipo tiksliam ciklo laiko skaičiavimui, matavimai su osciloskopu nebuvo atlikti, tačiau remiantis dvyliktuoju šaltiniu galime sakyti, jog vienas impulso periodas trunka apie 0,25 mikrosekundžių, naudojant **digitalWriteFast.h** bibliotekos komandomis (žr. **32 pav.**).

2.6. *SolidWorks* prototipo modelio importavimas į *Simscape* aplinką

Programa *Matlab* turi galimybę importuoti suprojektuotą prototipo *CAD* failą tiesiai į *Simulink* aplinką tolimesniam tyrimui. Naudojant *Simscape Multibody Link* papildinį, kuris įdiegiamas į *SolidWorks*, galima automatiškai konvertuoti suprojektuotą prototipą *XML* formatu. Tada belieka tik pasinaudoti **smimport** funkcija *Matlab* aplinkoje, nurodant vietovę į gautą *XML* failą ir automatiškai sugeneruotas modelis (žr. **33 pav.**) [21].

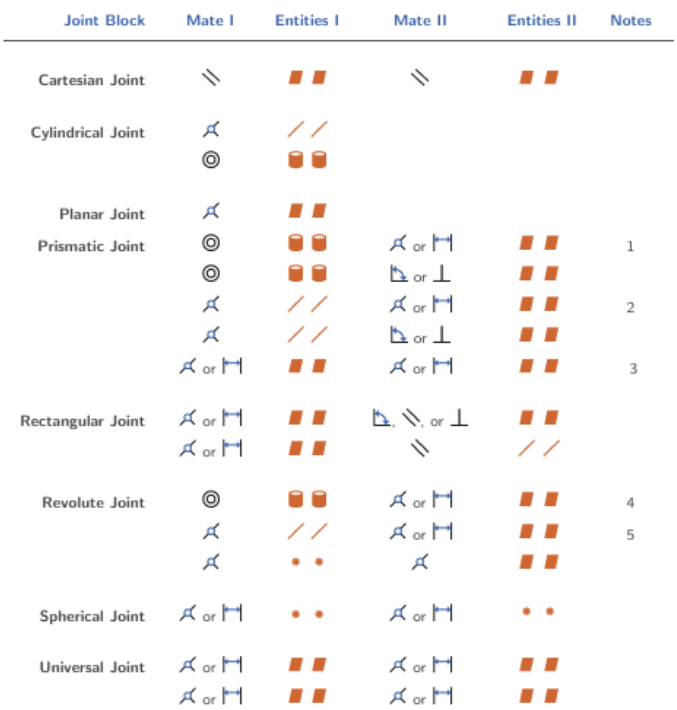


33 pav. Sugeneruotas prototipo modelis *Matlab Simscape* aplinkoje

Svarbu atkreipti dėmesį į tai, jog nėra palaikomi visi detalių sąlyčio suporavimo metodai (žr. **34 pav.**). Todėl prototipo modeliui taikoma prielaida, jog tarp sferinio sąnario ir platformos duobes tarpo nėra, kadangi nėra palaikomas tangentinis detalių porų jungimo metodas [21].

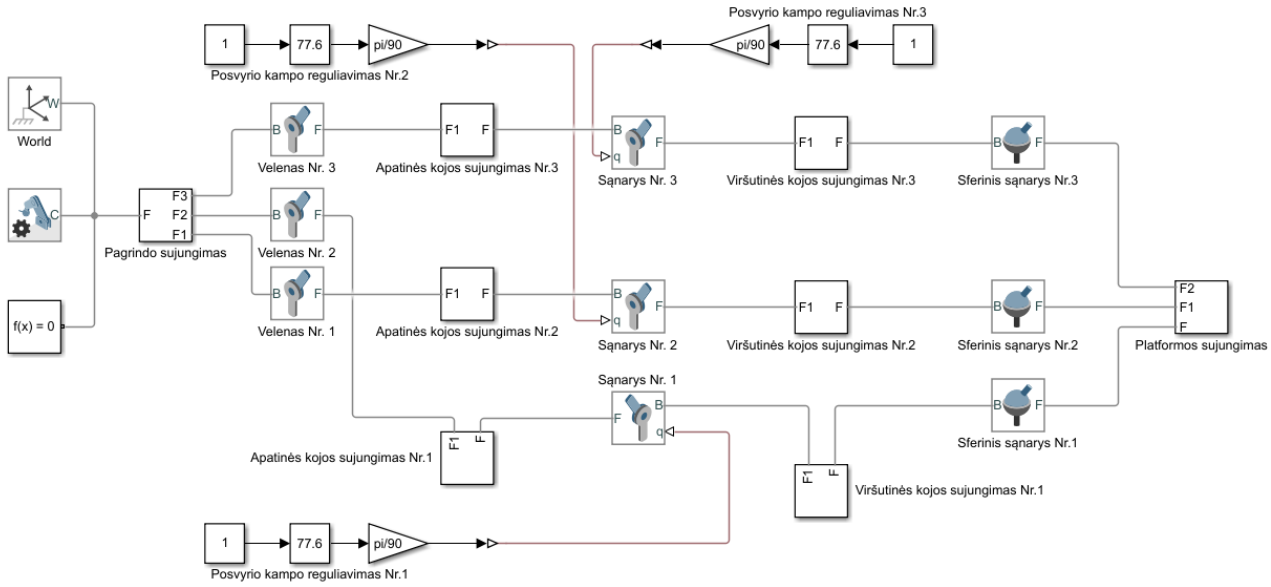
34 pav. figūrų tipai bei palaikomi detalių porų sujungimo būdai *Matlab Simscape* aplinkoje [21]

Taip pat, priklausomai nuo sąnarių jungimų, ne visi porų jungimų tipai yra tinkami (žr. **35 pav.**) [21].



35 pav. Palaikomi porų jungimai ir figūrų tipai pagal sąnario jungimo tipą *Simscape* modeliui [21]

Jeigu *SolidWorks* modelis sumodeliuotas su palaikomomis poromis, importuotą *Simscape* modelį *Simulink* aplinkoje (žr. **36 pav.**) galima toliau tyrinėti, keičiant sąnarių ašies pozicijas ir stebint kaip kinta platformos pozicija.

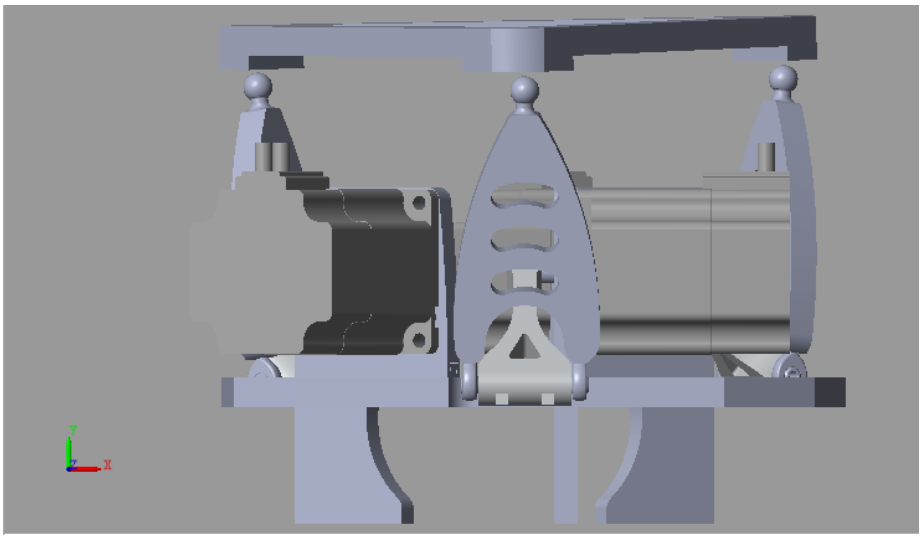


36 pav. Importuotas prototipo modelis *Simulink* aplinkoje

3. Tiriamoji dalis

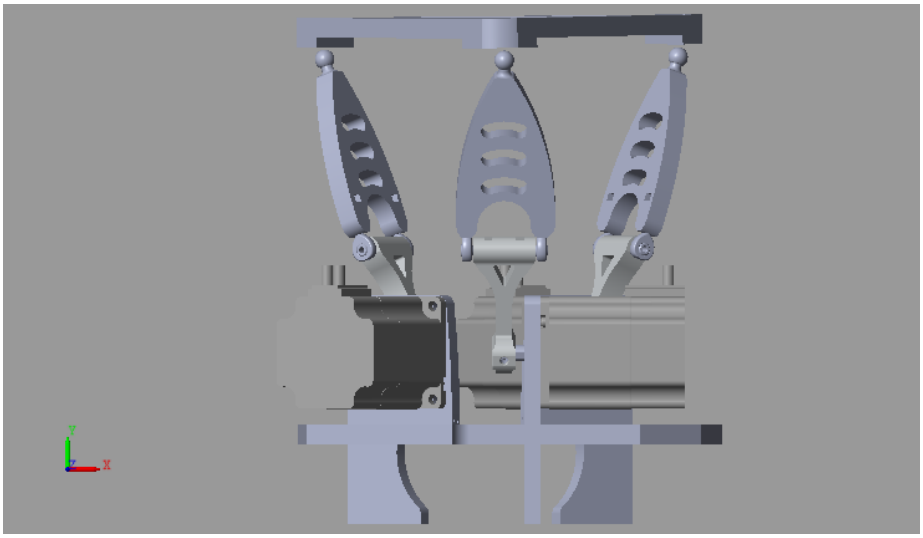
3.1. Prototipo modelio judėjimo amplitudės tyrimas

Tiriant projekto prototipo modelio judėjimo amplitudę, panaudota importuota *Simscape* blokinė schema *Simulink* aplinkoje. Keičiant sąnarių įvesties parametrus sugeneruojama nauja prototipo modelio platformos pozicija. Įvedant lokalios ašies sąnario pasisukimo dydį laipsniais visoms trims ašims, nustatyta, jog platforma yra pilnai nusileidusi ir liečia savo fizinę ribą su pagrindu ties nustatytais 92,4° (žr. **37 pav.**). Pats ašies pasisukimo atskaitos taškas vertinamas pagal vidurinį šarnyrinės jungties sąnarį.



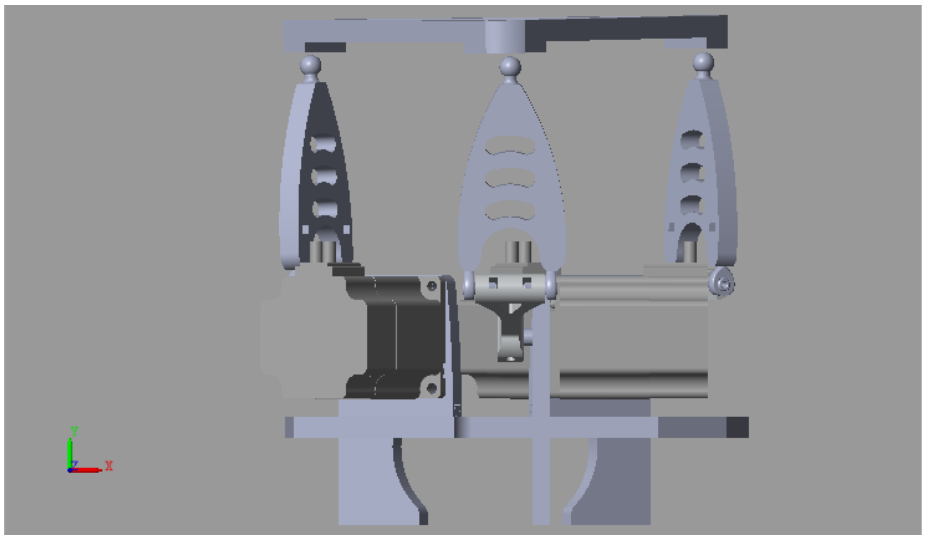
37 pav. Sugeneruotas žemiausioje nusileidimo pozicijoje *Simscape* modelis

Toliau keičiant sąnarių lokalių ašių pasisukimo įvesties parametrus, gauta, jog platforma pasiekia maksimalią viršutinę poziciją ties 218° įvedama reikšme (žr. **38 pav.**). Toliau reikšmę didinant gaunamas kinematinis singuliarumas ir nebeįmanoma sugeneruoti modelio.



38 pav. Aukščiausioje pakilimo pozicijoje sugeneruotas *Simscape* modelis

Atlikus tokį tyrimą, apibrėžta 125,6° platformos posvyrio amplitude. Suskaičiavus skirtumą tarp sugeneruoto platformos modelio lokalių sąnarių reikšmių pasiekus žemiausią bei aukščiausią poziciją. Taigi įvedus judėjimo amplitudės vidurinę reikšmę 155,2° lokaliose ašyse, sugeneruotame modelyje gaunama platformos neutrali pozicija. Šioje pozicijoje galime turėti lygiavertį neigiamą arba teigiamą laisvės laipsnių judėjimo potyrį (žr. **39 pav.**).



39 pav. Sugeneruotas neutralios pozicijos *Simscape* modelis

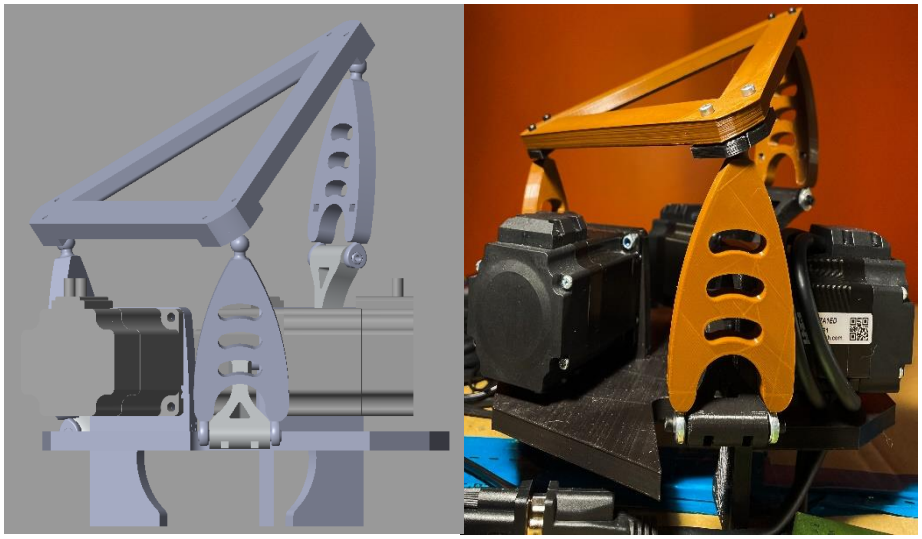
Testuojant fizinį prototipą, naudojantis *SimTools* įrankio pagalba, nustatyta, jog judėjimo amplitudė tarp žemiausios ir aukščiausios pozicijos yra 82 milimetrai. Tai rodo, jog platforma iš neutralios pozicijos gali judėti 41 milimetrų amplitude į viršų arba apačią (žr. **40 pav.**). Lyginant rinkoje esančių judesio platformų judėjimo amplitudę, gauta vidutinė reikšmė yra lygi 147 milimetrų [2],[17],[19]. Prototipo platformos aukščio nustatymui panaudota lygiagrečiai pastatyta liniuotė ant platformos pagrindo.

a) b)

40 pav. Fizinio prototipo *Heave* judesio amplitudė naudojant *SimTools* programą a) -100% b) +100%

3.2. Prototipo posvyrių tyrimas

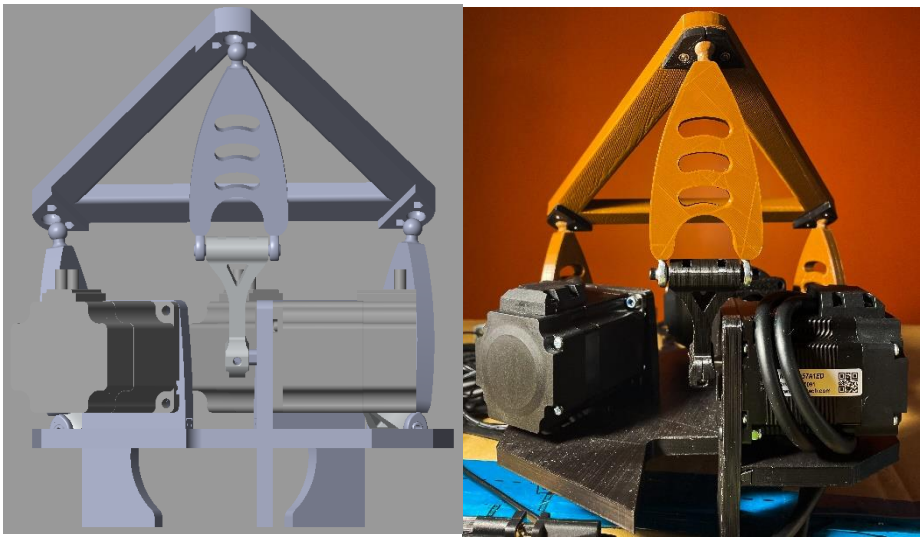
Fizinio prototipo *SimTools* programa imituojamiems *Pitch* ir *Roll* judesiams su *Simulink* generuojamo modeliu posvyriams palyginti, pritaisytas telefonas ant platformos vidurio su įjungtu giroskopo matavimo įrankiu. Atliekant *Pitch* +100% posvyrio judesį *SimTools* įrankiu, gauta, jog kampas yra lygus 29°. Prototipo *Simulink* modelio aplinkoje, atitinkamos pozicijos gautos įvedus vienoje lokalioje sąnario ašyje mažiausią galimą laipsnių vertę, o kitose dviejose ašyse – didžiausią (žr. **41 pav.**).



a) b)

41 pav. Modelio ir fizinio prototipo *Pitch* judesio posvyrio palyginimas a) *Simscape* b) *SimTools*

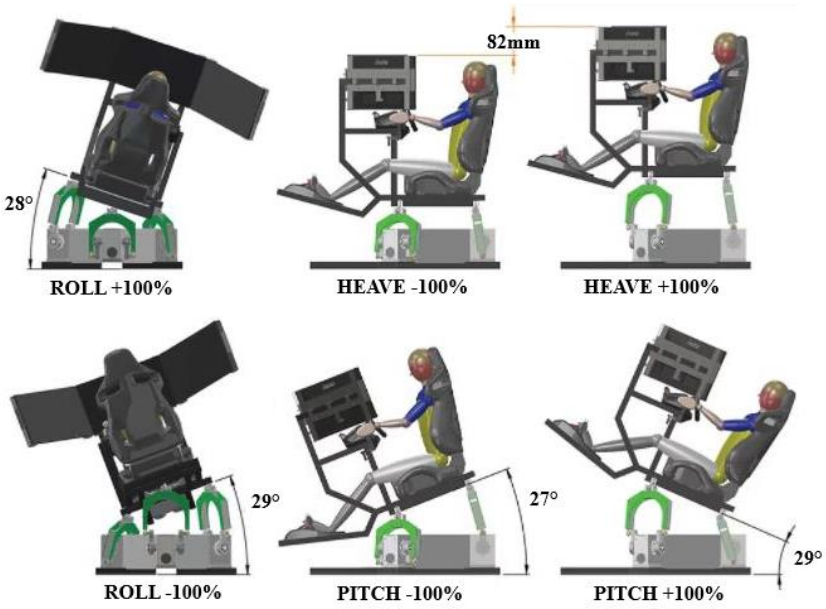
Atliekant *Roll* +100% posvyrio judesį *SimTools* įrankiu, gauta, jog kampas yra lygus 28° (žr. **42 pav.**).



a) b)

42 pav. Modelio ir fizinio prototipo *Roll* judesio posvyrio palyginimas a) *Simscape* b) *SimTools*

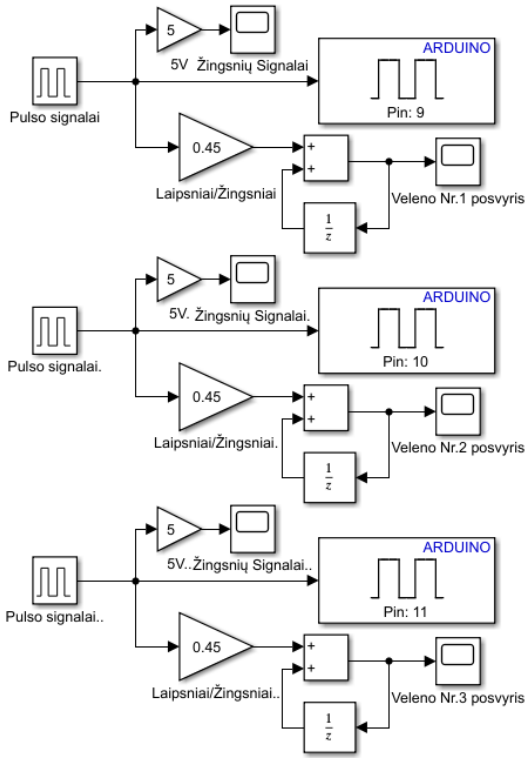
Atlikus posvyrio tyrimus platformai veikiant visomis įmanomomis laisvės laipsnių ašių judėjimo ir posvyrio kryptimis, skirtingose platformos pozicijose, nustatyta, jog modelio generuojami posvyriai vizualiai atitinka fizinio prototipo gautus posvyrius. Lyginant rinkoje esančių judesio platformų maksimalaus posvyrio galimybes, gauta, jog vidutinė maksimali reikšmė yra lygi 33,65° [2],[17],[19]. Visi likę atlikti prototipo matavimai pilnai pateikti trečiajame priede (žr. **43 pav.**).



43 pav. Fizinio prototipo judėjimo amplitudė ir galimi posvyriai pagal trečiojo priedo tyrimo duomenis

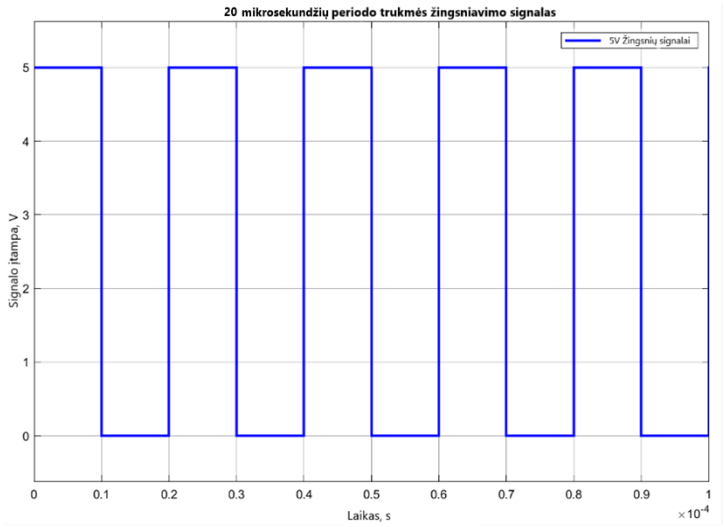
3.3. Prototipo greitaveikos tyrimas

Teorinio modelio bei fizinio prototipo greitaveikos palyginimui panaudota *Simulink Arduino* sąsajos papildinys. Jis tiesiogiai susieja mikrovaldiklio generuojamus signalus su *Simulink* modelio blokinėmis diagramomis (žr. **44 pav.**). Fizinio prototipo greitaveikos tyrimui naudota *SimTools* įrankio tiesioginio laisvės laipsnių valdymo sąsaja ir chronometras, matuojantis tūkstantosios sekundės dalies tikslumu.



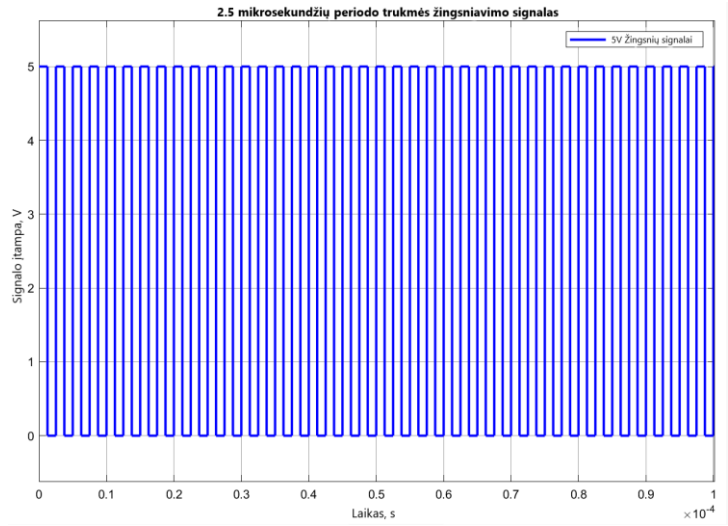
44 pav. *Simulink* modelis su *Arduino* sąsajos prototipo valdymu

Prototipo greičio palyginimui panaudoti įprastas **digitalWrite** bei **fastDigitalWrite** valdymo būdai. Remiantis teoriniais duomenimis, įprastu **digitalWrite** metodu vienas impulsas trunka 10 mikrosekundžių (žr. **45 pav.**).



45 pav. Teorinis **digitalWrite** metodo impulsų generavimas

Tačiau naudojant **fastDigitalWrite** metodą gaunamas aštuonis kartus greitesnis 1,25 mikrosekundžių trunkantis impulsas (žr. **46 pav.**).

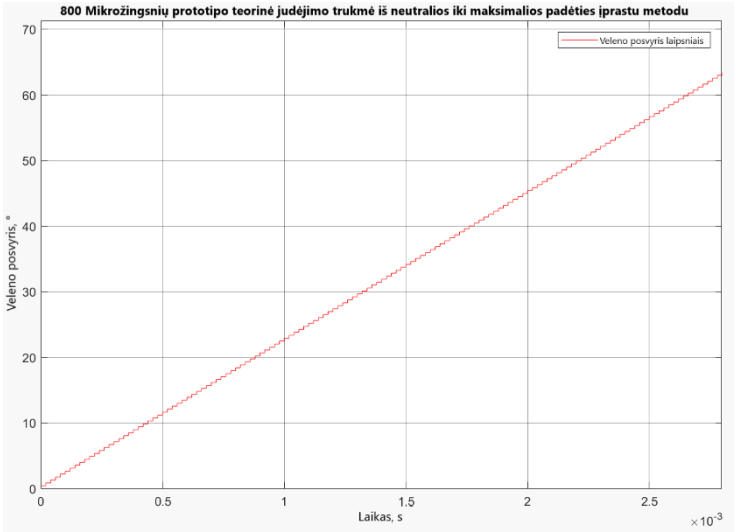


46 pav. Teorinis **fastDigitalWrite** metodo impulsų generavimas

Kadangi tyrimų duomenimis gauta, jog prototipo sąnario maksimali posvyrio amplitudė yra 125,6°. Greitaveikos tyrimui platformai judant iš neutralios pozicijos į *Heave* +100% padėtį nustatytas 62.8° veleno pasisukimo trukmės matavimas. Teorinio modelio ir fizinio prototipo greitaveikos tyrimas atliktas keičiant žingsninių variklių valdiklių mikrožingsnių skaičių vienam veleno apsisukimui, naudojant abu minėtus valdymo metodus. Tačiau fizinio prototipo tyrimui, dėl per didelio judėjimo greičio fizinei prototipo platformai , **fastDigitalWrite** metodu panaudoti tik trys aukščiausi mikrožingsnių nustatymai.

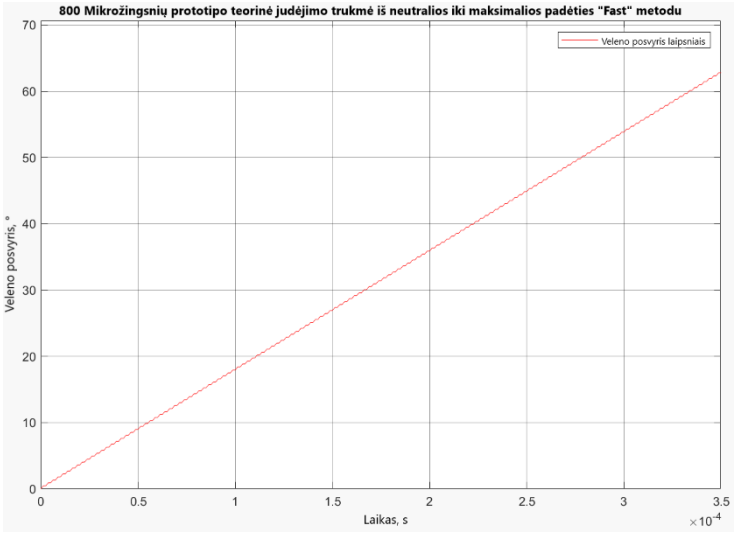
Judesio atlikimui panaudojant teorinį prototipo *Simulink* modelį, valdymas nėra atliekamas realiu laiku ir trunka iki kelių minučių. Todėl atliekant teorinį greitaveikos tyrimą įprastuoju **digitalWrite** bei **fastDigitalWrite** metodų impulso pločio signalais, galima atlikti visais žingsninių variklio valdiklių nustatymais.

Įprastuoju metodu ištirta, jog teoriškai variklio velenas pasisuka į užduotą padėti praėjus 2800 mikrosekundžių, nustačius 800 mikrožingsnių tikslumą žingsniniams varikliams (žr. **47 pav.**).



47 pav. Teorinė **digitalWrite** metodo veleno sukimosi trukmė 800 mikrožingsnių tikslumu

Greituoju metodu gauta, jog, teoriškai, variklio velenas užduotą padėtį pasiekia aštuonis kartus greičiau negu įprastuoju metodu – per 350 mikrosekundžių (žr. **48 pav.**).

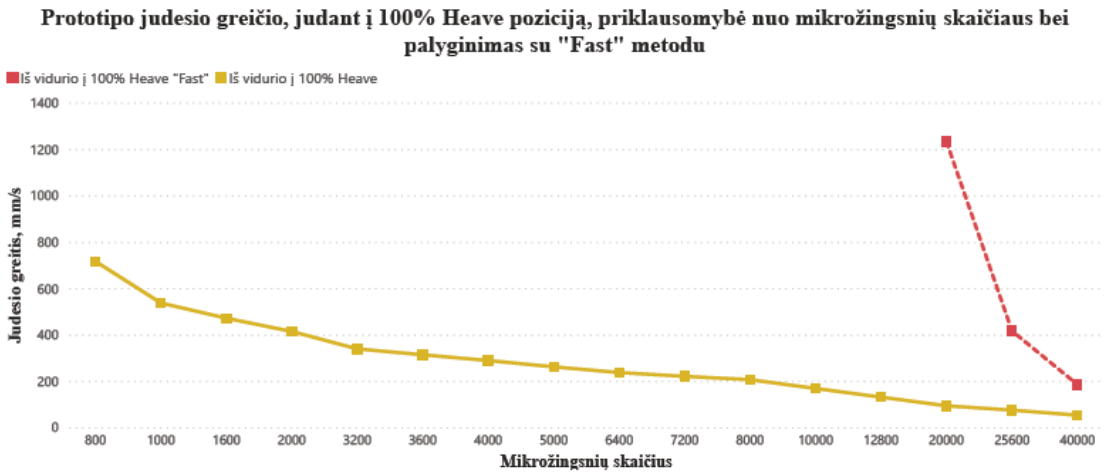


48 pav. Teorinė **fastDigitalWrite** metodo veleno sukimosi trukmė 800 mikrožingsnių tikslumu

Tokio pačio pobūdžio matavimai atlikti su fiziniu prototipu *SimTools* įrankio pagalba, naudojant tiesioginį, rankinį judesių manipuliavimo sąsają. Kiekviena laiko matavimo pradžia pradedama platformai esant neutralioje. Vidurinėje pozicijoje, atitinkamo laisvės laipsnio judėjimo krypties aktyvavimo metu. Tada, platformai pasiekus +100% laisvės laipsnio *Pitch*, *Roll* arba *Heave* judėjimo krypties galutinę poziciją, matavimo trukmė užfiksuojama chronometru. Kiekvienai judėjimo krypčiai, su visais įmanomais mikrožingsnių skaičiaus nustatymais, atlikti 4 matavimo bandymai, kurių vidurkis apskaičiuotas remiantis (6) šaknies vidurkio kvadrato formule (angl. *RMS*).

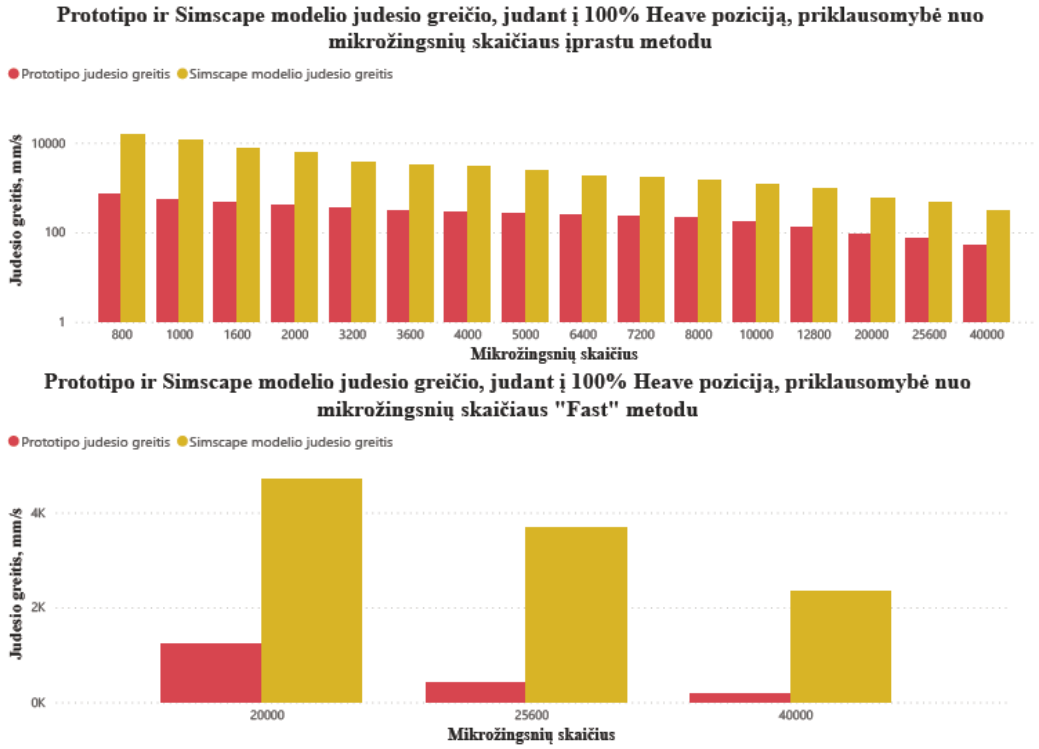
$$x_{RMS} = \sqrt{\frac{1}{4}({x_1}^2 + {x_2}^2 + {x_3}^2 + {x_4}^2)} \quad (6)$$

Platformos greičiui surasti, skaičiuotas santykis tarp nueito kelio ir gautos *RMS* trukmės (žr. **49 pav.**).



49 pav. fizinio prototipo judesio greičio priklausomybės nuo mikrožingsnių skaičiaus metodų palyginimas kylant iš neutralios pozicijos į viršutinę padėtį

Iš fizinio prototipo ir teorinio *Simscape* modelio greitaveikos duomenų palyginimo akivaizdu, jog fizinio prototipo greitaveika yra žymiai mažesnė, nei teorinio modelio. Skirtumas tarp jų didėja naudojant vis didesnį mikrožingsnių skaičiaus nustatymą. Nesvarbu koks valdymo metodas naudojamas (žr. **50 pav.**). Palyginimo duomenys pateikti nuo ketvirtojo iki šeštojo priedo.



50 pav. Judesio greičio priklausomybės nuo mikrožingsnių skaičiaus kylant iš neutralios pozicijos į viršutinę padėtį tarp fizinio prototipo ir teorinio modelio

Tokį skirtumą tarp teorinio ir fizinio prototipo galima įvardinti dėl kelių priežasčių:

a) plaukiojantis *Arduino* ciklas;
b) *UART* komunikacijos apribojimai;
c) ilgai trunkantis *Arduino* programos algoritmo valdymas;
d) *SimTools* komunikacijos apribojimai;

Arduino programoje nėra fiksuoto programos ciklo nustatymų, todėl kiekvienas ciklas gali būti skirtingos trukmės. Priklausomai nuo to, kaip programos kodas suformuluotas ir kuo daugiau instrukcinių kodo eilučių užrašoma, tuo ilgiau gali trukti kiekvienas ciklo periodas. Pati **digitalWrite** komanda vidutiniškai užtrunka 0,26 mikrosekundžių, tai reiškia programoje vieną kartą iškvietus valdymo impulso generavimo funkcijoje **monoPulse** sugaištama 0,52 mikrosekundės [5]. Valdymo platformai judant iš neutralios pozicijos į viršutinę būseną, naudojant 40000 mikrožingsnių nustatymą sugaištama apie 3628 mikrosekundžių, remiantis ketvirtojo priedo duomenimis.

Taip pat pagal *Arduino* valdymo algoritmą, fizinio prototipo varikliai nepradeda judėti, kol duomenys nėra pilnai perduodami *UART* komunikacijos protokolu ir apdoroti pačios programos. Pagal *UART* komunikacijos nustatymus, galima siųsti 8 bitų duomenų paketą 500000 bitų per sekundę greičiu. Kadangi kiekvienas baitas siunčiamas su *Start* ir *Stop* papildomais bitais, kiekvienas duomenų paketas siunčiamas su dvejais papildomais bitais (be papildomojo duomenų tikrinimo pariteto bito).

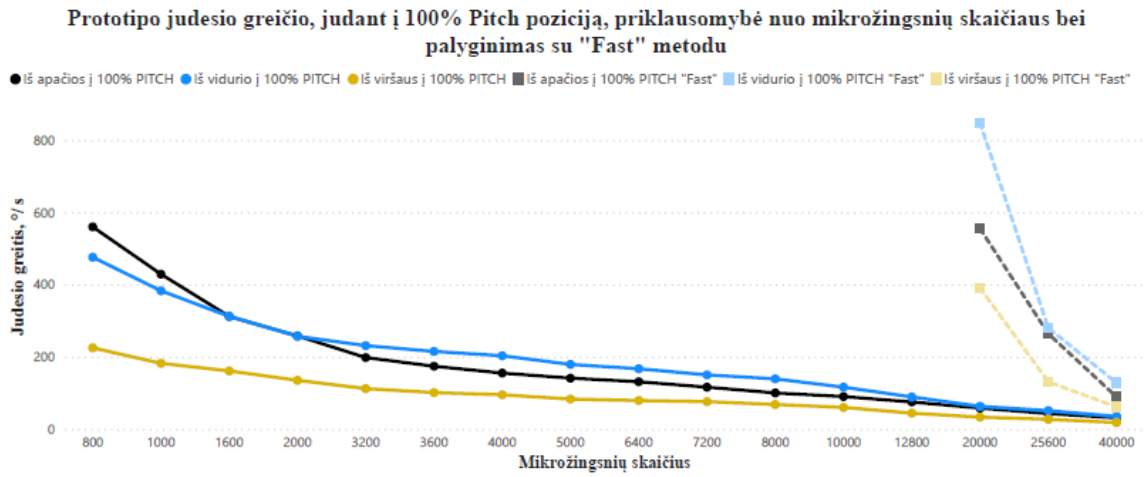
$$\frac{1(bitas)}{500000(\frac{bitai}{s})} = 2000ns$$

Kiekvienas duomenų paketo siuntimas trunka 20 mikrosekundžių ($10bit * 2000ns = 20\mu s$).
SimTools įrankiu komunikacija perduodami ašies pozicijos duomenys susideda iš 15 simbolių.

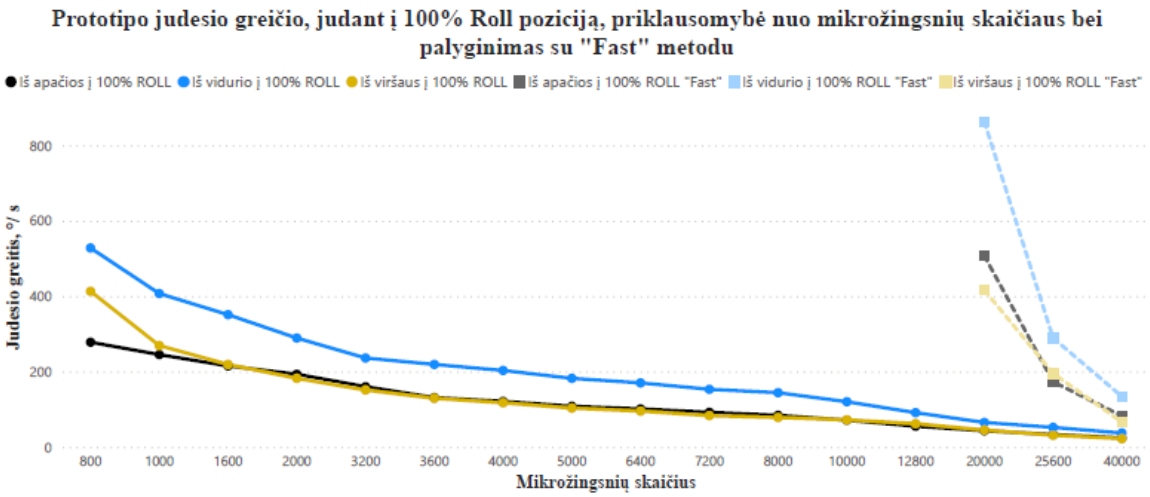
A	2	5	5	~	B	2	5	5	~	C	2	5	5	~
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Šie simboliai perduodami per komunikaciją atskirais duomenų paketais, tai vėl užtrunka papildomas 300 mikrosekundžių ($15 * 20\mu s = 300\mu s$). Ašies pozicijos duomenų atnaujinimo minimalus periodas *SimTools* programoje yra kas 1000 mikrosekundžių. Kadangi programoje ašies pozicijos duomenys yra 8 bitų diapazone, ($2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 = 255$) reiškia, jog platformos neutralioji pozicija yra ties diapazono reikšme 128. Taigi vien komunikacijos apdorojimui galima užtrukti iki 0,1664 sekundės ($1300\mu s * 128 = 166400\mu s = 0{,}1664s$). Tačiau priklausomai nuo to kaip greitai simuliacijos modelis juda, *SimTools* programai atnaujinant prototipo pozicijos duomenis, gali būti praleidžiamos tarpinės reikšmės, todėl gali užtrukti ir trumpiau.

Toliau tiriant prototipo greitaveiką, atlikti matavimai, platformos posvyrio greičiui nustatyti, judant iš apatinės, vidurinės ir viršutinės pozicijos į +100% *Pitch* ir *Roll* poziciją. Posvyrių greičiai nustatyti skaičiuojant santykį tarp gautų trečiojo priedo posvyrių dydžių ir užtrukusio laiko penktojo ir šeštojo priedo duomenų (žr. **51 pav.** ir **52 pav.**).



51 pav. Fizinio prototipo posvyrio greičio priklausomybės nuo mikrožingsnių skaičiaus metodų palyginimas iš visų pozicijų į šoninę *Pitch* padėtį



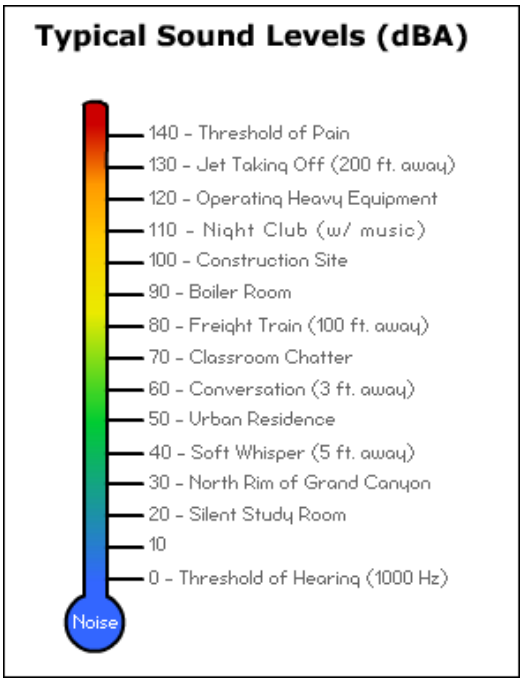
52 pav. Fizinio prototipo posvyrio greičio priklausomybės nuo mikrožingsnių skaičiaus metodų palyginimas iš visų pozicijų į šoninę *Roll* padėtį

Lyginant rezultatus matoma, jog prototipo platformai esant neutralioje pozicijoje, gaunama didžiausia greitaveika greituoju bei įprastuoju metodu. Ties 40000 mikrožingsnių nustatymu prototipo platformos posvyrio greitis valdant **fastDigitalWrite** metodu gautas vidutinis platformos posvyrio greitis 131°/s. Greitis judant į 100% *Pitch* poziciją siekia 128°/s, o judant į 100% *Roll* poziciją siekia 134°/s. Tai yra labai artimas greitis, lyginant simuliacinių judesio platformų rinkos vidutiniu posvyrio greičiu, kuris yra 170,4°/s [2], [16], [17], [19], [26].

3.4. Prototipo garso lygio tyrimas

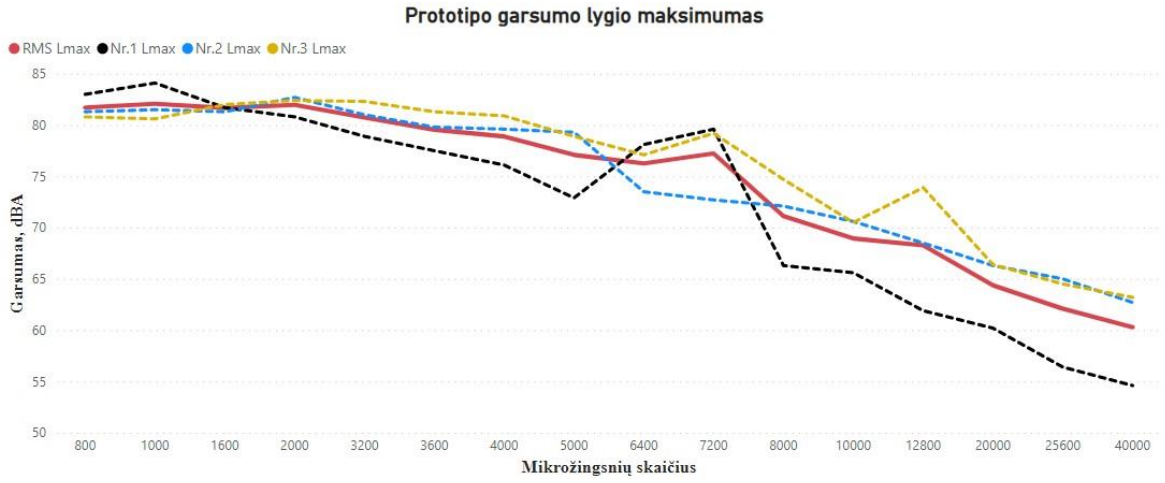
Vienas svarbiausių judesio platformų kokybinių parametrų yra skleidžiamas garso lygis. Garsas matuojamas **A** svertiniais decibelais, arčiausiai atitinka žmogaus ausies garsumo suvokimą. Decibelai matuojami logaritmine skale, o tai reiškia, jog net ir nedidelis decibelų pokytis, gali lemti didžiulį triukšmo kiekio pokyti ir gali sukelti pavojų žmogaus klausai. Pagal *OSHA* darbų saugą, aštuonių

valandų darbo dienos leistina poveikio riba yra 90 dBA [13]. Sekančiame paveikslėlyje pateikta triukšmo lygių palyginimas su žmogaus ausies jaučiamomis situacijomis (žr. **53 pav.**).



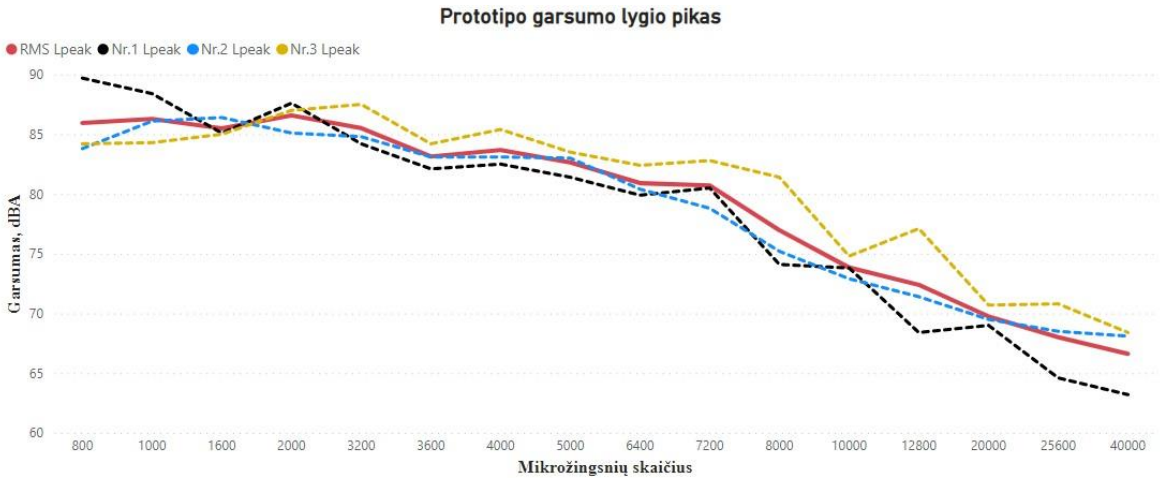
53 pav. Garsų palyginimas A svertiniais decibelais [13]

Prototipo garso matavimui panaudotas mikrofonas su A dažnio svorio filtru, kuris matuoja garso lygio maksimumą, vidurkį bei piką. Bandymo metu mikrofonas įrašinėjo 30 centimetrų atstumu nuo prototipo platformos 30 sekundžių kiekvienam mikrožingsnių skaičiaus nustatymu. Tuo tarpu prototipo platforma įrašinėjimo metu su *SimTools* programa rankiniu būdu buvo judinama įvairiomis laisvės laipsnių kryptimis pilna judėjimo amplitude. Iš viso atlikti trys tokie bandymai ir pagal (6) formulę surastos *RMS* vertės.



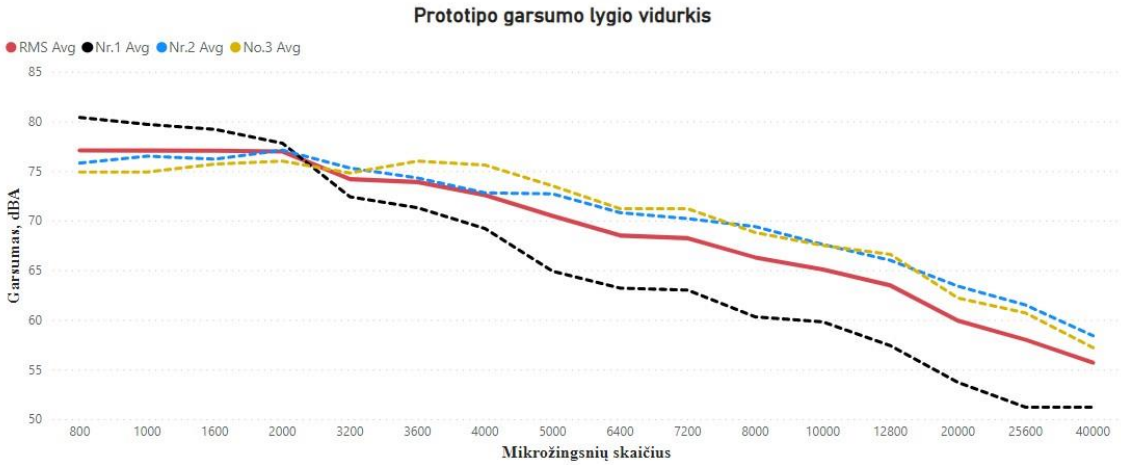
54 pav. Fizinio prototipo garso lygio maksimumo priklausomybė nuo mikrožingsnių skaičiaus

Garso lygio maksimumas matuojamas tam tikrais nustatytais laiko intervalais ir apskaičiuojama aptikta didžiausio garso lygio atitinkamo laiko intervalo *RMS* vertė. Remiantis trečiojo priedo duomenimis, ties 40000 mikrožingsnių skaičiaus nustatymu, gauta 60,3dBA reikšmė (žr. **54 pav.**).



55 pav. Fizinio prototipo garso lygio piko priklausomybė nuo mikrožingsnių skaičiaus

Tuo tarpu garso lygio pikas tai yra maksimali užfiksuota garso bangos slėgio momentinė reikšmė. Remiantis trečiojo priedo duomenimis, ties 40000 mikrožingsnių skaičiaus nustatymu, gauta 66,6dBA reikšmė (žr. **55 pav.**).



56 pav. Fizinio prototipo garso lygio vidurkio priklausomybė nuo mikrožingsnių skaičiaus

Toliau gautas garso lygio vidurkis, kuris pagal trečiojo priedo duomenimis, ties 40000 mikrožingsnių skaičiaus nustatymu, yra lygus 55,7dBA. Remiantis devynioliktuoju literatūros šaltiniu, toks triukšmo lygis yra visiškai normalus, kadangi šiame šaltinyje judesio platforma generuoja 60dBA garso lygį (žr. **56 pav.**).

Išvados

1. Atlikus literatūros analizę, nustatyta, jog trijų laisvės laipsnių judesio platformos įgyvendinimui tinkama *RRS* kinematinių grandžių lygiagretaus manipuliatoriaus konstrukcija. Būtent tokios architektūros mechanizmas yra vienas iš paprasčiausių, su dvigubai mažesniu vykdiklių kiekiu ir gali atlikti panašius judesius kaip Stiuarto platforma. Tokio tipo prototipo platforma gali judėti lygiagrečiai su O_z ašimi ir atlikti posvyrius O_x ir O_y ašimis.
2. Suprojektuotos šešios unikalios detalės, kurios buvo pritaikytos pagal užsakytus žingsninius variklius ir tvirtinimo elementus, naudojantis *SolidWorks* programine įranga. Taip pat suprojektuota elektros schema skirta *Arduino* mikroprocesoriaus ir žingsninių variklių sujungimui bei jų maitinimui. Realizuota prototipo judesio valdymo ir komunikacijos duomenų apsikeitimo programa tarp *Arduino* ir *SimTools* simuliacijos duomenų generavimo įrankio.
3. Tiriant prototipo sugeneruotą modelį *Matlab Simulink* aplinkoje su *Simscape* sąsajos blokais, nustatyta, jog vykdiklių velenų maksimali leistina posvyrio amplitudė yra 125,6°. Toliau tiriant modelį su *Arduino* sąsaja, nustatyta teorinė 2800 mikrosekundžių trukmė judant iš vidurinės padėties į viršutinę įprastuoju metodu ir 350 mikrosekundžių trukmė tiesioginiu prievado junginėjimo metodu. Atliekant bandymus su fiziniu prototipu, nustatyta, jog greitesniuoju metodu valdomi vykdikliai pasiekia 134°/s platformos posvyrio greitį lyginant su rinkoje esančių judesio platformų vidutiniu posvyrio greičiu 170,4°/s. Toliau tiriant prototipo maksimalias platformos posvyrio bei judesio amplitudes, gauta, jog daugiausiai pakrypsta per 29° ir juda 82 milimetrų amplitude. Tuo tarpu lyginant rinkoje esančias platformas gauta vidutinė maksimali 33,65° posvyrio ir 147 milimetrų judėjimo amplitudės reikšmė Taip pat įvertintas prototipo skleidžiamo garso lygio vidurkis 55,7dBA, kuris neviršija rinkoje esančių judesio platformų 60dBA garso lygio.

Literatūros sąrašas

1. 3DOF Ball on Plate using Closed Loop Stepper Motors. (2016 m. Gegužės 15 d.). Žiūrėta 2022–01–15. Prieiga per internetą: https://www.instructables.com/3DOF-Ball-on-Plate-Using-Closed-Loop-Stepper-Motor/
2. 3DOF Platform. Žiūrėta 2022–01–15. Prieiga per internetą: https://active-game.com/3dof-platform/
3. Abhilash Nayak, Stéphane Caro, Philippe Wenger, Comparison of 3-[PP]S Parallel Manipulators based on their Singularity Free Orientation Workspace, Parasitic Motions and Complexity. (2018 m. Rugpjūčio 21 d.). Žiūrėta 2022–01–15. Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S0094114X1830630X
4. Arduino code for steppers? (2015 m. Vasario 5 d.). Žiūrėta 2022–01–15. Prieiga per internetą: https://www.xsimulator.net/community/threads/arduino-code-for-steppers.5434/
5. Arduino Fast digitalWrite. Žiūrėta 2022–01–15. Prieiga per internetą: https://roboticsbackend.com/arduino-fast-digitalwrite/
6. Balčiūnas Audrius, Dviejų Laisvės Laipsnių Paralelinio Roboto Projektavimas Ir Tyrimas. (2015 m.). Žiūrėta 2022–01–15. Prieiga per internetą: https://epubl.ktu.edu/object/elaba:8656000/8656000.pdf
7. DongGyu Kim, Sung-Ho Hwang, Kinematic Implementation of 3-DOF 2-Link Type Vehicle Simulator. (2020 m.). Žiūrėta 2022–01–15. Prieiga per internetą: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9260532&tag=1
8. Eisele Robert, Inverse Kinematics of a Stewart Platform. (2016 m. Gegužės 15 d.). Žiūrėta 2022–01–15. Prieiga per internetą: https://www.xarg.org/paper/inverse-kinematics-of-a-stewart-platform/
9. Gapšys Mindaugas, Roboto Su Uždara Kinematine Grandine Valdymo Galimybių Tyrimas. (2017 m.). Žiūrėta 2022–01–15. Prieiga per internetą: https://talpykla.elaba.lt/elaba-fedora/objects/elaba:22736087/datastreams/MAIN/content
10. How Fast is an Arduino: Guide to Arduino Speeds. Žiūrėta 2022–01–15. Prieiga per internetą: https://chipwired.com/arduino-speed-guide/
11. Jianfeng Li, Jinsong Wang, Wusheng Chou, Yuru Zhang, Tianmiao Wang, Qixian Zhang, Inverse Kinematics And Dynamics Of The 3-Rrs Parallel Platform. (2001 m. Vasaris). Žiūrėta 2022–01–15. Prieiga per internetą: https://www.researchgate.net/publication/3902626_Inverse_kinematics_and_dynamics_of_the_3-RRS_parallel_platform
12. Nickson Yap, DigitalWriteFast. (2020 m. Sausio 27 d.). Žiūrėta 2022–01–15. Prieiga per internetą: https://github.com/NicksonYap/digitalWriteFast
13. Occupational Noise Exposure. Žiūrėta 2022–03–18. Prieiga per internetą: https://www.osha.gov/noise
14. Parinkti komponentai prototipo gaminiui. Žiūrėta 2022–01–15. Prieiga per internetą:
    - Arduino UNO R3: https://www.anodas.lt/arduino-uno-r3-lt?search=arduino%20uno%20r3
    - Žingsninis variklis: https://www.aliexpress.com/item/33016826807.html?spm=a2g0o.9042311.0.0.21db4c4dOapgiE

- Šarnyrinės galvutės:
https://www.anodas.lt/sarnyras-m4
- Maitinimo šaltinis:
https://www.lemona.lt/impulsinis-maitinimo-saltinis-24v-14-6a-uzdaras-mean-well.html
- Veržlės:
https://shop.anjese.lt/lt/tvirtinimo-detales/verzles/verzle-su-nail-din985?page=3
- Varžtai:
https://shop.anjese.lt/lt/tvirtinimo-detales/varztai-sraigtai/varztas-din6912
- Laidai:
https://www.senukai.lt/p/kabelis-lietkabelis-mounting-wires-pv3-1x0-75-h05v-k-blue/ay4y?cat=94a&index=1

15. Parinkti komponentai realaus dydžio gaminiui. Žiūrėta 2022–01–15. Prieiga per internetą:
- Reduktorius:
http://elektros-prekes.lt/063-gabaritas/112-reduktorius-063-1-100-pam71b14.html
- Sferinė jungtis:
https://www.igus.si/product/17824?artNr=GFSM-10-AG-ES
- Stabdymo varža
https://www.dazniokeitiklis.lt/stabdymo-varza-300w/
- Servo variklis:
https://www.ebay.com/itm/322767295145?hash=item4b266e12a9%3Ag%3AsusAAOSwOTVZpSDm&LH_ItemCondition=3
- DB25 komunikacijos jungtis:
https://www.amazon.de/-/en/PENGLIN-Solderless-Connector-Breakout-Accessories/dp/B096P2LZB8/ref=sr_1_3?crid=2HUHOXB7W6YBH&keywords=PENGLIN%2BDB25&qid=1648407724&sprefix=penglin%2Bdb25%2Caps%2C80&sr=8-3&th=1
- Avarinis mygtukas:
https://narvija.com/guoliai/sarnyrines-galvutes/sarnyrine-galvute-sikac-10-m
- Šarnyrinės galvutės:
https://narvija.com/guoliai/sarnyrines-galvutes/sarnyrine-galvute-sikac-10-m

16. Professional P3. Žiūrėta 2022–01–15. Prieiga per internetą: https://dofreality.com/
17. PS-3TM-200. Žiūrėta 2022–01–15. Prieiga per internetą:
https://motionsystems.eu/product/motion-platforms/ps-3tm-200/
18. Račiūnas Dovydas, Stiuarto platformos judesio modeliavimas. (2015 m.). Žiūrėta 2022–01–15. Prieiga per internetą:
https://epubl.ktu.edu/object/elaba:8650593/8650593.pdf
19. RC4 4DOF. Žiūrėta 2022–01–15. Prieiga per internetą:
https://fasetech.com/shop/catalog/racingcube-m1/slider-c1/rc4-4dof-p12
20. Servo technology. Žiūrėta 2022–01–15. Prieiga per internetą:
https://motionforsimulators.com/servo-technology/
21. Simscape Multibody Link. (2018 m.). Žiūrėta 2022–01–15. Prieiga per internetą:
https://ww2.mathworks.cn/help/physmod/smlink/index.html?s_tid=CRUX_lftnav
22. Šmaižys Vaidotas, Automobilių Sporto Simuliatoriaus Su Judesio Imitacija Kūrimas. (2016 m.). Žiūrėta 2022–01–15. Prieiga per internetą:
https://talpykla.elaba.lt/elaba-fedora/objects/elaba:16213875/datastreams/MAIN/content

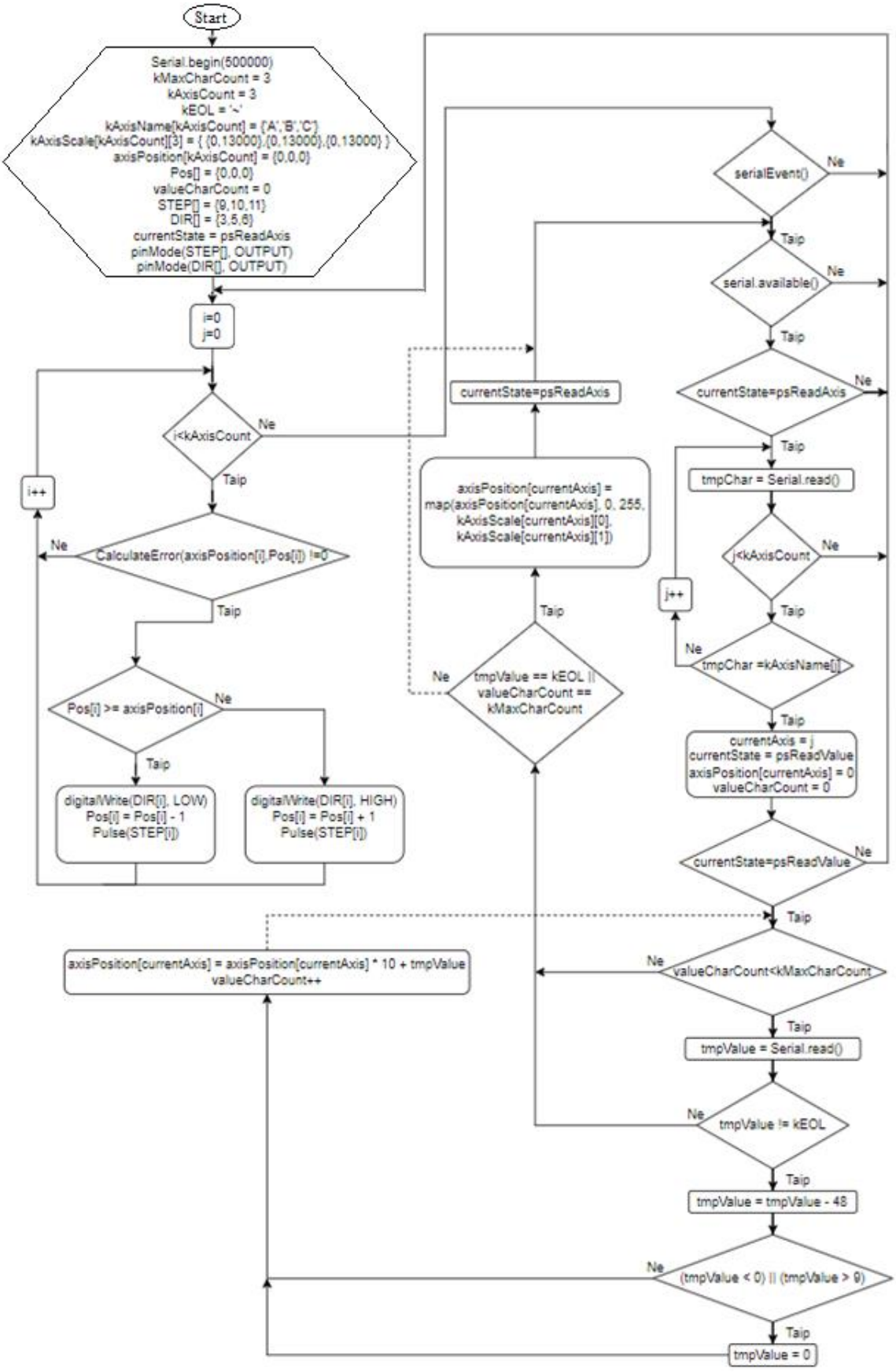
23. Tetik Halil, Modelling And Control Of A 3-RRS Parallel Manipulator. (2016 m. Liepa) Žiūrėta 2022–01–15. Prieiga per internetą:
https://gcris.iyte.edu.tr/bitstream/11147/2831/1/T001464.pdf
24. UART Explained. Žiūrėta 2022–01–15. Prieiga per internetą:
https://developer.electricimp.com/resources/uart
25. Simtools User Manual v2.4. (2019 m. Birželis). Žiūrėta 2022–01–15. Prieiga per internetą:
https://simtools.us/wp-content/uploads/2019/06/SimToolsUserManual-v2.4.pdf
26. YAW2 ARCADE Edition. Žiūrėta 2022–01–15. Prieiga per internetą:
https://shop.yawvr.com/product/yaw2-arcade-edition-2dof-without-seat-preorder/

Informacijos šaltinių sąrašas

1. Online stopwatch. Žiūrėta 2022–03–07. Prieiga per internetą: https://www.online-stopwatch.com/large-stopwatch/
2. Use iPhone as level. Žiūrėta 2022–03–07. Prieiga per internetą: https://support.apple.com/guide/iphone/use-the-level-iphbd435673d/ios
3. Decibel X, Pro Sound Meter. Žiūrėta 2022–03–07. Prieiga per internetą: https://skypaw.com/decibelx.html

Priedai

1 priedas. Prototipo valdymo algoritmas



57 pav. Prototipo komunikacijos bei ašių valdymo algoritmas

2 priedas. Arduino programos kodas

```
//#define MODE1
#define MODE2                   // nustatomas veikimo režimas
//#define MODE3
#include <digitalWriteFast.h>  // įtraukiama biblioteka skirta antrajam režimui
#define pinNum1 9               // nustatomi kontaktai antrajam režimui
#define pinNum2 10
#define pinNum3 11
#define pinDir1 3
#define pinDir2 5
#define pinDir3 6
const int   kAxisCount                = 3;   //nustatomas ašių kiekis
const char  kEOL                      = '~'; //duomenų atskyrimo simbolis
const char  kAxisName[kAxisCount]     = {'A','B','C'}; //SimTools ašių simboliai
//nustatomi prototipo fizinio judesio ribų masteliai
const int   kAxisScale[kAxisCount][3] = { {0,13000},{0,13000},{0,13000} };
const int   kMaxCharCount             = 3;   //ašies pozicijos simbolių kiekis
//nustatomi kontaktai pirmajam režimui
const int   STEP[]                    = {9,10,11};//žingsniavimo kontaktai
const int   DIR[]                     = {3,5,6};  //sukimosi krypties kontaktai
//inicializuojamos nulinės reikšmės pozicijos sekimo masyvams
int         Pos[]                     = {0,0,0};  //prototipo pradinės reikšmės
int         axisPosition[kAxisCount]  = {0,0,0};  //SimTools pradinės reikšmės
//inicializuojama nuskaitytų SimTools komunikacijos duomenų pradinė reikšmė
int         valueCharCount            = 0;
int         pulseWidth                = 10;       //digitalWrite impulso ploti
int         currentAxis;
enum TPortState { psReadAxis, psReadValue }; //komunikacijos apsikeitimo būsena
TPortState currentState = psReadAxis;        //įrašoma ašies pavadinimo
nuskaitymo būsena

void setup()          //ši funkcija iškviečiama po kiekvieno įjungimo vieną kartą
{
    Serial.begin(500000);    //nustatomas komunikacijos bitų apsikeitimo greitis
        #ifdef MODE2         //apibrėžiamos antrojo režimo operacijos
      pinModeFast(pinNum1,OUTPUT); //nustatomas kontakto išėjimo signalo režimas
      pinModeFast(pinDir1,OUTPUT);
      pinModeFast(pinNum2,OUTPUT);
      pinModeFast(pinDir2,OUTPUT);
      pinModeFast(pinNum3,OUTPUT);
      pinModeFast(pinDir3,OUTPUT);
        #endif
        #ifdef MODE1    //apibrėžiamos pirmojo režimo operacijos
        #ifdef MODE3    //apibrėžiamos trečiojo režimo operacijos
      for (int i=0; i < kAxisCount; i++) //ciklas kartojamas tiek, kiek variklių
      {
        pinMode(STEP[i],  OUTPUT); //nustatomas kontakto išėjimo signalo režimas
        pinMode(DIR[i],   OUTPUT);
      }
        #endif
        #endif
}
//skaičiuojamas reikalingas žingsnių kiekis pasiekti pozicijai
int Calc(int TargetPosition, int CurrentPosition)
{
    static int Error=0;
    Error = abs(TargetPosition - CurrentPosition);
    return Error;
}
```

```
    #ifdef MODE2                 //apibrėžiamos antrojo režimo operacijos
void FastPulse(int stepPin)     //funkcija skirta fastDigitalWrite žingsniavimui
{
  digitalWriteFast(stepPin, HIGH);//nustatoma žingsniavimo kontakto aukšta vertė
  digitalWriteFast(stepPin, LOW); //nustatoma žingsniavimo kontakto žema vertė
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
  digitalWriteFast(stepPin, HIGH);
  digitalWriteFast(stepPin, LOW);
}
    #endif

    #ifdef MODE3                  //apibrėžiamos trečiojo režimo operacijos
    #ifdef MODE1                  //apibrėžiamos pirmojo režimo operacijos
void monoPulse(int stepPin)       //funkcija skirta digitalWrite žingsniavimui
{
  digitalWrite(stepPin, HIGH);    //nustatoma žingsniavimo kontakto aukšta vertė
  delayMicroseconds(pulseWidth); //nustatyto impulso aukštos vertės užlaikymas
  digitalWrite(stepPin, LOW);     //nustatoma žingsniavimo kontakto žema vertė
  delayMicroseconds(pulseWidth); //nustatyto impulso žemos vertės užlaikymas
}
    #endif
    #endif
void loop()
{
    #ifdef MODE3            //apibrėžiamos trečiojo režimo operacijos
  monoPulse(STEP[0]);       //variklių maksimalaus greičio testas
  monoPulse(STEP[1]);       //kiekvienas papildomas funkciją prailgina ciklo laiką
  monoPulse(STEP[2]);       //veikiant visiems varikliams kartu, sulėtėja greitis
    #endif

    #ifdef MODE1                          //apibrėžiamos pirmojo režimo operacijos
  for (int i = 0; i < kAxisCount; i++)   //ciklas kartojamas tiek, kiek variklių
  {//tikrinamas skirtumas tarp prototipo ir Simtools pozicijos matricos reikšmės
    if (Calc(axisPosition[i],Pos[i]) !=0)
    {     //jeigu skirtumas nelygus nuliui, tikrinamos pozicijos reikšmės
      if (Pos[i] > axisPosition[i])
      {  //jeigu prototipo pozicija didesnė, tada nustatoma taip:
        digitalWrite(DIR[i], LOW);       //priešinga sukimosi kryptis
        Pos[i] = Pos[i] - 1;   //iš prototipo pozicijos atimama vieneto reikšmė
      }
      Else  //jeigu prototipo pozicija mažesnė, tada nustatoma taip:
      {
        digitalWrite(DIR[i], HIGH);      //pagrindinė sukimosi kryptis
        Pos[i] = Pos[i] + 1;//prie prototipo pozicijos pridedama vieneto reikšmė
      }
      monoPulse(STEP[i]);   //atliekame vieną impulso periodą pirmuoju metodu
    }
  }
    #endif
```

```
    #ifdef MODE2                          //apibrėžiamos antrojo režimo operacijos
//tikrinamas skirtumas tarp prototipo ir Simtools pozicijos pirmajam varikliui
    if (Calc(axisPosition[0],Pos[0]) !=0)
    {
      if (Pos[0] > axisPosition[0])
      {
        digitalWriteFast(pinDir1, LOW);//priešinga sukimosi kryptis „fast"
        Pos[0] = Pos[0] - 1;
      }
      else
      {
        digitalWriteFast(pinDir1, HIGH); //pagrindinė sukimosi kryptis „fast"
        Pos[0] = Pos[0] + 1;
      }
      FastPulse(pinNum1); //generuoja impulso periodą „fastDigitalWrite" metodu
    }
//tikrinamas skirtumas tarp prototipo ir Simtools pozicijos antrajam varikliui
    if (Calc(axisPosition[1],Pos[1]) !=0)
    {
      if (Pos[1] > axisPosition[1])
      {
        digitalWriteFast(pinDir2, LOW);
        Pos[1] = Pos[1] - 1;
      }
      else
      {
        digitalWriteFast(pinDir2, HIGH);
        Pos[1] = Pos[1] + 1;
      }
      FastPulse(pinNum2);
    }
    if (Calc(axisPosition[2],Pos[2]) !=0)
//tikrinamas skirtumas tarp prototipo ir Simtools pozicijos trečiajam varikliui
    {
      if (Pos[2] > axisPosition[2])
      {
        digitalWriteFast(pinDir3, LOW);
        Pos[2] = Pos[2] - 1;
      }
      else
      {
        digitalWriteFast(pinDir3, HIGH);
        Pos[2] = Pos[2] + 1;
      }
      FastPulse(pinNum3);
    }
      #endif
}
```

```
void serialEvent() //ši funkcija iškviečiama, kai komunikacija perduoda duomenis
{
    char  tmpChar; //deklaruojamas kintamasis skirtas ašies simbolių duomenims
    int   tmpValue;//deklaruojamas kintamasis skirtas ašies pozicijos duomenims
    while (Serial.available()) //kai komunikacijoje yra sukauptas duomenų baitas
    {     //tikrinama ar dabartinė būsena ašies pavadinimo nuskaitymo būsena
      if (currentState == psReadAxis)
      {
        tmpChar = Serial.read();    //nuskaitomi ašies pavadinimo duomenys
         //ciklas kartojamas tiek, kiek variklių arba kol randamas atitikmuo
        for (int i = 0; i < kAxisCount; i++)
        { // jeigu nuskaitomas simbolis lygus ašies pavadinimo simboliui
          if (tmpChar == kAxisName[i])
          {
            currentAxis = i;        //sutapusi ašis įsimenama
          //įrašoma ašies pozicijos duomenų nuskaitymo būsena
            currentState = psReadValue;
          //inicializuojama pradinė nulinė reikšmė komunikacijos nuskaitymui
            axisPosition[currentAxis] = 0;
          //inicializuojama pradinė nulinė reikšmė nuskaitytų simbolių skaičių
            valueCharCount = 0;
            break;                  //nutraukiamas ciklas
          }
        }
       }
       //tikrinama ar dabartinė būsena ašies pozicijos duomenų nuskaitymo būsena
       if (currentState == psReadValue)
       {//vykdomas ciklas kol nuskaitytų duomenų baitų kiekis pasiekią maksimalų
        while ((valueCharCount < kMaxCharCount) && Serial.available())
        {
          tmpValue = Serial.read(); //nuskaitomi ašies pozicijos duomenys
            //tikrinama ar nuskaitomas duomenų baitas yra atskiriamasis simbolis
          if (tmpValue != kEOL)
          { //konvertuojama iš ASCII į dešimtainę sistemą
            tmpValue = tmpValue - 48;
            //tikrinama ar konvertuota reikšmė yra tarp 0 ir 9 skaičių reikšmių
            if ((tmpValue < 0) || (tmpValue > 9))
            { tmpValue = 0; }       //jeigu neatitinka tada prilyginama nuliui
            //nuskaityta baito skaitinė reikšmė įrašoma į pozicijos masyvą
            axisPosition[currentAxis] = axisPosition[currentAxis] * 10
            + tmpValue;
            valueCharCount++;       //skaičiuojama kiek įrašyta skaitmenų
          }
          else break;
        }   //tikrinama ar nuskaitytas simbolis yra atskiriamasis simbolis
            //arba pasiektas maksimalus trijų simbolių skaičius
        if (tmpValue == kEOL || valueCharCount == kMaxCharCount)
        {   //ašies pozicijos duomenų mastelis keičiamas į fizinio prototipo
            //judesių ribų mastelį
          axisPosition[currentAxis] = map(axisPosition[currentAxis], 0, 255,
          kAxisScale[currentAxis][0], kAxisScale[currentAxis][1]);
            //inicializuojama ašies pavadinimo nuskaitymo būsena
          currentState = psReadAxis;
        }
       }
    }
}
```

3 priedas. Prototipo garso lygio ir posvyrių tyrimų duomenys

3 lentelė. Prototipo garso lygio tyrimo duomenys

Mikrožingsnių skaičius	**Vidurkio *RMS, dBA***	**Maksimumo *RMS, dBA***	**Piko *RMS, dBA***
800	77.1	81.7	85.9
1000	77.1	82.1	86.3
1600	77.0	81.7	85.5
2000	77.0	82.0	86.6
3200	74.2	80.7	85.5
3600	73.9	79.5	83.1
4000	72.6	78.9	83.7
5000	70.5	77.1	82.6
6400	68.5	76.3	80.9
7200	68.2	77.2	80.7
8000	66.3	71.1	77.0
10000	65.1	68.9	73.8
12800	63.5	68.3	72.4
20000	59.9	64.4	69.7
25600	58.0	62.1	68.0
40000	55.7	60.3	66.6

4 lentelė. Prototipo įvarių posvyrių tyrimo duomenys

Platformos posvyris laipsniais pagal platformos poziciją ir veikiamas jėgas	***Roll* 0% & *Pitch* 0%**	***Roll* -100%**	***Roll* +100%**	***Pitch* -100%**	***Pitch* +100%**	***Roll* -100% & *Pitch* -100%**	***Roll* +100% & *Pitch* +100%**	***Roll* +100% & *Pitch* -100%**	***Roll* -100% & *Pitch* +100%**
Nusileidusi	0	17	17	18	17	31	30	31	1
Pakilusi	0	13	13	13	13	28	28	0	29
Vidurinė	0	29	28	27	29	29	28	14	16

4 priedas. Teoriniai modelio greitaveikos tyrimo duomenys

5 lentelė. Teoriniai platformos judesio iš vidurinės į viršutinę padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	Žingsnio tikslumas, °/žingsniui	Nukeliauta žingsnių	Judėjimo trukmė, s	Greitis, mm/s
800	0.4500000	140	0.0003500	117143
1000	0.3600000	174	0.0004350	94253
1600	0.2250000	279	0.0006975	58781
2000	0.1800000	349	0.0008725	46991
3200	0.1125000	558	0.0013950	29391
3600	0.1000000	628	0.0015700	26115
4000	0.0900000	698	0.0017450	23496
5000	0.0720000	872	0.0021800	18807
6400	0.0562500	1116	0.0027900	14695
7200	0.0500000	1256	0.0031400	13057
8000	0.0450000	1396	0.0034900	11748
10000	0.0360000	1744	0.0043600	9404
12800	0.0281250	2233	0.0055825	7344
20000	0.0180000	3489	0.0087225	4700
25600	0.0140625	4466	0.0111650	3672
40000	0.0090000	6978	0.0174450	2350

6 lentelė. Teoriniai platformos judesio iš vidurinės į viršutinę padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	Žingsnio tikslumas, °/žingsniui	Nukeliauta žingsnių	Judėjimo trukmė, s	Greitis, mm/s
800	0.4500000	140	0.00280	14643
1000	0.3600000	174	0.00348	11782
1600	0.2250000	279	0.00558	7348
2000	0.1800000	349	0.00698	5874
3200	0.1125000	558	0.01116	3674
3600	0.1000000	628	0.01256	3264
4000	0.0900000	698	0.01396	2937
5000	0.0720000	872	0.01744	2351
6400	0.0562500	1116	0.02232	1837
7200	0.0500000	1256	0.02512	1632
8000	0.0450000	1396	0.02792	1468
10000	0.0360000	1744	0.03488	1175
12800	0.0281250	2233	0.04466	918
20000	0.0180000	3489	0.06978	588
25600	0.0140625	4466	0.08932	459
40000	0.0090000	6978	0.13956	294

5 priedas. Prototipo greitaveikos fastDigitalWrite metodu tyrimo duomenys

7 lentelė. Prototipo judesio iš apačios į 100% *Pitch* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
20000	0.032	0.031	0.032	0.031	0.032	555
25600	0.066	0.068	0.065	0.066	0.066	264
40000	0.193	0.192	0.199	0.2	0.196	89

8 lentelė. Prototipo judesio iš apačios į 100% *Roll* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
20000	0.033	0.034	0.034	0.033	0.034	507
25600	0.097	0.099	0.098	0.1	0.099	173
40000	0.196	0.229	0.195	0.207	0.207	82

9 lentelė. Prototipo judesio iš vidurio į 100% *Pitch* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
20000	0.032	0.033	0.033	0.034	0.033	848
25600	0.102	0.099	0.101	0.098	0.1	280
40000	0.224	0.232	0.195	0.22	0.218	128

10 lentelė. Prototipo judesio iš vidurio į 100% *Roll* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
20000	0.034	0.033	0.032	0.033	0.033	863
25600	0.098	0.1	0.099	0.097	0.099	289
40000	0.195	0.194	0.228	0.229	0.212	134

11 lentelė. Prototipo judesio iš vidurio į 100% *Heave* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, mm/s**
20000	0.034	0.034	0.032	0.033	0.033	1233
25600	0.099	0.1	0.097	0.098	0.099	416
40000	0.191	0.225	0.241	0.226	0.222	185

12 lentelė. Prototipo judesio iš viršaus į 100% *Pitch* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
20000	0.033	0.033	0.034	0.033	0.033	391
25600	0.098	0.099	0.1	0.101	0.1	131
40000	0.228	0.197	0.227	0.196	0.213	61

13 lentelė. Prototipo judesio iš viršaus į 100% *Roll* padėtį **fastDigitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
20000	0.031	0.031	0.032	0.031	0.031	416
25600	0.066	0.065	0.067	0.069	0.067	195
40000	0.196	0.195	0.196	0.201	0.197	66

6 priedas. Prototipo greitaveikos digitalWrite metodu tyrimo duomenys

14 lentelė. Prototipo judesio iš apačios į 100% *Pitch* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
800	0.031	0.031	0.032	0.031	0.031	560
1000	0.041	0.045	0.039	0.038	0.041	429
1600	0.055	0.058	0.051	0.06	0.056	312
2000	0.065	0.069	0.07	0.067	0.068	258
3200	0.081	0.091	0.089	0.092	0.088	198
3600	0.099	0.101	0.105	0.098	0.101	174
4000	0.11	0.115	0.109	0.116	0.113	155
5000	0.125	0.121	0.128	0.124	0.125	141
6400	0.135	0.133	0.131	0.136	0.134	131
7200	0.145	0.151	0.159	0.149	0.151	116
8000	0.168	0.17	0.182	0.18	0.175	100
10000	0.196	0.201	0.195	0.185	0.194	90
12800	0.23	0.256	0.22	0.225	0.233	75
20000	0.311	0.302	0.299	0.296	0.302	58
25600	0.421	0.428	0.394	0.396	0.41	43
40000	0.557	0.554	0.556	0.589	0.564	31

15 lentelė. Prototipo judesio iš apačios į 100% *Roll* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
800	0.059	0.061	0.066	0.058	0.061	278
1000	0.069	0.071	0.07	0.068	0.07	245
1600	0.079	0.075	0.082	0.08	0.079	215
2000	0.085	0.089	0.091	0.087	0.088	193
3200	0.106	0.111	0.102	0.105	0.106	160
3600	0.128	0.13	0.131	0.129	0.13	131
4000	0.14	0.145	0.135	0.142	0.141	121
5000	0.155	0.159	0.154	0.16	0.157	108
6400	0.17	0.166	0.165	0.169	0.168	101
7200	0.191	0.185	0.181	0.179	0.184	92
8000	0.197	0.22	0.196	0.198	0.203	84
10000	0.261	0.263	0.221	0.215	0.241	71
12800	0.329	0.315	0.295	0.299	0.31	55
20000	0.401	0.393	0.395	0.399	0.397	43
25600	0.525	0.521	0.492	0.495	0.508	33
40000	0.718	0.688	0.723	0.692	0.705	24

16 lentelė. Prototipo judesio iš vidurio į 100% *Pitch* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
800	0.059	0.061	0.055	0.06	0.059	476
1000	0.074	0.071	0.072	0.075	0.073	383
1600	0.087	0.092	0.095	0.084	0.09	312
2000	0.111	0.099	0.115	0.11	0.109	257
3200	0.121	0.12	0.125	0.119	0.121	231
3600	0.131	0.129	0.134	0.128	0.131	215
4000	0.135	0.141	0.137	0.14	0.138	203
5000	0.155	0.159	0.152	0.16	0.157	179
6400	0.165	0.169	0.171	0.166	0.168	167
7200	0.185	0.191	0.189	0.179	0.186	150
8000	0.196	0.197	0.211	0.201	0.201	139
10000	0.23	0.26	0.229	0.245	0.241	116
12800	0.329	0.312	0.299	0.321	0.315	89
20000	0.426	0.484	0.428	0.431	0.443	63
25600	0.556	0.557	0.561	0.524	0.55	51
40000	0.853	0.788	0.829	0.754	0.807	35

17 lentelė. Prototipo judesio iš vidurio į 100% *Roll* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
800	0.049	0.056	0.06	0.05	0.054	528
1000	0.075	0.071	0.069	0.065	0.07	407
1600	0.085	0.081	0.079	0.08	0.081	351
2000	0.109	0.099	0.095	0.091	0.099	289
3200	0.118	0.122	0.12	0.123	0.121	236
3600	0.129	0.133	0.13	0.128	0.13	219
4000	0.137	0.142	0.144	0.139	0.141	203
5000	0.155	0.16	0.159	0.153	0.157	182
6400	0.169	0.166	0.165	0.17	0.168	170
7200	0.188	0.185	0.19	0.181	0.186	153
8000	0.197	0.196	0.199	0.2	0.198	144
10000	0.23	0.241	0.256	0.225	0.238	120
12800	0.298	0.315	0.31	0.325	0.312	91
20000	0.401	0.459	0.462	0.423	0.437	65
25600	0.524	0.523	0.556	0.59	0.549	52
40000	0.751	0.818	0.757	0.783	0.778	37

18 lentelė. Prototipo judesio iš vidurio į 100% *Heave* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, mm/s**
800	0.052	0.056	0.069	0.05	0.057	716
1000	0.072	0.071	0.082	0.08	0.076	537
1600	0.088	0.085	0.089	0.086	0.087	471
2000	0.111	0.091	0.093	0.1	0.099	414
3200	0.125	0.12	0.118	0.121	0.121	339
3600	0.131	0.135	0.129	0.128	0.131	314
4000	0.147	0.139	0.146	0.136	0.142	289
5000	0.155	0.16	0.159	0.153	0.157	262
6400	0.179	0.18	0.163	0.17	0.173	237
7200	0.188	0.181	0.19	0.185	0.186	220
8000	0.211	0.194	0.19	0.2	0.199	206

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, mm/s**
10000	0.235	0.239	0.257	0.243	0.244	168
12800	0.299	0.32	0.319	0.315	0.313	131
20000	0.42	0.434	0.452	0.449	0.439	93
25600	0.536	0.546	0.521	0.585	0.548	75
40000	0.735	0.799	0.813	0.746	0.774	53

19 lentelė. Prototipo judesio iš viršaus į 100% *Pitch* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
800	0.055	0.051	0.061	0.063	0.058	225
1000	0.071	0.07	0.069	0.075	0.071	182
1600	0.081	0.083	0.079	0.08	0.081	161
2000	0.095	0.09	0.099	0.1	0.096	135
3200	0.116	0.121	0.11	0.119	0.117	112
3600	0.13	0.125	0.129	0.131	0.129	101
4000	0.135	0.132	0.14	0.141	0.137	95
5000	0.152	0.16	0.158	0.154	0.156	83
6400	0.164	0.166	0.161	0.169	0.165	79
7200	0.174	0.17	0.169	0.175	0.172	76
8000	0.197	0.196	0.185	0.181	0.19	68
10000	0.215	0.23	0.205	0.215	0.216	60
12800	0.294	0.301	0.289	0.295	0.295	44
20000	0.393	0.395	0.394	0.4	0.396	33
25600	0.452	0.472	0.523	0.46	0.478	27
40000	0.721	0.685	0.694	0.726	0.707	18

20 lentelė. Prototipo judesio iš viršaus į 100% *Roll* padėtį **digitalWrite** metodu tyrimo duomenys

Mikrožingsnių skaičius	**Judėjimo trukmė Nr.1, s**	**Judėjimo trukmė Nr.2, s**	**Judėjimo trukmė Nr.3, s**	**Judėjimo trukmė Nr.4, s**	**Judėjimo trukmės *RMS, s***	**Greitis, °/s**
800	0.031	0.032	0.032	0.031	0.032	413
1000	0.042	0.05	0.049	0.052	0.048	269
1600	0.055	0.06	0.063	0.059	0.059	219
2000	0.071	0.075	0.069	0.07	0.071	182
3200	0.09	0.085	0.089	0.081	0.086	151
3600	0.098	0.099	0.102	0.105	0.101	129
4000	0.11	0.109	0.115	0.112	0.112	117
5000	0.121	0.125	0.129	0.13	0.126	103
6400	0.14	0.139	0.135	0.131	0.136	95
7200	0.155	0.152	0.16	0.162	0.157	83
8000	0.163	0.166	0.17	0.169	0.167	78
10000	0.174	0.183	0.175	0.185	0.179	72
12800	0.198	0.201	0.23	0.215	0.211	62
20000	0.285	0.296	0.279	0.302	0.291	45
25600	0.393	0.459	0.401	0.428	0.421	31
40000	0.624	0.592	0.589	0.593	0.6	22