KAUNO TECHNOLOGIJOS UNIVERSITETAS
PANEVĖŽIO TECHNOLOGIJŲ IR VERSLO FAKULTETAS

Dovydas Karosas

LAZERIO PLUOŠTELIO VALDYMO TRIMATĖJE ERDVĖJE TYRIMAS

Baigiamasis magistro projektas

Vadovas
Prof. dr. Darius Viržonis

PANEVĖŽYS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
PANEVĖŽIO TECHNOLOGIJŲ IR VERSLO FAKULTETAS

LAZERIO PLUOŠTELIO VALDYMO TRIMATĖJE ERDVĖJE TYRIMAS

Baigiamasis magistro projektas
Valdymo technologijos (kodas 621H66001)

Vadovas
Prof. dr. Darius Viržonis
2017-06-01

Recenzentas

Projektą atliko
Dovydas Karosas
2017-06-01

PANEVĖŽYS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Panevėžio technologijų ir verslo fakultetas
(Fakultetas)

Dovydas Karosas
(Studento vardas, pavardė)

Valdymo technologijos, 621H66001
(Studijų programos pavadinimas, kodas)

„Lazerio pluoštelio valdymo trimatėje erdvėje tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2017 m. birželio 1 d.
Panevėžys

Patvirtinu, kad mano, **Dovydo Karoso**, baigiamasis projektas tema „Lazerio pluoštelio valdymo trimatėje erdvėje tyrimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	*(parašas)*

TVIRTINU:
KTU Panevėžio technologijų ir verslo fakulteto
Technologijų katedros vedėjas

_____

20….. ....

BAIGIAMOJO PROJEKTO UŽDUOTIS

Išduota studentui: *Dovydui Karosui* Grupė *PME-5*

1. Darbo tema:

Lietuvių kalba: *Lazerio pluoštelio valdymo trimatėje erdvėje tyrimas*

Anglų kalba: *Research of Laser Beam Control in Three-Dimensional Space*

Patvirtinta 2017 m. kovo mėn. 30 d. dekano potvarkiu Nr.V25-13-8.

2. Darbo tikslas: *Sukurti programinę įrangą dviašiam lazerio pozicionavimui ir atlikti sistemos tyrimus.*

3. Reikalavimai ir sąlygos: *Baigiamajame magistro projekte turi būti atlikti du sistemos tyrimai: statikos ir dinamikos. Pagal gautus tyrimo rezultatus turi būti sukurtas sistemos kalibravimo algoritmas.*

4. Projekto struktūra. *Turinys konkretizuojamas kartu su vadovu, atsižvelgiant į BP pobūdį.*

1. *Analitinė dalis*
   - *Prototipinės sistemos konstrukcijos analizė*
   - *Naudojamos įrangos apžvalga*
   - *Sektorinio pozicionavimo sistemų valdymo algoritmų analizė*
2. *Projektinė dalis*
   - *Sistemos valdymo kinematinė dalis*
   - *Valdymo algoritmo kūrimas*
   - *Valdymo programos kūrimas*
   - *Statikos tyrimas*
   - *Gautų rezultatų įvertinimas ir išvados*
   - *Dinamikos tyrimas*
   - *Gautų rezultatų įvertinimas ir išvados*
   - *Sistemos kalibravimo sprendimo planas*
   - *Kalibravimo algoritmo kūrimas*

5. Ši užduotis yra neatskiriama baigiamojo projekto dalis.

6. Projekto pateikimo gynimui kvalifikacinėje komisijoje terminas *2017-06-01* *(data)*

Užduotį gavau: _____ *2017-02-01*
(studento vardas, pavardė, parašas) *(data)*

Vadovas: _____ *2017-02-01*
(pareigos, vardas, pavardė, parašas) *(data)*

Karosas, Dovydas. Lazerio pluoštelio valdymo trimatėje erdvėje tyrimas. Magistro baigiamasis projektas / vadovas prof. dr. Darius Viržonis; Kauno technologijos universitetas, Panevėžio technologijų ir verslo fakultetas.

Mokslo kryptis ir sritis: Valdymo technologijos.

Reikšminiai žodžiai: *lazeris, pozicionavimas, statika, dinamika, kalibravimas.*

Panevėžys, 2017. 47 p.

SANTRAUKA

Dažniausiai naudojamos pozicionavimo sistemos sudarytos iš vienos ar kelių pavarų, įvairių kreipiančiųjų ir kitų judesio perdavimo mechanizmų. Visa tai leidžia atlikti ortogonalų objekto pozicionavimą *X*, *Y* ir *Z* ašyse. Siekiant ekonomiškumo naudojamos sektorinio pozicionavimo sistemos, kurioms užtenka tik pavarų ir nereikia papildomų mechanizmų. Tačiau tokios sistemos reikalauja sudėtingesnio valdymo, o taip pat mažėja pozicionavimo tikslumo potencialas. Todėl šiame darbe siekiama ištirti lazerio pluoštelio sektorinio pozicionavimo sistemos valdymą ir nustatyti šios sistemos charakteristikas.

Šio darbo tikslas - sukurti programinę įrangą dviašiam lazerio pozicionavimui ir atlikti sistemos tyrimus.

Darbo metodai: mokslinės literatūros, internetinių išteklių analizė, darbas programomis *AutoCAD*, *MatLAB*, *Arduino IDE*, *ImageJ*, *Gimp.*

Šiame darbe išnagrinėta prototipinė sistemos konstrukcija – nustatyta naudojama įranga, išanalizuoti sistemos valdymo principai. Toliau nubraižyta kinematinė sistemos schema, pagal kurią rastos lazerio pozicionavimui reikalingos lygtys. Remiantis gautomis lygtimis sukurtas sistemos valdymo algoritmas ir parašyta valdymo programa. Su veikiančia sistema atliktas sistemos statikos tyrimas, kurio metu gautos statinės ir sisteminės paklaidos. Čia apskaičiuoti statistiniai rodikliai. Darbo metu taip pat atliktas dinamikos tyrimas, kurio metu gauti svyravimai apie koordinatę. Čia pateiktos galimos dinaminių paklaidų priežastys, jų eliminavimo būdai, nustatyta sistemos greitaveika. Atsižvelgiant į statikos tyrimo metu gautus sisteminius nuokrypius sukurtas programinio kalibravimo algoritmas – nubraižytos skaičiavimų schemos, pagal kurias gautos ekrano pasvirimo kampo ir koordinatės korekcijos lygtys.

Karosas, Dovydas. *Research of Laser Beam Control in Three-Dimensional Space*: *Master's* thesis in Control Technologies / supervisor assoc. prof. Darius Viržonis. Panevėžys Faculty of Technologies and Business, Kaunas University of Technology.

Research area and field: Control Technologies.

Key words: *laser, positioning, statics, dynamics, calibration.*

Panevėžys, 2017. 47 p.

SUMMARY

The positioning systems usually consist of one or more drives, various guides and other motion transfer mechanisms. All of these elements allow to orthogonally position the object in X, Y and Z axes. To achieve better economy the sectorial positioning is used instead. This type of systems only need drives without any other mechanisms. However, these systems require more complicated control and the potential of accuracy is also reduced. Therefore, the aim of this thesis is to research the control of sectorial laser beam positioning system and identify characteristics of this system.

The objective of this thesis is to develop two-axes laser positioning system software and carry out the research of the system.

Research methods: analysis of scientific literature and online resources, work with programs AutoCAD, MatLAB, Arduino IDE, ImageJ, Gimp.

The prototype construction of system was analyzed in this thesis – used equipment was identified and principles of controlling was researched. Furthermore, a kinematics scheme was drawn, which allowed to find equations for laser positioning. According to these equations an algorithm and software of system control were developed. Using the developed software, a statics research was carried out. Statics research shown static and systematic errors in system. Therefore, statistical parameters were calculated in this stage. A dynamics research was also carried out in this thesis. This research shown fluctuations around coordinate. The possible causes of dynamic errors and their elimination methods were given and speed of system was identified. Considering systematic errors that were obtained from statics research an algorithm of calibration was developed. In this stage calculation schemes were drawn and equations for coordinate correction were found.

TURINYS

ĮVADAS

Pozicionavimo sistemos itin dažnai naudojamos tiek gamyboje, tiek robotuose, tiek įvairiausiuose mechanizmuose. Lazerio pluoštelio pozicionavimas reikalingas atlikti įvairiems su lazeriu susijusiems darbams: pjaustymui, deginimui, atvaizdavimui. Dažniausiai naudojamos pozicionavimo sistemos sudarytos iš vienos ar kelių pavarų, įvairių kreipiančiųjų ir kitų judesio perdavimo mechanizmų. Visa tai leidžia atlikti ortogonalų objekto pozicionavimą *X*, *Y* ir *Z* ašyse. Siekiant ekonomiškumo naudojamos sektorinio pozicionavimo sistemos, kurioms užtenka tik pavarų ir nereikia papildomų mechanizmų. Tačiau tokios sistemos reikalauja sudėtingesnio valdymo, o taip pat mažėja pozicionavimo tikslumo potencialas. Todėl šiame darbe siekiama ištirti lazerio pluoštelio sektorinio pozicionavimo sistemos valdymą ir nustatyti šios sistemos charakteristikas.

Tyrimo objektas – lazerio pluoštelio valdymo trimatėje erdvėje sistema.

Darbo tikslas – sukurti programinę įrangą dviašiam lazerio pozicionavimui ir atlikti sistemos tyrimus.

Darbo uždaviniai

1. Išanalizuoti prototipinę sistemos konstrukciją ir naudojamą įrangą;
2. Išanalizuoti panašaus tipo algoritmus;
3. Pateikti sistemos valdymo sprendimo planą;
4. Nubraižyti kinematinę sistemos schemą ir atlikti reikiamus skaičiavimus;
5. Sukurti sistemos valdymo algoritmą ir parašyti programą;
6. Atlikti statikos ir dinamikos tyrimus;
7. Įvertinti gautus tyrimo rezultatus;
8. Pateikti sistemos kalibravimo algoritmą.

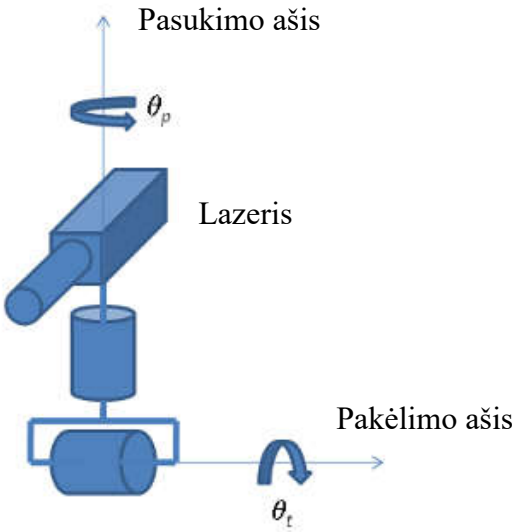
Tyrimo metodai – mokslinės literatūros, internetinių išteklių analizė, darbas programomis *AutoCAD*, *MatLAB*, *Arduino IDE*, *ImageJ*, *Gimp*.

1. ANALITINĖ DALIS

1.1. Sistemos analizė

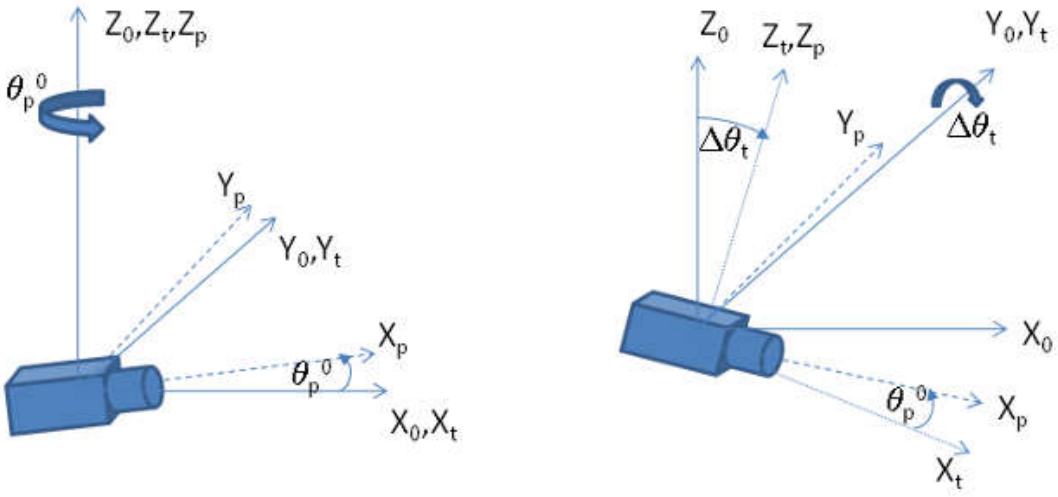
1.1.1. Prototipinė konstrukcija

Šiame darbe nagrinėjamos sistemos konstrukcija susideda iš kelių pagrindinių elementų – pavarų ir lazerio. Tokia sistema skiriasi nuo roboto rankos tuo, jog abi pavaros ir lazeris montuojami vieni ant kitų. Taip gaunama pasukimo-pakėlimo (angl. *Pan-Tilt*) sistema (1.1 pav.).



1.1 pav. Pan-Tilt sistemos prototipinė schema

Taip įtvirtinus sistemos elementus, lazerį galima pasukti į šonus, pakelti į viršų arba nuleisti žemyn. Visi lazerio posūkio kampai skaičiuojami iš vieno atskaitos taško (1.2 pav.).

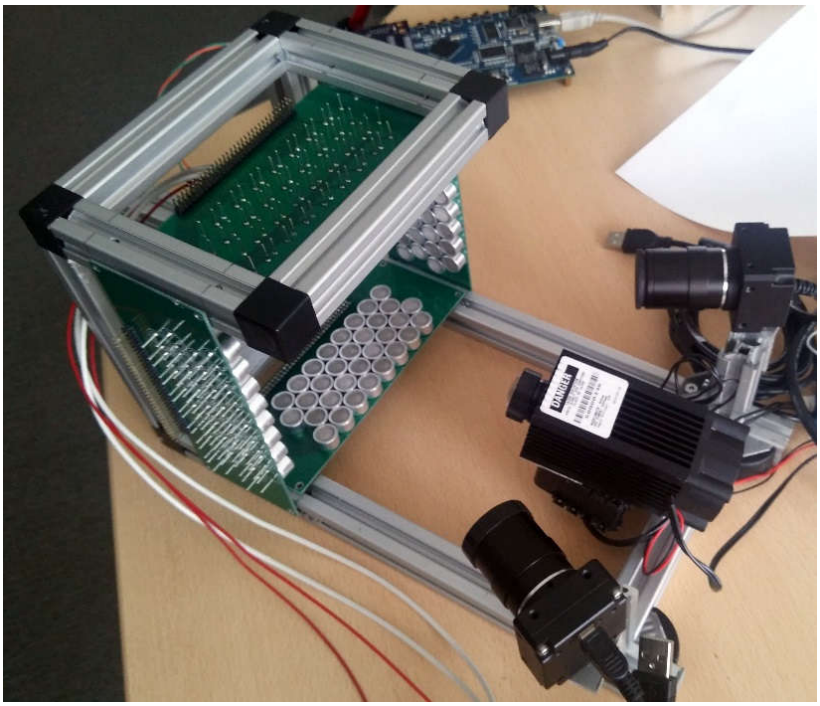


1.2 pav. *Pan-Tilt* sistemos posūkio kampai

Tokiu būdu sumontuotos sistemos dažnai naudojamos valdomose vaizdo kamerose arba robotuose jų matymui, kai dvi kameros imituoja žmogaus akis (1.3 pav.). Panašių lazerio pluoštelio pozicionavimo sistemų galima rasti ir internetiniuose šaltiniuose, tačiau tokiuose pavyzdžiuose neaptariamas lazerio pozicionavimas koordinačių sistemoje, o tik pavarų valdymas.

1.3 pav. Prototipinių konstrukcijų pavyzdžiai [1, 2]

Šiame darbe tiriamos sistemos konstrukcija susideda iš metalinio rėmo, pavarų sistemos ir lazerio. Rėmas daromas iš automatikoje plačiai naudojamų aliuminio profilių, kurie sujungti į 1.4 pav. pavaizduotą konstrukciją. Viename jos gale sumontuota pavarų sistema su lazeriu, o kitame- ekranas, kuriame fokusuojamas ir pozicionuojamas lazerio pluoštelis.



1.4 pav. Sistemos konstrukcija

1.1.2. Servo pavaros ir kita įranga

Sistemoje reikalingos dvi servo pavaros, kurios veiktų dideliu tikslumu, bei būtų patogiai ir paprastai valdomos. Tam naudojamos *Dynamixel AX-12A* servo pavaros. Šios pavaros skirtos didelio tikslumo sistemoms, jos sudarytos iš nuolatinės srovės variklių, turi reduktorių bei integruotą valdymo schemą. Pavarų valdymas vyksta per pusiau dvipusį (angl. *Half duplex*) nuoseklųjį ryšį, o pagrindinis tokių pavarų privalumas – galimybė į magistralę sujungti net 254 pavaras ir valdyti jas per vieną nuoseklųjį prievadą. Pagrindiniai pavaros parametrai pateikti 1 lentelėje [3].

1 lentelė

Pavaros parametrai [3]

Posūkio kampo skyra	0,35°
Darbinis kampas	300°
Įtampa	7V – 10 V
Maksimali srovė	900 mA
Darbinė temperatūra	-5 °C - +85 °C
Komunikacijų greitis	7 343 bps – 1 Mbps
Grįžtamasis ryšys	Padėtis, temperatūra, apkrovimas, įtampa ir kt.
Reduktoriaus perdavimo skaičius	1/254
Maksimalus pozicijos laikymo momentas	1,62 Nm

Norint valdyti šias pavaras reikalingas mikrovaldiklis. Šiame darbe naudojamas *Arduino* mikrovaldiklis, kurį paprasta programuoti ir sujungti su kitais prietaisais. Šis mikrovaldiklis turi nuoseklųjį prievadą, per kurį galima komunikuoti su pavaromis, tačiau *Arduino* per nuoseklųjį prievadą komunikuoja pilnai dvipusiu (angl. *Full duplex*) ryšiu, o pavaros – pusiau dvipusiu, todėl pavaroms valdyti reikalingas *Full duplex-Half duplex* komunikacijų keitiklis. Tam panaudota *AX-12 CDS55xx* plokštė [4], kuri susideda iš įtampos reguliatoriaus ir *Half duplex* schemų. Plokštė maitinama atskiru šaltiniu 7-16 V įtampa. Šaltiniuose [3] teigiama, kad *Dynamixel* pavaroms tinkamiausia įtampa – 9 V.

Dynamixel AX-12A valdymui su *Arduino* yra sukurtos bibliotekos [5], kuriomis pasinaudojus galima paprasčiau komunikuoti su pavaromis. Pirmiausia pavaroms reikia suteikti ID numerį, pagal kurį jos atpažįstamos. Kadangi pagal nutylėjimą pavaros turi ID numerį 1, antros pavaros ID būtina pakeisti į 2. ID numerio pakeitimo kodas pateiktas 1 priede.

Norint pasukti pavarą į norimą padėtį, reikia naudoti padėties užduoties įrašymo kodą (2 priedas). Užduoties įrašymas atliekamas tokiais žingsniais:

1. įrašomas pavaros identifikavimo numeris, įrašymo kiekis ir įrašymo komanda;

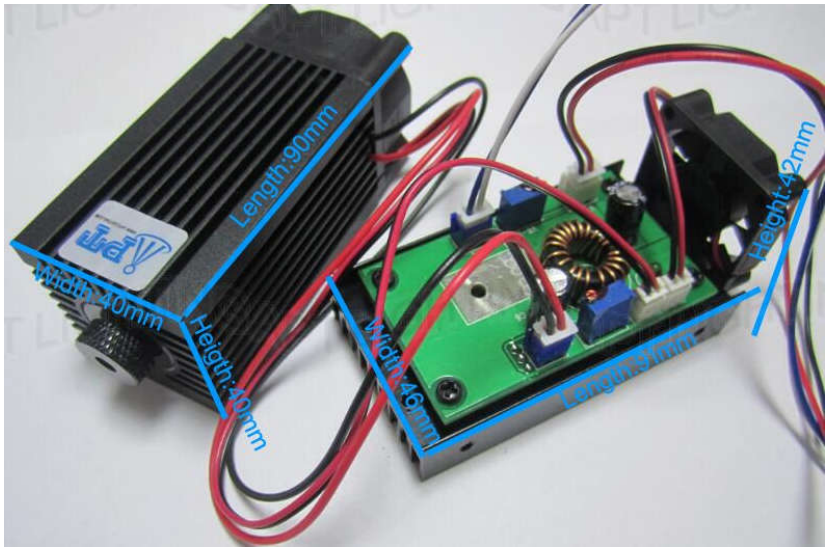
2. įrašoma užduoties komanda;
3. įrašomos padėties ir greičio vertės;
4. įrašomi patikrinimui skirti duomenys;
5. palaukiama 2 ms, kol duomenys bus apdoroti;
6. atmetami visi gauti duomenys.

Norint gauti enkoderio duomenis apie padėtį, pirmiausiai siunčiama užklausa, o po to gaunamas atsakymas iš atitinkamos pavaros (3 priedas):

1. įrašomas pavaros identifikavimo numeris, įrašymo kiekis ir nuskaitymo komanda;
2. įrašoma užduoties komanda;
3. įrašomas nuskaitomų duomenų kiekis;
4. įrašomi patikrinimui skirti duomenys;
5. palaukiama 2 ms, kol duomenys bus apdoroti;
6. nuskaitomas pirmasis baitas;
7. nuskaitomas antrasis baitas;
8. baitai sujungiami į posūkio kampo vertę.

1.1.3. Lazeris

Šiame darbe nagrinėjama sistema lazerio pluoštelio pagalba turės atlikti 3D spausdintuvo funkcijas, paremtas medžiagos sukietinimo principais. Norint tai atlikti reikalingas pakankamai galingas lazeris. Nagrinėjamoje sistemoje naudojamas 4 W galios ir 810 nm bangos ilgio lazerio modulis (1.5 pav.) [6]. Lazerio sistema susideda iš lazerio modulio ir valdymo plokštės. Modulis turi radiatorių su ventiliatoriumi aušinimui, o lazerio pluoštelis fokusuojamas stacionariu lęšiu, kuris pastatomas rankomis. Lazeris valdomas tranzistorius-tranzistorius loginiu lygiu (TTL) bei platumine impulso moduliacija (PWM). Pagrindiniai lazerio sistemos parametrai pateikti 2 lentelėje.



1.5 pav. Lazerio sistema [6]

2 lentelė

Lazerio sistemos parametrai

Bangos ilgis	805 – 810 nm
Galia	4 000 mW
Fokusavimas	<5 mrad
Įtampa	12 V
Srovė	<5 A
TTL valdymo dažnis	0 – 20 kHz

Dirbant su šiuo lazeriu būtina atkreipti dėmesį į tai, kad jis labai pavojingas žmogaus akims. Todėl būtina turėti apsauginius akinius, skirtus 810 nm ilgio šviesos bangoms.

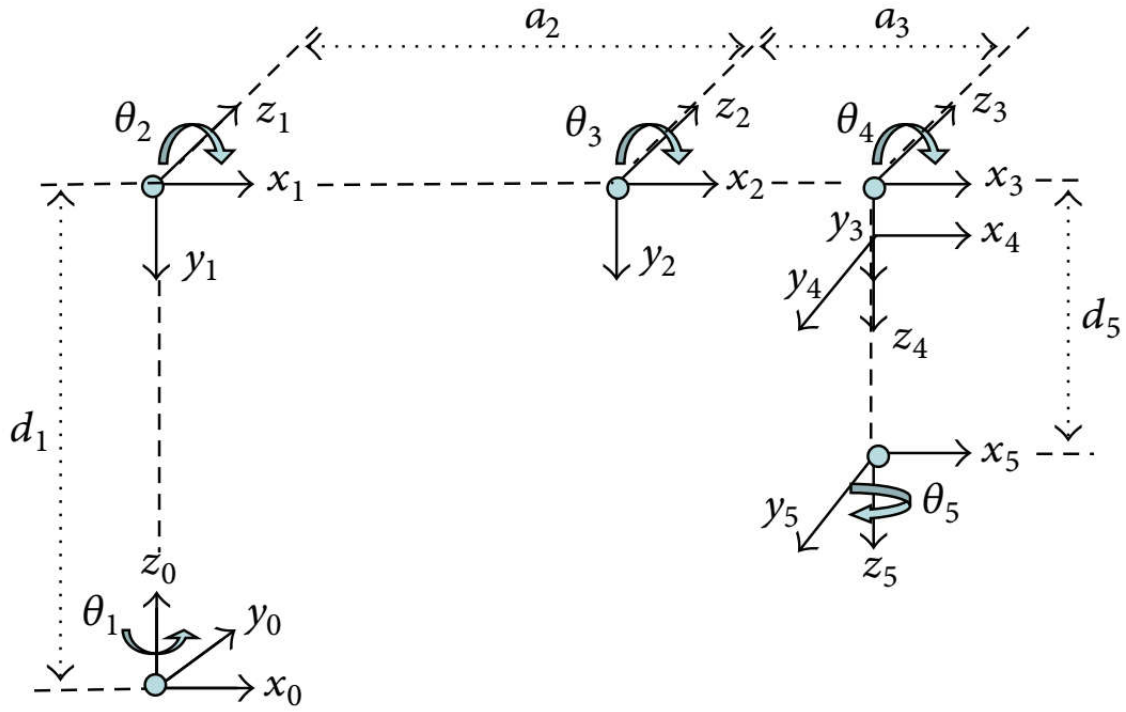
Atliekant eksperimentinius tyrimus patogu naudoti mažos galios regimojo bangos ilgio lazerinį žymeklį. Tokiu atveju nereikia apsauginių akinių, todėl paprasčiau pakeisti ar pakoreguoti tiriamą sistemą.

1.1.4. Algoritmas

Norint lazerio pluoštelį tiksliai pozicionuoti plokštumoje X ir Y koordinatėmis, reikia tinkamai perskaičiuoti koordinates į servo pavarų posūkių kampus. Kaip ir robotikoje, šiame darbe analizuojamoje sistemoje servo pavaros mechaniškai yra tiesiogiai priklausomos ir sudaro vientisą sistemą. Robotikoje naudojamos tiesioginės ir atvirkštinės kinematikos sąvokos. Tiesioginės kinematikos atveju reikia rasti roboto rankos, arba šiuo atveju – lazerio taško, padėtį, kai žinomi pavarų pasisukimo kampai bei atstumai tarp jungčių. Atvirkštinės kinematikos atveju ieškomi pavarų pasisukimo kampai, kai žinomi atstumai tarp jungčių ir galutinė padėtis [7].

Šiame darbe reikalinga atvirkštinė kinematika, kadangi nežinoma, kaip turi pasisukti pavaros, kad lazeris šviestų į mūsų nustatytą tašką. Norint suprasti, kaip atliekama atvirkštinės kinematikos analizė, panagrinėsime 5 ašių roboto ranką [8]:

1. Pirmiausia nubraižoma sujungimų schema, kurioje matomi rankos dalių matmenys (1.6 pav.).



1.6 pav. 5 ašių roboto rankos sujungimų schema

2. Remiantis Denavit-Hartenberg [9] tiesioginės kinematikos analizės metodu ir turimais roboto rankos duomenimis (3 lentelė) sudarome 4 koordinačių matricas ir randame jų sandaugą (1.1).

3 lentelė

Roboto rankos duomenys

Sujungimas i	d_i (mm)	a_i (mm)	α_i	θ_i
1	$d_1 = 275$	0	$\alpha_1 = -\pi/2$	θ_1
2	0	$a_2 = 275$	0	θ_2
3	0	$a_3 = 275$	0	θ_3
4	0	0	$\alpha_4 = -\pi/2$	θ_4
5	$d_5 = 195$	0	0	θ_5

$$^{i=1}A_i = T(Z,d)T(Z,\theta)T(X,a)T(X,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\,\theta_i & -sin\,\theta_i & 0 & 0 \\ sin\,\theta_i & cos\,\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\,\alpha_i & -sin\,\alpha_i & 0 \\ 0 & sin\,\alpha_i & cos\,\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \tag{1.1}$$

$$= \begin{bmatrix} cos\,\theta_i & -cos\,\alpha_i\,sin\,\theta_i & sin\,\alpha_i\,sin\,\theta_i & a_i\,cos\,\theta_i \\ sin\,\theta_i & cos\,\alpha_i\,cos\,\theta_i & -sin\,\alpha_i\,cos\,\theta_i & a_i\,sin\,\theta_i \\ 0 & sin\,\alpha_i & cos\,\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3. Įstatę lentelės duomenis gauname penkias matricas (1.2-1.6).

$$^{0}A_1 = T(Z,d_1)T(Z,\theta_1)T(X,0)T\left(X,-\frac{\pi}{2}\right) = \begin{bmatrix} cos\,\theta_1 & 0 & -sin\,\theta_1 & 0 \\ sin\,\theta_1 & 0 & cos\,\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.2}$$

$$^{1}A_2 = T(Z,0)T(Z,\theta_2)T(X,a_2)T(X,0) = \begin{bmatrix} cos\,\theta_2 & -sin\,\theta_2 & 0 & a_2\,cos\,\theta_2 \\ sin\,\theta_2 & cos\,\theta_2 & 0 & a_2\,sin\,\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.3}$$

$$^{2}A_3 = T(Z,0)T(Z,\theta_3)T(X,a_3)T(X,0) = \begin{bmatrix} cos\,\theta_3 & -sin\,\theta_3 & 0 & a_3\,cos\,\theta_3 \\ sin\,\theta_3 & cos\,\theta_3 & 0 & a_3\,sin\,\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.4}$$

$$^{3}A_4 = T(Z,0)T(Z,\theta_4)T(X,0)T\left(X,-\frac{\pi}{2}\right) = \begin{bmatrix} cos\,\theta_4 & 0 & -sin\,\theta_4 & 0 \\ sin\,\theta_4 & 0 & cos\,\theta_4 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.5}$$

$$^{4}A_5 = T(Z,d_5)T(Z,\theta_5)T(X,0)T(X,0) = \begin{bmatrix} cos\,\theta_5 & -sin\,\theta_5 & 0 & 0 \\ sin\,\theta_5 & cos\,\theta_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.6}$$

4. Sudauginę šias matricas gauname pagrindinę matricą (1.7), iš kurios galime rasti *x*, *y*, *z* koordinates.

$$
{}^{R}T_{H} = {}^{0}A_{5} = {}^{0}A_{1} \cdot {}^{1}A_{2} \cdot {}^{2}A_{3} \cdot {}^{3}A_{4} \cdot {}^{4}A_{5} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\
= \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & -1 & 0 & y \\ 0 & 0 & -1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.7}
$$

5. Iš gautos matricos galime išreikšti pasisukimo kampus (1.8-1.12).

$$\theta_1 = \theta_5 = a\tan 2\left(\frac{y}{x}\right), \tag{1.8}$$

$$\theta_2 = a\tan 2\left(\frac{(a_2 + a_3 \cos\theta_3)(d_1 - d_5 - z) - a_3 b \sin\theta_3}{(a_2 + a_3 \cos\theta_3) b - a_3 b \sin\theta_3 (d_1 - d_5 - z)}\right), \tag{1.9}$$

$$\theta_3 = a\tan 2\left(\frac{b^2 (d_1 - d_5 - z)^2 - a_2^2 - a_3^2}{2 a_2 a_3}\right), \tag{1.10}$$

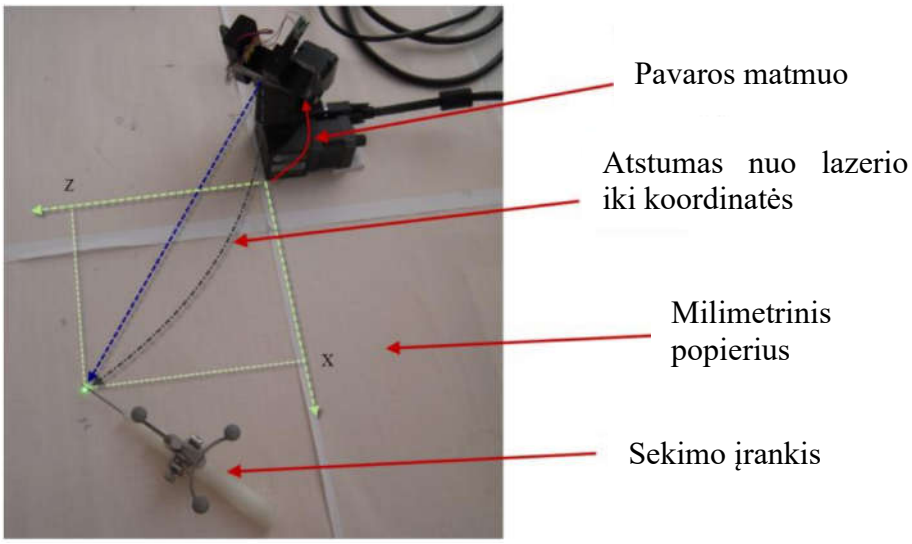
$$\theta_4 = -\theta_2 - \theta_3. \tag{1.11}$$

$$b = \pm\sqrt{\left(x^2 + y^2\right)}. \tag{1.12}$$

Taigi, pasinaudoję atvirkštinės kinematikos analizės metodais, galima rasti pasisukimo kampus bet kokiai panašiai sistemai.

1.1.5. Eksperimentinis tyrimas

Pavarų posūkių kampai, kai lazerio pluoštelis nutaikytas į tam tikrą koordinatę, gali būti rasti ir eksperimentiniu būdu. Tokiu atveju ant plokštumos dedamas milimetrinis popierius, kuriame sukuriama koordinačių sistema ir pažymimas atskaitos taškas. Lazerio pluoštelis nukreipiamas į reikiamą koordinatę ir užrašomi pavarų posūkių kampai (1.7 pav.) [10].

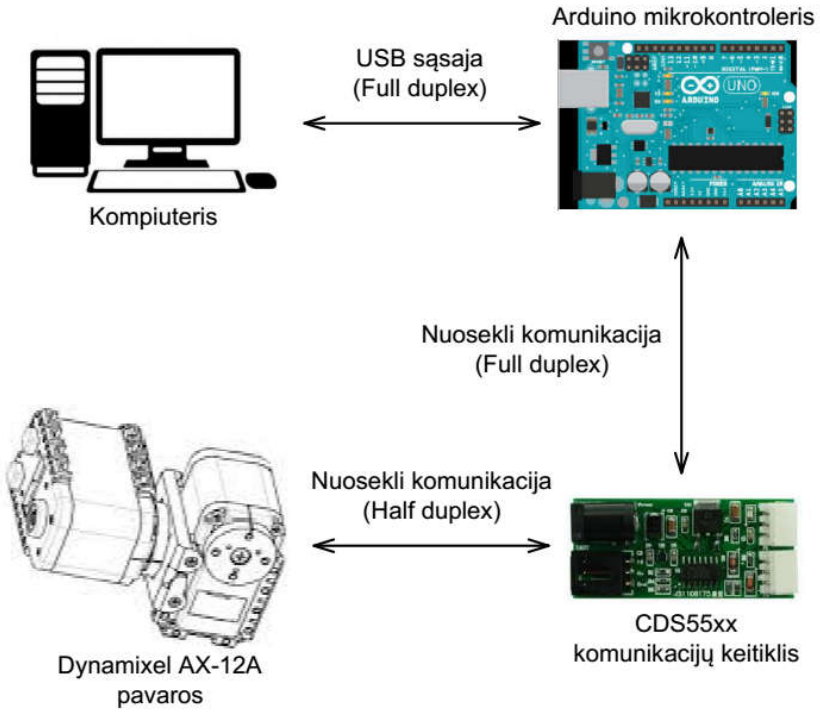


1.7 pav. Eksperimentinis sistemos tyrimas [10]

Pasinaudojant Shinji Umeyama aprašytu metodu [11], galima surasti koordinačių sistemų transformacijos matricas pagal keturis eksperimentiškai nustatytus taškus lazerio ir plokštumos koordinačių sistemose.

1.1.6. Sprendimo planas

Šiame darbe nagrinėjama sistema susideda iš metalinio rėmo, ant kurio montuojama dviašė pavarų sistema su lazeriu. Pavarų sistema susideda iš dviejų servo pavarų, kurios montuojamos viena ant kitos. Šios pavaros pritaikytos dirbti dideliu tikslumu, ir jos turi grįžtamąjį ryšį. Valdymo sistemai reikalingas kompiuteris, iš kurio bus siunčiamos komandos ir gaunama informacija iš valdiklio. Ryšys tarp šių elementų atliekamas per *USB* sąsają. Taip pat reikalingas komunikacijų *Full-duplex/Half-duplex* keitiklis, kuris leidžia komunikuoti su pavaromis. Valdymo sistemos funkcinė schema pateikta 1.8 paveiksle.



1.8 pav. Valdymo sistemos funkcinė schema [12, 13, 14, 15]

Vartotojo sąsaja kompiuteryje gali būti kuriama įvairiomis programomis, pavyzdžiui, *MatLAB* arba *LabView* programiniais paketais. Pavaros per komunikacijų keitiklį valdomos paprastomis komandomis iš jau sukurtų *Arduino Dynamixel* bibliotekų.

Sumontuotą sistemą būtina ištirti norint lazerį pozicionuoti plokštumoje *X* ir *Y* koordinatėmis. Tam tikslui galima pasinaudoti atvirkštinės kinematikos principais ir apskaičiuoti pavarų posūkių kampus prie norimos koordinatės arba atlikti eksperimentinį tyrimą ir iš gautų duomenų apskaičiuoti posūkių kampus. Žinoma, didžiausias tikslumas bus pasiektas pasinaudojus abiem metodais. Taip pat eksperimentų metu galima naudoti paprastą lazerinį žymeklį.

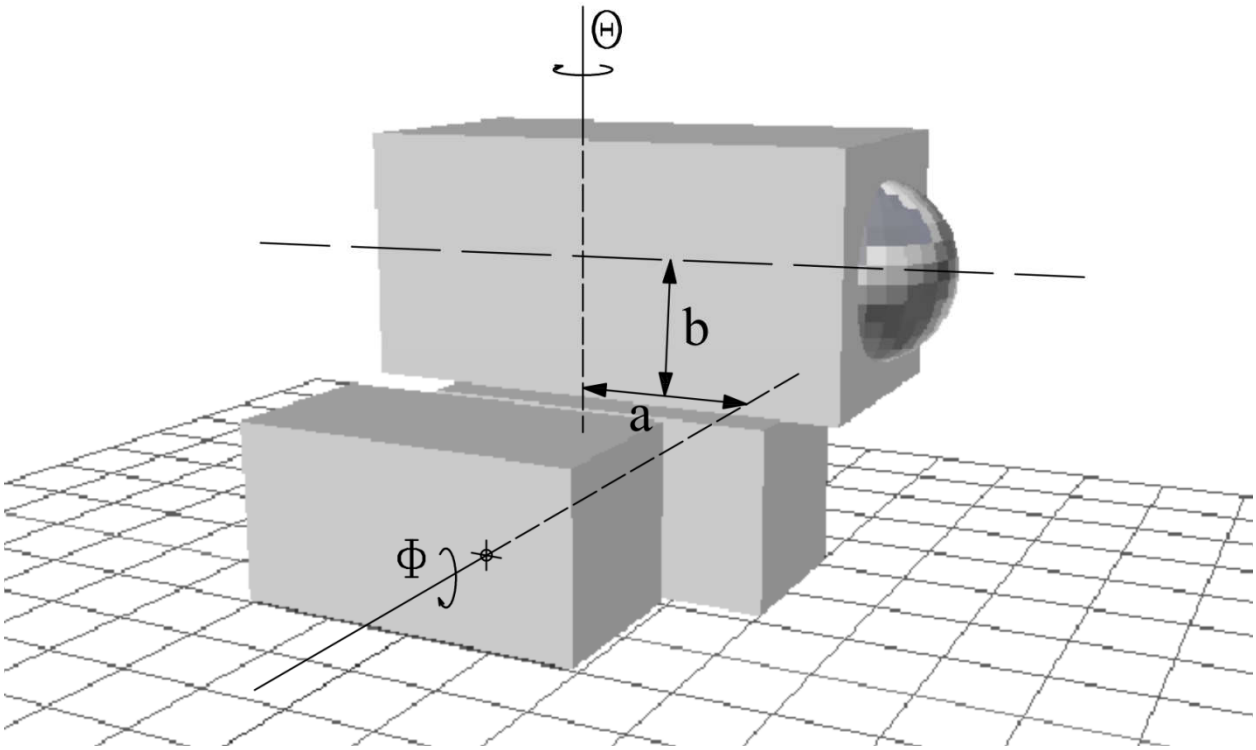
Remiantis 1.8 pav. atvaizduota funkcine schema, sistemoje naudojamas mikrovaldiklis turi turėti du nuoseklius komunikacijų prievadus. Šiuo atveju dažniausiai naudojamas *Arduino Uno* yra netinkamas, kadangi jis turi tik vieną nuoseklųjį prievadą, todėl reikia naudoti *Arduino Mega*, kuris turi 4 prievadus.

2. PROJEKTINĖ DALIS

2.1. Sistemos valdymas

2.1.1. Kinematinė schema

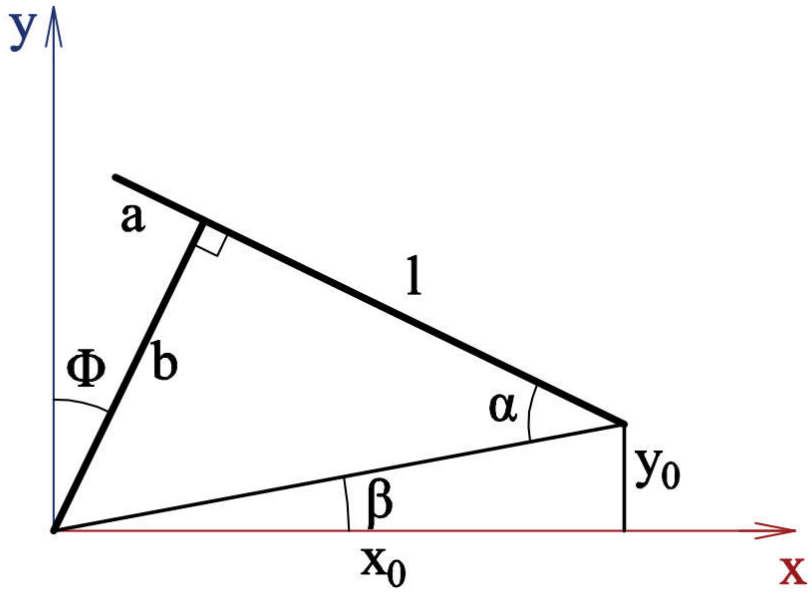
Kuriant nagrinėjamos sistemos kinematinę schemą būtina atsižvelgti į pavarų matmenis, kadangi jų ašys yra skirtinguose erdvės taškuose. Kinematinėje schemoje (2.1 pav.) pavarų matmenys pažymėti *a* ir *b* raidėmis. Matmuo *a* yra atstumas nuo vienos ašies iki kitos, o *b* – atstumas nuo antros ašies iki lazerio pluoštelio. Pirmosios pavaros posūkio kampas pažymėtas Φ, o antrosios – Θ.



2.1 pav. Kinematinė sistemos schema: *a* – pirmos pavaros matmuo; *b* – antros pavaros matmuo; Φ – pirmos pavaros posūkio kampas; Θ – antros pavaros posūkio kampas

2.1.2. Kinematiniai skaičiavimai

Norint atlikti skaičiavimus pirmiausia kinematinė schema buvo suprojektuota *X* ir *Y* ašyse, kai lazeris pozicionuojamas ekrane pagal *Y* koordinatę (2.2 pav.). Parametrai *a* ir *b* atitinka pavarų matmenis, x_0 atitinka atstumą nuo pirmosios ašies iki ekrano, kuriame pozicionuojamas lazerio pluoštelis. Visi šie dydžiai išmatuojami eksperimentiškai ir yra pateikti 4 lentelėje. y_0 yra skaičiuojant įvedama *y* koordinatė ant ekrano.



2.2 pav. Kinematinės schemos projekcija *X* ir *Y* ašyse

4 lentelė

Eksperimentiškai išmatuoti dydžiai

Paaiškinimas	Simbolis	Vertė
Atstumas nuo pirmos ašies iki ekrano	x_0	155 mm
Pirmos pavaros matmuo	b	5,5 mm
Antros pavaros matmuo	a	3,3 mm

Šioje schemoje ieškomas dydis – pasisukimo kampas Φ. Jį galima rasti pasinaudojus (2.1) lygtimi:

$$\phi = \alpha - \beta \,. \tag{2.1}$$

α ir β kampai randami remiantis Pitagoro teorema ir trigonometrija (2.2, 2.3):

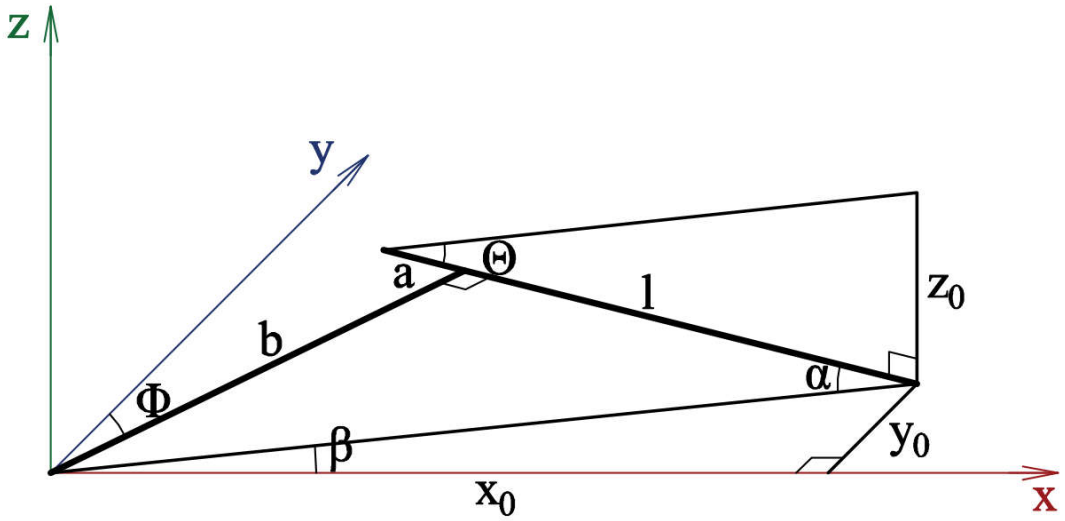
$$\alpha = arcsin \frac{b}{\sqrt{x_0^2 + y_0^2}}, \tag{2.2}$$

$$\beta = arctg \frac{y_0}{x_0} \,. \tag{2.3}$$

Taigi, kampas Φ randamas iš 2.4 lygties:

$$\phi = arcsin \frac{b}{\sqrt{x_0^2 + y_0^2}} - arctg \frac{y_0}{x_0} \,. \tag{2.4}$$

Toliau kinematinė schema suprojektuota trimatėje erdvėje (2.3 pav.). Tuomet galima apskaičiuoti *Θ* kampą, kuris įvertina antros ašies pasisukimą pozicionuojant lazerio pluoštelį ekrane *X* ašimi. Čia reikalingas a matmuo, kadangi pirmoji ašis yra šiek tiek atitolusi nuo antrosios ašies.



2.3 pav. Kinematinė schema trimatėje erdvėje

Θ kampas apskaičiuojamas remiantis 2.5 lygtimi:

$$\theta = arctg\frac{z_0}{l}. \tag{2.5}$$

Čia (2.6):

$$l = a + \sqrt{x_0^2 + y_0^2 - b^2}\ . \tag{2.6}$$

Todėl galutinė lygtis atrodo taip (2.7):

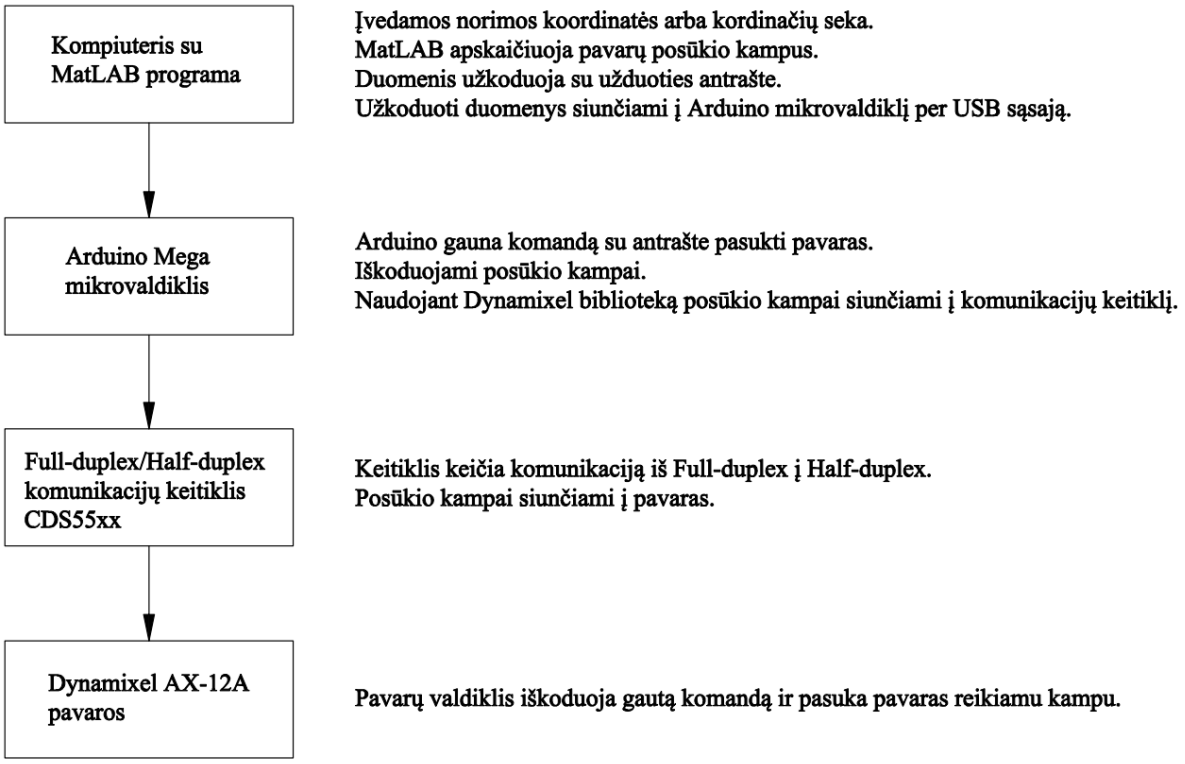
$$\theta = arctg\frac{z_0}{a + \sqrt{x_0^2 + y_0^2 - b^2}}. \tag{2.7}$$

Taigi, remiantis 2.4 ir 2.7 lygtimis galime rasti abiejų ašių pasisukimo kampus lazerį pozicionuojant pasirinktose koordinatėse.

2.1.3. Valdymo algoritmas

Sistemai valdyti reikalingas kompiuteris su *MatLAB* programiniu paketu, *Arduino Mega* mikrovaldiklis, turintis 4 nuosekliuosius prievadus, komunikacijų keitiklis *CDS55xx* ir *Dynamixel AX-12A* pavaros. Valdymas pradedamas *MatLAB* aplinkoje (2.4 pav.). Programoje įvedamos norimos

koordinatės arba jų seka, ir programa, pagal anksčiau darbe gautas lygtis, apskaičiuoja reikiamus pavarų posūkio kampus. Šie kampai užkoduojami ir siunčiami į *Arduino* mikrovaldiklį per *USB* sąsają. Valdiklyje gautos komandos iškoduojamos. Gavus komandą ir kampų vertes valdiklis jas siunčia pavaroms per komunikacijų keitiklį, o pavarų valdiklis pasuka pavaras reikiamais kampais.



2.4 pav. Sistemos valdymo algoritmas

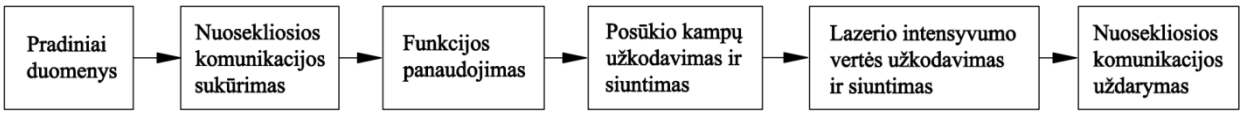
Komandos koduojamos tam tikra tvarka, kad *Arduino* gautų teisingą informaciją ir nepadarytų klaidos. Kodavimo struktūrinė schema pateikta 2.5 pav. Pirmiausia komanda prasideda antrašte. Ji naudojama atskirti komandoms, pavyzdžiui, viena antraštė skirta servo pavaroms valdyti, kita – lazerio pluoštelio intensyvumui nustatyti ir jį įjungti. Po antraštės siunčiamas dvitaškis, kuris nurodo antraštės pabaigą. Toliau siunčiamos reikiamos kintamųjų vertės. Tarp verčių būtina siųsti atskyrimo ženklą, šiuo atveju kabliataškį, kad vertes būtu galima atskirti. Tokiu principu užkoduotą komandą *Arduino* mikrovaldiklis gali lengvai iškoduoti ir atlikti norimus veiksmus.

2.5 pav. Kodavimo struktūrinė schema

2.1.4. Valdymo programa

Kaip ir minėta, sistemos valdymas prasideda kompiuteryje, *MatLAB* sukurtoje programoje. Pirmiausia sukurta funkcija (1 priedas), kurios įėjime įvedamos norimos koordinatės bei atstumas iki ekrano ir išėjime gaunami pavarų posūkių kampai. Funkcijoje įrašytos a ir b matmenų vertės, taip pat pradiniai posūkio kampai, kadangi pavaros turi būti vidurinės padėties. Sistemoje naudojamos pavaros 300° posūkio kampui turi 1 024 padėtis, todėl vidurinė padėtis bus 512 žingsnyje. Toliau pagal kinematikos lygtis gaunami posūkių kampai. Funkcijoje koordinatės apribotos 50 mm atstumu nuo koordinačių pradžios, todėl, jei įvedamos didesnės koordinatės, pavaros grįžta į pradinę padėtį. Galiausiai gauti kampai perskaičiuojami į pavarų žingsnius.

Pagrindinėje *MatLAB* programoje (2.6 pav., 2 priedas) pirmiausia suvedami reikalingi parametrai – *x* ir *y* koordinatės, atstumas iki ekrano ir lazerio intensyvumo vertė. Toliau vyksta komunikacijos sukūrimas su *Arduino* mikrovaldikliu. Čia patikrinama, ar nėra jau sukurtų komunikacijų (jeigu yra – jos panaikinamos), kadangi *MatLAB* vienu metu gali turėti tik vieną nuosekliąją komunikaciją, ir sukuriama nauja, su 115 200 bitų per sekundę greičiu. Sukūrus komunikaciją panaudojama anksčiau sukurta funkcija, ir gauti kampai užkoduojami pagal algoritme aprašytą principą su antrašte „Servo:“. Šioje programoje taip pat galima nusiųsti lazerio intensyvumo vertę. Šios komandos antraštė yra „Laser:“. Programos pabaigoje nuoseklioji komunikacija uždaroma.



2.6 pav. *MatLAB* programos struktūra

Arduino mikrovaldiklio programoje (3 priedas) sukurti reikalingi kintamieji, pradinės pavarų posūkių kampų vertės ir lazerio intensyvumas. Šioje programoje panaudotos *Dynamixel*, *SoftwareSerial* ir *Wire* bibliotekos. Programos inicializavimo etape sukuriamos komunikacijos su pavaromis ir kompiuteriu bei įrašomos pradinės sistemos elementų vertės. Toliau vyksta gautų komandų iškodavimas pagal algoritmą. Gavus komandą „Servo:“ pagal iškoduotus posūkių kampus pasukamos pavaros, o gavus komandą „Laser:“ – pakeičiamas lazerio intensyvumas.

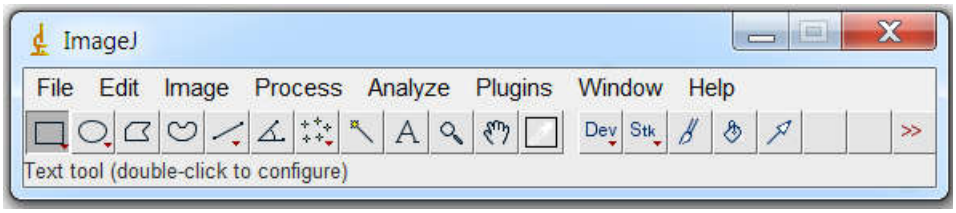
2.2. Statikos tyrimas

2.2.1. Tyrimo duomenys

Sukūrus valdymo programą atlikti du sistemos tyrimai – statikos ir dinamikos. Statikos tyrimo metu kamera, esančia pozicionieriaus dešinėje, buvo filmuojamas lazerio darbas, kai šis piešė tris geometrines figūras – apskritimą, kvadratą ir trikampį. Figūros buvo sudaromos iš maksimalaus skaičiaus taškų. Dinamikos tyrimo metu aukštos raiškos kamera dideliu greičiu buvo filmuojamas greitas lazerio koordinatės pakeitimas. Pirmojo tyrimo tikslas – išanalizuoti sistemos tikslumą, o antrojo – įvertinti inercijos ir dinaminių procesų sukeltus nuokrypius.

2.2.2. Duomenų apdorojimas

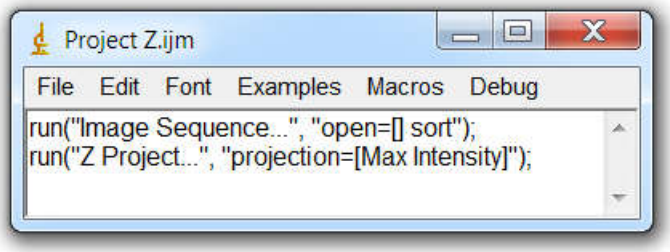
Statikos tyrimo metu gauta informacija negali būti iš karto įvertinta, kadangi kamera stovėjo kampu į ekraną ir negalima daryti tikslių išvadų. Tam reikalingas informacijos apdorojimas specialiomis programomis, todėl šiam tikslui pasirinkta programa *ImageJ*. Ši programa skirta vaizdams apdoroti ir dažnai naudojama moksliniuose tyrimuose, kadangi yra labai paprasta ir plačiai pritaikoma. Programa parašyta *Java* programavimo aplinkoje ir yra atviro kodo. Ja galima apdoroti beveik visų tipų nuotraukas, matuoti įvairius parametrus ir netgi apdoroti visą nuotraukų masyvą (2.7 pav.).



2.7 pav. Pagrindinis *ImageJ* programos langas

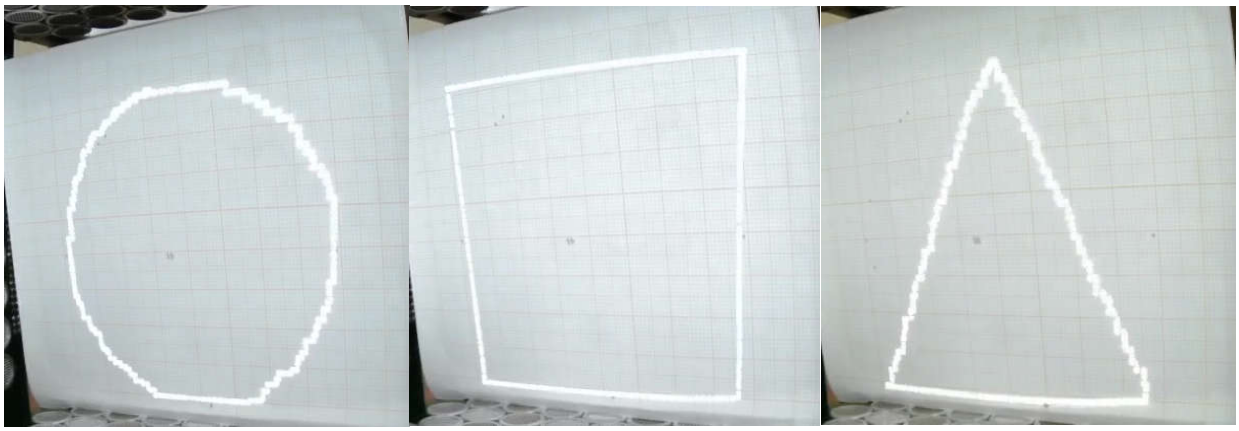
Norint apdoroti turimą informaciją *ImageJ* programa pirmiausia filmuota medžiaga išskaidyta į atskirus kadrus. Tam galima panaudoti įvairias programas, tačiau reikia atkreipti dėmesį, kad skaidant medžiagą į kadrus kokybė turi nukentėti kuo mažiau, kadangi kiekvieną kartą apdorojant vaizdus programomis kokybė prastėja, o kuo daugiau apdorojimo žingsnių daroma, tuo galutinis rezultatas darosi prastesnis. Filmuotai medžiagai išskaidyti pasinaudota *Dvdvideosoft* sukurta programa *Video to JPG Converter*, kuri maksimaliai išsaugo kadrų kokybę. Šioje programoje įkeliame vaizdo medžiagą ir pasirenkame, kas kiek kadrų ją skaidyti. Gauti kadrai sudedami atskirame aplanke ir sunumeruojami eilės tvarka.

Turint atskirus kadrus galima pradėti juos apdoroti programa *ImageJ*. Darbui palengvinti sukurta makro-komanda (2.8 pav.), kurią panaudojus iš karto atidaromas kadrų įkėlimo langas, o juos pasirinkus iš karto vykdoma *Z Project* komanda su *Max Intensity* pasirinkimu.



2.8 pav. Sukurta makro-komanda.

Z Project komanda suklijuoja nuotraukų (kadrų) masyvą į vieną nuotrauką ir gali išryškinti tam tikras vietas arba sumažinti susidariusį triukšmą. Pasirinkimas *Max Intensity* kiekviename taške išrenka didžiausią vertę turintį kadrą tame taške, todėl pasinaudojus šia funkcija gautos lazeriu nupieštos figūros (2.9 pav).



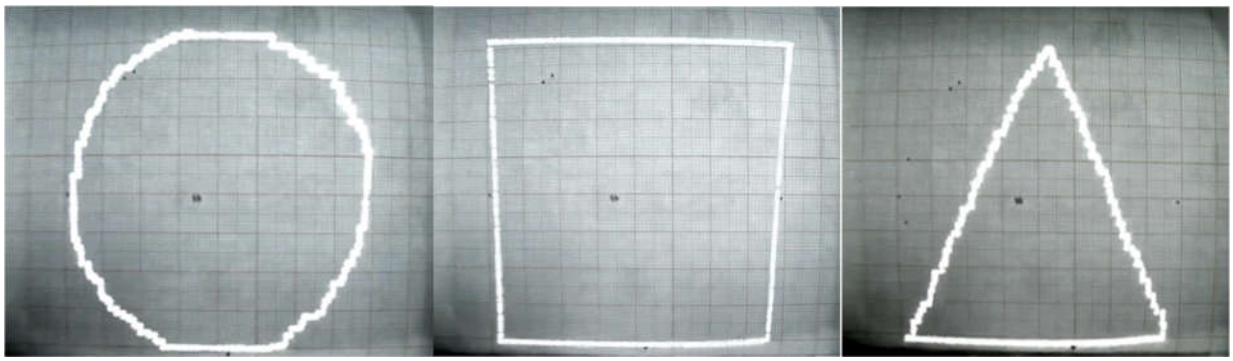
2.9 pav. Lazeriu nupieštos figūros

2.2.3. Kitų parametrų įvertinimas

Gautose nuotraukose matomos lazerio nupieštos figūros, tačiau dar negalima vertinti turimos informacijos. Pirmiausia koreguotas nuotraukų kontrastas, kad milimetrinio popieriaus linijos matytųsi kuo aiškiau. Tada įvertinta perspektyva, kadangi filmuota kampu į ekraną. Perspektyva gali būti koreguojama programa *GIMP 2*. Programoje gauta tokia transformacijos matrica (2.8):

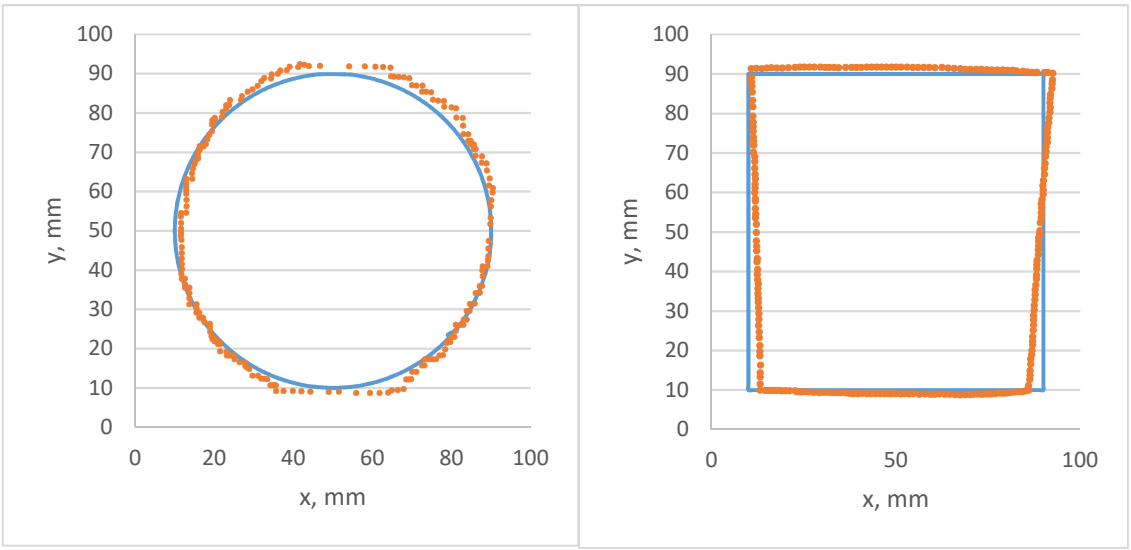
$$A = \begin{pmatrix} 1{,}46184 & -0{,}15483 & -25 \\ 0{,}16488 & 1{,}16249 & -127 \\ 0{,}00067 & -0{,}00013 & 1 \end{pmatrix}. \tag{2.8}$$

Atlikus perspektyvos koregavimą, pastebėta, kad vaizdai šiek tiek išsikraipė, todėl koreguotas ir mastelis. *X* ašimi mastelis koreguotas 17 taškų, *y* ašimi – 73 taškais. Atlikus visus koregavimo darbus gauta apdorota tyrimo informacija (2.10 pav.).

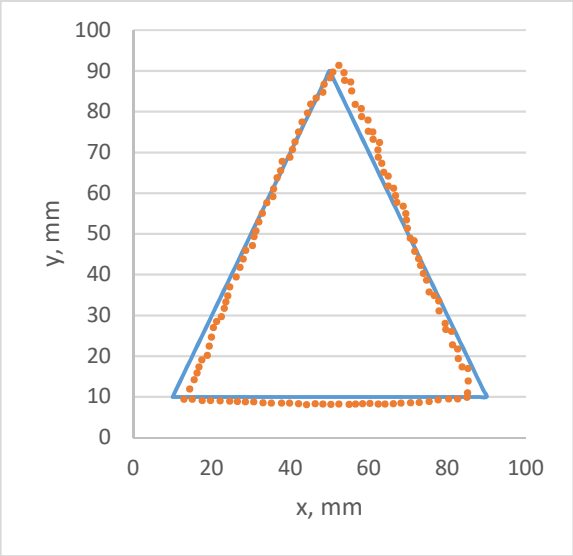


2.10 pav. Apdorota statikos tyrimo informacija

Tolimesniam darbui informacija perkelta į *AutoCAD* aplinką. Čia pažymėti visi atskiri taškai ir gautos jų koordinatės. Visos gautos figūros palygintos su tomis, kurios užprogramuotos piešimui (2.11, 2.12 pav.).



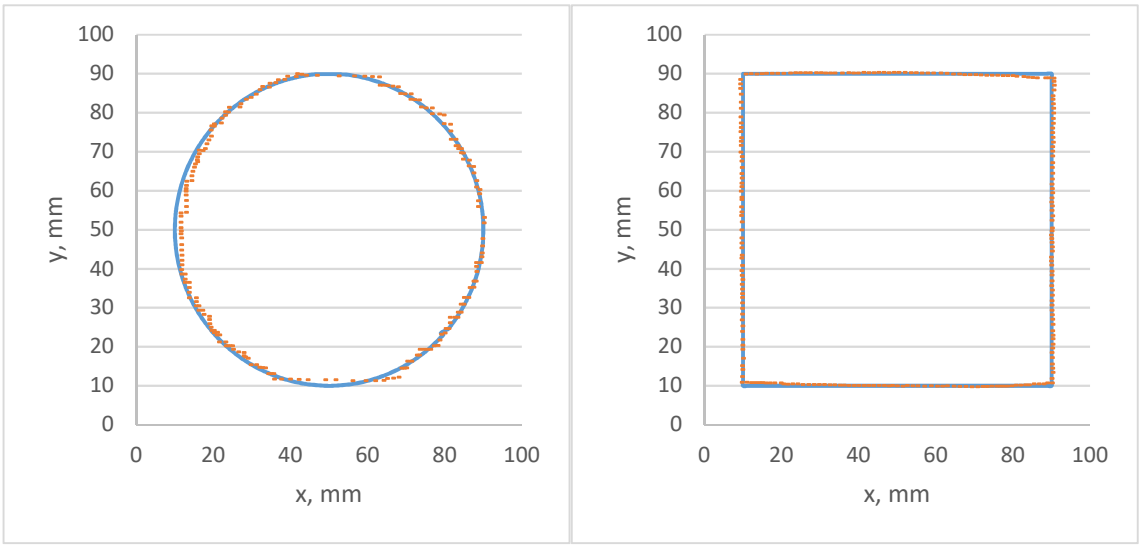
2.11 pav. Lazeriu nupieštas apskritimas ir kvadratas



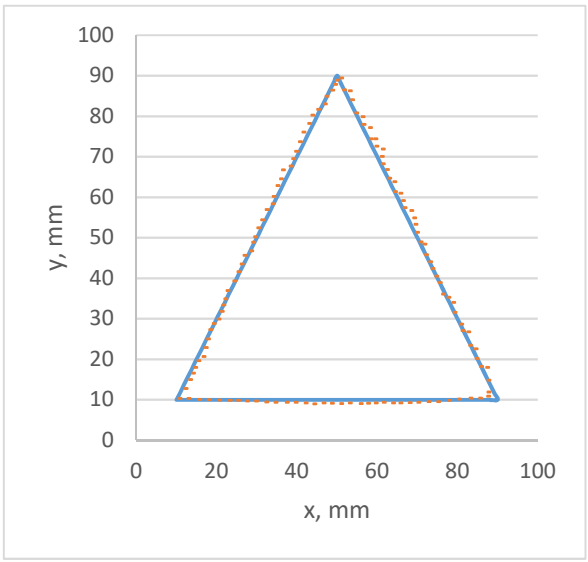
2.12 pav. Lazeriu nupieštas trikampis

2.2.4. Rezultatų įvertinimas ir skaičiavimai

Remiantis 2.11, 2.12 pav. padaryta išvada, kad koordinatėse yra sisteminė klaida, kadangi nupieštos figūros šiek tiek ištemptos. Galima daryti prielaidą, kad šios sisteminės klaidos galėjo atsirasti dėl kelių priežasčių: ekrano pasvirimo ir pavarų pasvirimo įtvirtinimo taške. Dėl šių pasvirimų atstumas iki ekrano gaunamas nevienodas, todėl rezultatai iškraipomi. Tokio tipo klaidas ištaisyti galima kalibruojant sistemos taškus, t. y. ekraną ir pavarų įtvirtinimo tašką. Sisteminės klaidos taip pat galėjo atsirasti apdorojant tyrimo medžiagą programomis, tačiau koregavimo metu buvo siekiama kuo tiksliau atkurti milimetrinio popieriaus proporcijas, todėl šie nukrypimai toliau nevertinami.



2.13 pav. Apskritimas ir kvadratas eliminavus atpažintus sisteminius nuokrypius



2.14 pav. Trikampis eliminavus atpažintus sisteminius nuokrypius

Kadangi sisteminės klaidos iškraipė rezultatus, jas būtina pašalinti norint atlikti statistinius skaičiavimus. Todėl prieš statistinių nuokrypių skaičiavimą figūros koreguotos atsižvelgiant į atpažintas sistemines klaidas. Remiantis gautomis koreguotomis figūromis (2.13, 2.14 pav.) atlikti statistiniai skaičiavimai.

Statistiškai likusius nuokrypius galima įvertinti apskaičiuojant dispersiją ir standartinį nuokrypį. Dispersija – viena populiariausių matavimų įverčių sklaidos charakteristikų. Jos privalumas yra tas, kad atsižvelgiama į visus duomenis, ir pateikiamas vidutinis skirtumų nuo vidurkio kvadratas [16]. Dispersija randama remiantis 2.9 lygtimi:

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}\Delta x^2 , \tag{2.9}$$

čia: s^2 – dispersija; N – taškų skaičius; Δx – nuokrypis x ašimi.

Apskaičiavus dispersiją dar negalima spręsti apie nuokrypius, kadangi nuokrypiai pakelti kvadratu. Tam skaičiuojamas vidutinis kvadratinis nuokrypis (2.10). Kadangi jis matuojamas tokiais pačiais vienetais kaip ir kitokie nuokrypiai, jį lengviau interpretuoti ir lyginti su duomenimis. Skaičiavimų rezultatai pateikti 5 lentelėje.

$$s = \sqrt{s^2} . \tag{2.10}$$

5 lentelė

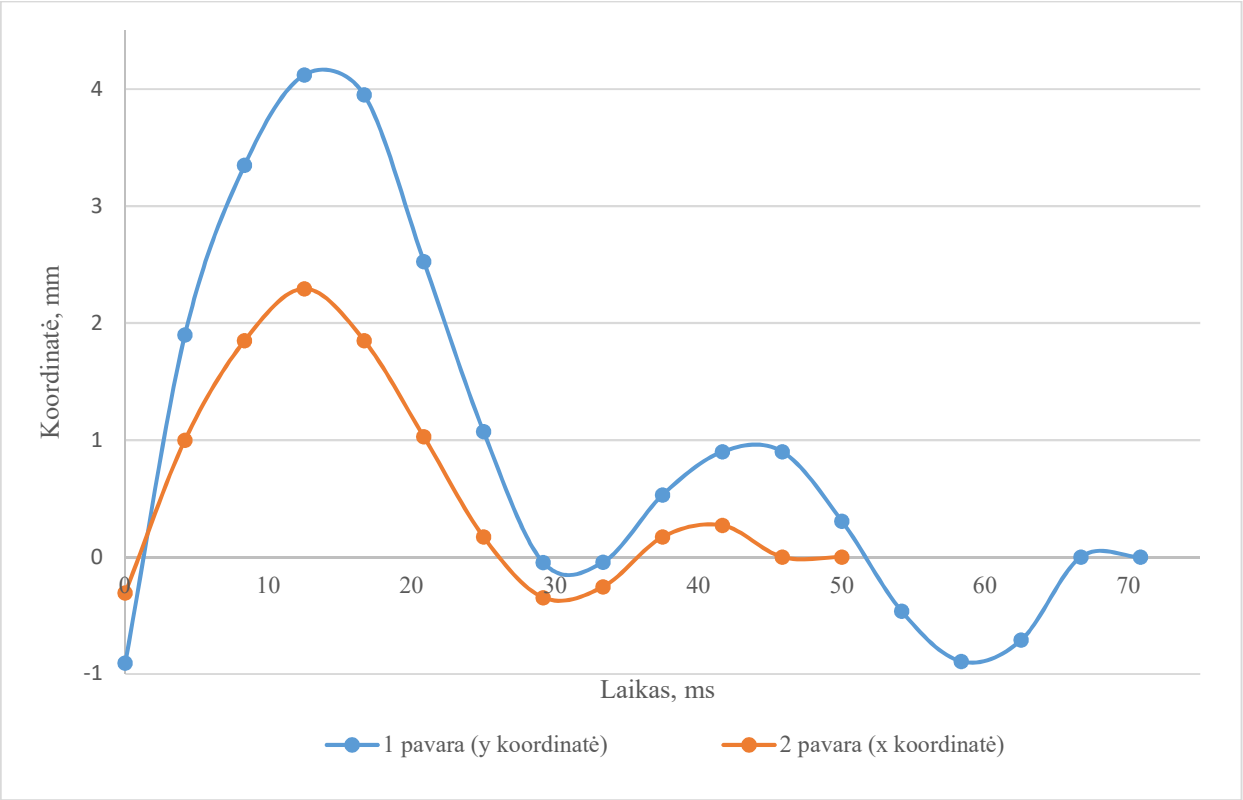
Skaičiavimų rezultatai

Figūra	Dispersija		Vidutinis kvadratinis nuokrypis	
	x (mm^2)	y (mm^2)	x (mm)	y (mm)
Apskritimas	0,66	0,89	0,812	0,94
Kvadratas	0,71	0,62	0,85	0,79
Trikampis	1,51	2,66	1,23	1,63

2.3. Dinamikos tyrimas

2.3.1. Tyrimo duomenys

Dinamikos tyrimo metu koordinatės keičiamos atskirai X ir Y ašimi. Atstumai tarp koordinačių maksimalūs viena ašimi (100 mm), kad būtų galima užfiksuoti svyravimus apie tašką, į kurį lazeris turi šviesti. Norint pamatyti tokius svyravimus, panaudotas *Canon SX40 HS* fotoaparatas, galintis filmuoti 240 kadrų per sekundę greičiu. Iš gautų kadrų gautos koordinatės ir apskaičiuota, kad vienas kadras trunka 4,167 ms. Gauti rezultatai pateikti 2.15 pav.



2.15 pav. Dinamikos tyrimo rezultatai

2.3.2. Rezultatų įvertinimas

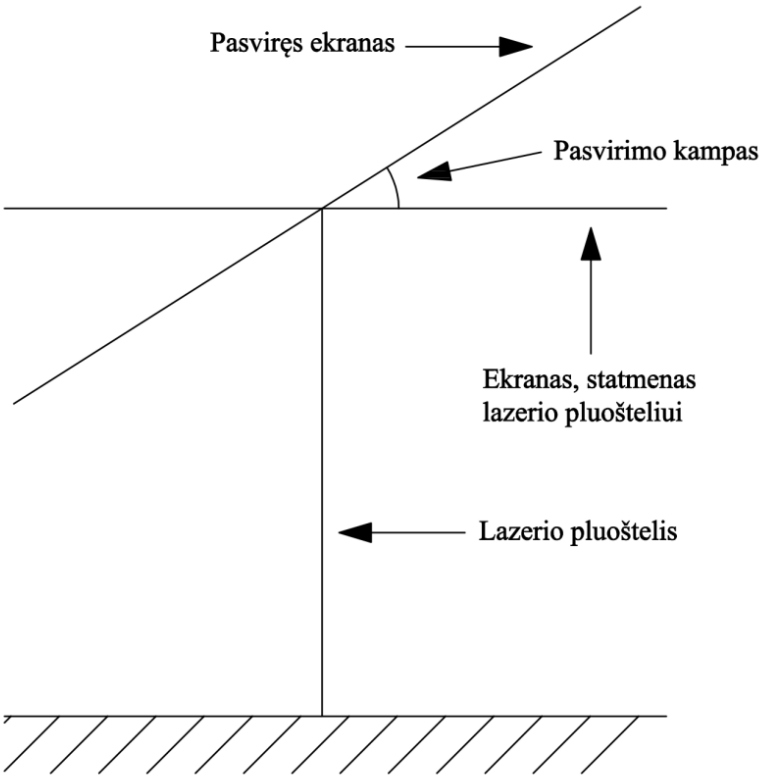
Iš diagramos matyti, kad pirmoji pavara apie tašką susvyruoja 4 kartus, o antroji – tris. Taip pat antrosios svyravimai mažesni ir koordinatę ji pasiekia greičiau. Visa sistema nusistovi apytiksliai per 70 ms. Gauti svyravimai apie tašką vyksta dėl sistemos inercijos – jos elementai turi tam tikrą masę, matmenis ir jie juda tam tikru greičiu. Skirtumai tarp pavarų gaunami todėl, kad pirmoji pavara pozicionuoja antrąją pavarą kartu su lazeriu, o antroji – tik lazerį. Dėl to skiriasi masės ir inercija, o tai lemia pozicionavimo skirtingus dinaminius nuokrypius. Įtaką taip pat daro ir pavarų judėjimo greitis, tačiau tyrimo metu jis maksimalus tiek vienai pavarai, tiek kitai. Dinaminės paklaidos taip pat atsiranda dėl vibracijų ašyse ir laisvumo pavarų įtvirtinimo taškuose.

Kadangi sistemos elementų masės ir matmenys yra pastovūs dydžiai, dinamines paklaidas galima kontroliuoti keičiant greitį – kuo jis mažesnis, tuo paklaidos mažesnės. Tačiau jei sistema turi veikti greitai ir tiksliai, greičio apriboti negalima. Tokiu atveju galima mažinti sistemos masę arba keisti konstrukciją. Taip pat būtina kiek įmanoma labiau sumažinti vibracijas įtvirtinimo taškuose. Sektorinio valdymo sistemose tas ypač aktualu, kadangi visa sistema montuojama viename taške.

2.4. Sistemos kalibravimas

2.4.1. Sprendimo planas

Sistemos kalibravimo tikslas – pašalinti proceso eigoje susidarančius sisteminius nuokrypius, kurie gauti statikos tyrimo metu. Kaip ir minėta, šie nuokrypiai atsiranda dėl pavarų sistemos ir ekrano pasvirimų. Šiuos pasvirimus pirmiausia reikėtų mažinti pasitelkiant mechanines priemones: pozicionavimo sistema ir ekranas turi būti statmeni vienas kito atžvilgiu. Kadangi šiame darbe nagrinėjamą sistemą sukalibruoti mechaniškai sudėtinga, galima naudoti programinį kalibravimą. Programinio kalibravimo algoritmų yra labai įvairių, tačiau kiekvienas jų pritaikytas konkrečiai sistemai, todėl šiame darbe siekiama sukurti naują, šiai sistemai tinkantį kalibravimo algoritmą. Kuriant algoritmą daroma prielaida, kad pavarų sistema pasvirimo neturi, todėl skaičiuojamas tik ekrano pasvirimo kampas. Tokio algoritmo principinė schema pateikta 2.16 pav.

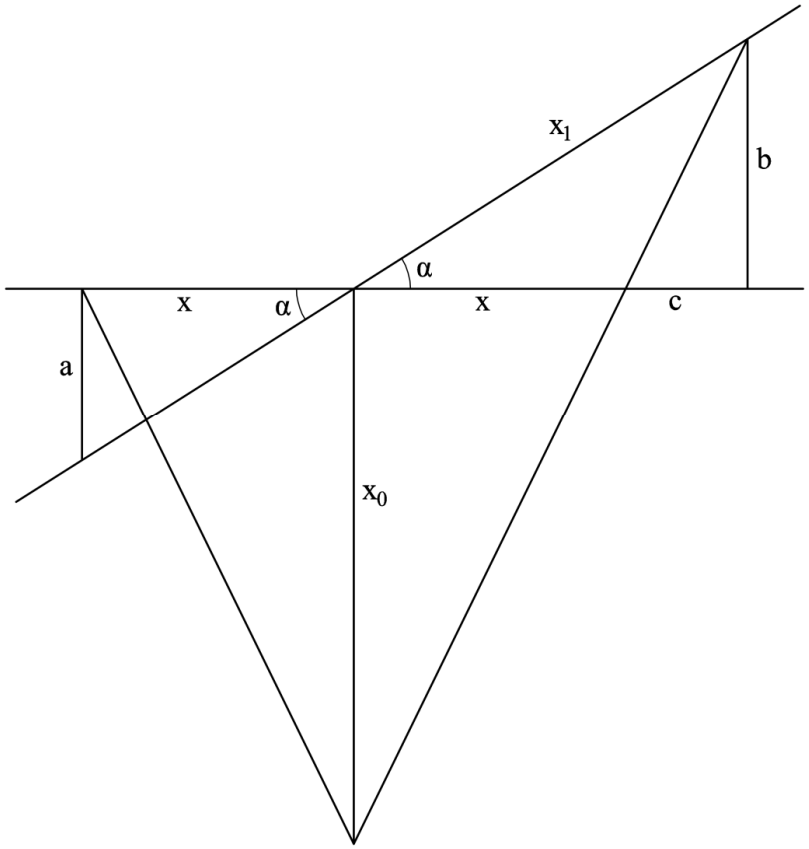


2.16 pav. Kalibravimo algoritmo principinė schema

2.4.2. Skaičiavimai

Norint koreguoti koordinates pagal ekrano pasvirimą, būtina žinoti, koks yra pasvirimo kampas tiek viena ašimi, tiek kita. Todėl pirmiausia ieškomas kampas α. Skaičiavimams atlikti nubraižyta schema, kurioje pavaizduotas ekrano pasvirimas viena ašimi (2.17 pav.). Šiuo atveju turi būti žinomi šie dydžiai: x_0 – atstumas nuo lazerio iki ekrano (išmatuotas eksperimentiškai), x –

koordinatė viena ašimi, į kurią pozicionuojamas lazerio pluoštelis, x_1 – eksperimentiškai išmatuota koordinatė ant ekrano (darant prielaidą, kad ji didesnė už koordinatę x).



2.17 pav. Ekrano pasvirimo kampo skaičiavimų schema

Remiantis šiomis sąlygomis ir nubraižyta schema galima užrašyti 2.11-2.14 lygtis:

$$\frac{a}{b}=\frac{x}{x+c}, \tag{2.11}$$

$$\frac{x_0}{b}=\frac{x}{c}, \tag{2.12}$$

$$a=tg\alpha\cdot x, \tag{2.13}$$

$$b=sin\alpha\cdot x_1. \tag{2.14}$$

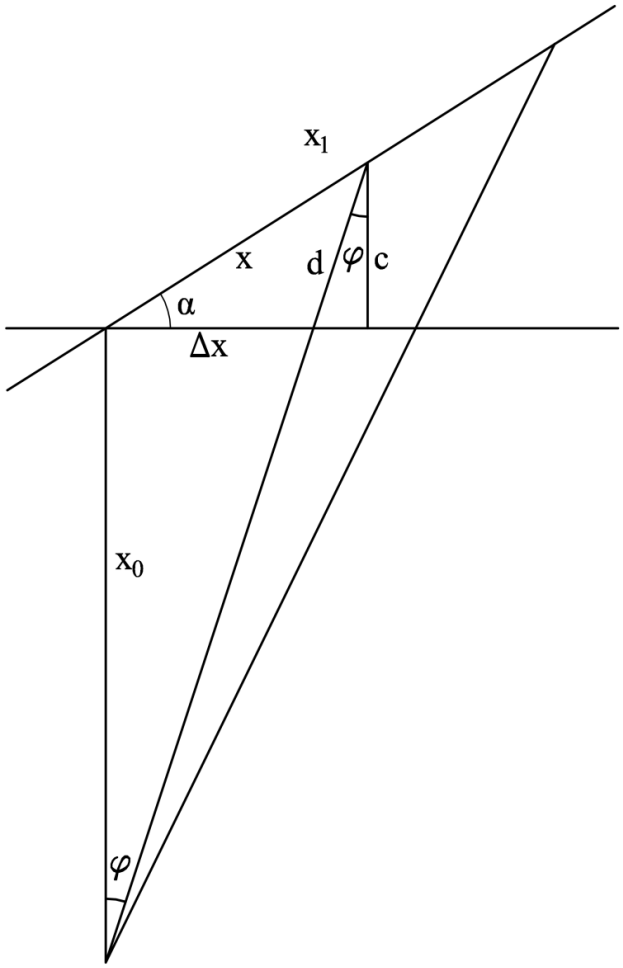
Iš 2.11-2.12 lygčių galima išreikšti 2.15 lygtį:

$$\frac{b-a}{a}=\frac{b}{x_0}. \tag{2.15}$$

Įstačius turimas lygtis į 2.15 išraišką ir atlikus reikalingus veiksmus gauname α kampo apskaičiavimo lygtį 2.16:

$$\alpha = arccos\left(\frac{x\sqrt{x_1^2x^2 + x_1^2x_0^2 - x^2x_0^2} + x_0^2}{x_1\left(x^2 + x_0^2\right)}\right). \tag{2.16}$$

Taigi, turint ekrano pasvirimo kampą galima atlikti pozicionavimo koregavimą viena ašimi. Toliau nubraižyta schema (2.18 pav.), kai lazeris pozicionuojamas ant pasvirusio ekrano norima koordinate. Šioje schemoje ieškomas dydis yra *Δx* – nauja koreguota koordinatė, į kurią lazeris turi šviesti, kad ant pasvirusio ekrano matytųsi reikiama koordinatė.



2.18 pav. Koordinatės koregavimo skaičiavimų schema

Taigi, remiantis antrąja schema gaunama 2.17-2.19 lygtis:

$$d = \frac{c}{cos\,\varphi} = \frac{sin\,\alpha \cdot x}{cos\,\varphi}, \tag{2.17}$$

$$tg\varphi = \frac{\Delta x}{x_0}, \tag{2.18}$$

$$cos^2\varphi = \frac{x_0^2}{\Delta x^2 + x_0^2}. \tag{2.19}$$

Toliau, remiantis kosinusų teorema gaunama 2.20 išraišką:

$$d^2 = x^2 + \Delta x^2 - 2\Delta x x cos\alpha. \tag{2.20}$$

Atlikus reikiamus veiksmus gauta koordinatės apskaičiavimo lygtis 2.21:

$$\left(sin^2\alpha \cdot x^2 - x_0^2\right)\Delta x^2 + \left(2cos\alpha \cdot x x_0^2\right)\Delta x + \left(sin^2\alpha \cdot x_0^2 x^2 - x_0^2 x^2\right) = 0. \tag{2.21}$$

Šios lygties toliau spręsti nereikia, kadangi ją galima išspręsti *MatLAB* komandos *Roots* pagalba. Kadangi gauta lygtis kvadratinė, ji turi du sprendinius. Bandymų metu nustatyta, kad pirmasis sprendinys naudojamas, kai kalibravimo ir pozicionavimo koordinatės vienodų ženklų (tik teigiamos arba tik neigiamos), o antrasis – kai ženklai skiriasi. Taigi, remiantis 2.16 ir 2.21 lygtimis galima koreguoti koordinates, taip ištaisant sisteminius nuokrypius.

2.4.3. Kalibravimo algoritmas

Sistemos valdymo programa papildyta kalibravimo algoritmu *MatLAB* aplinkoje (7 priedas). Norint atlikti sistemos kalibravimą, pirmiausia reikia atlikti eksperimentinius matavimus, pagal kuriuos randami pasvirimo kampai *X* ir *Y* ašimis. Šis etapas atliekamas tokiais žingsniais:

1. Lazeris pozicionuojamas viena ašimi teigiama ir neigiama tokio pat dydžio koordinate;
2. Išmatuojamas atstumas nuo centro iki pirmos ir antros koordinatės;
3. Programoje kaip x_1 įrašoma ta išmatuota koordinatė, kuri yra didesnė už tikrąją;
4. Programoje kaip *x* įrašoma su išmatuota koordinatė susijusi tikroji koordinatė, atkreipiant dėmesį į jos ženklą.

Šie žingsniai kartojami du kartus *X* ir *Y* ašims, o atitinkami dydžiai įrašomi reikiamose programos vietose. Toliau programa pagal 2.16 lygtį apskaičiuoja pasvirimo kampus ir pagal 2.21 lygtį atliekama koordinačių korekcija. Bandymų metu pastebėta, kad kalibravimui geriausia pasirinkti maksimalias koordinates *X* ir *Y* ašimis, taip gaunami tiksliausi pasvirimo kampai.

IŠVADOS

1. Šiame darbe išnagrinėta prototipinė sistemos konstrukcija – nustatyta, kad sistemoje naudojamas kompiuteris, *Arduino* mikrovaldiklis, komunikacijų keitiklis, servo pavaros ir lazeris.
2. Išanalizuota, kad pavarų posūkio kampus, kai lazeris pozicionuojamas plokštumoje pagal *X* ir *Y* koordinates galima rasti naudojant atvirkštinės kinematikos principus arba atliekant eksperimentinį tyrimą.
3. Pateiktas sistemos valdymo sprendimo planas, kuriame pavaizduota valdymo sistemos funkcinė schema, naudojama įranga bei aptarti pozicionavimo nustatymo principai.
4. Nubraižyta sistemos kinematinė schema ir gautos kampų apskaičiavimo lygtys pagal norimas koordinates.
5. Sukurtas sistemos valdymo algoritmas panaudojant *MatLAB* programą ir *Arduino* mikrovaldiklį bei parašyta valdymo programa, kurioje panaudotos kinematinės lygtys.
6. Atliktas sistemos statikos tyrimas, kurio metu gautos statinės ir sisteminės paklaidos. Eliminavus atpažintas sistemines klaidas gautas vidutinis standartinis nuokrypis, apytiksliai lygus 1 mm.
7. Atliktas dinamikos tyrimas, kurio metu gauti svyravimai apie koordinatę. Čia pateiktos galimos dinaminių paklaidų priežastys ir jų eliminavimo būdai. Nustatyta, kad sistemos greitaveika yra apytiksliai 70 ms.
8. Atsižvelgiant į statikos tyrimo metu gautus sisteminius nuokrypius sukurtas programinio kalibravimo algoritmas – nubraižytos skaičiavimų schemos, pagal kurias gautos ekrano pasvirimo kampo ir koordinatės korekcijos lygtys. Kalibravimo eiga aprašyta ir išbandyta *MatLAB* aplinkoje.

LITERATŪRA

1. KNIGHT, J. and REID, I. Automated Alignment of Robotic Pan-Tilt Camera Units using Vision. *International Journal of Computer Vision*, 07, 2006, vol. 68, no. 3. pp. 219-237 ISSN 09205691.
2. Ascendent Technology Group. *3MP-IR-20X-PTZ-CS Camera.* Žiūrėta 2017-04-15. Prieiga per internetą: http://www.ascendentgroup.com/store/configure/3MP-IR-20X-PTZ-CS/237.
3. Robotis. *Dynamixel-AX-12-User-Manual.* [Dynamixel AX-12].Robotis. 2006-06-14, 2006. Žiūrėta 2017-04-10. Prieiga per internetą: http://www.generationrobots.com/media/Dynamixel-AX-12-user-manual.pdf.
4. Digitalmeans. *AX-12 CDS55xx Driver Board.* Žiūrėta 2017-03-20. Prieiga per internetą: https://digitalmeans.co.uk/shop/ax12_cds55xx-servo_driver_board.
5. J. WAKEFIELD. *Driving Robotis Dynamixel AX-12A Servos from an Arduino.* 2014-03-23, 2014. Žiūrėta 2017-04-11. Prieiga per internetą: http://www.robosoup.com/2014/03/driving-robotis-dynamixal-ax-12a-servos.html.
6. Laserlands. *Focusable 4W 808nm 810nm Infrared IR Laser Diode Module + 5V Power Supply TTL.* Amazon. Žiūrėta 2017-05-17. Prieiga per internetą: https://www.amazon.com/Focusable-808nm-Infrared-Module-Supply/dp/B00UO80M1A#detail-bullets.
7. HUNT, J.A. Robot Kinematics and the Gantry-Tau Parallel Machine. *Industrial Robot*, 08/28; 2016/04, 2007, vol. 34, no. 5. pp. 362-367 ISSN 0143-991X.
8. HAI LINH, B.T. and KUNG, Y. Digital Hardware Realization of Forward and Inverse Kinematics for a Five-Axis Articulated Robot Arm. *Mathematical Problems in Engineering*, 08/27, 2015, vol. 2015. pp. 1-10 ISSN 1024123X.
9. *Forward Kinematics: The Denavit-Hartenberg Convention.* Žiūrėta 2017-04-02. Prieiga per internetą: https://upcommons.upc.edu/bitstream/handle/2099.1/24573/A-Denavit%20Hartenberg.pdf?sequence=2&isAllowed=y.
10. L. Chen, D. Ojdanić, K. Michels, H.O. Peitgen. Supporting Navigated Surgery with Pan-Tilt Controlled Laser Pointer. *2011*, 2011-09-15. pp. 3.
11. Shinji Umeyama. Least-Squares Estimation of Transformation Parameters between Two Point Patterns, 1991, vol. 13, no. 4. pp. 376.
12. Freepik. *Multimedia Desktop PC Illustration.* Žiūrėta 2017-04-15. Prieiga per internetą: http://www.freepik.com/free-vector/multimedia-desktop-pc-illustration_721043.htm#term=desktop pc&page=1&position=0.

13. Arduino. *Arduino Board.* Žiūrėta 2017-03-27. Prieiga per internetą: https://www.arduino.cc/.
14. CNC LAB. *AX-12 CDS55xx Driver Board.* Žiūrėta 2017-03-22. Prieiga per internetą: http://www.cnclablb.com/product_responsive.aspx?scid=117&prid=213.
15. Robotis. *Ax-12/ Ax-12+/ Ax-12a.* Žiūrėta 2017-04-13. Prieiga per internetą: http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm.
16. SPSS analizė. *Duomenų Sklaidos Charakteristikos.* Žiūrėta 2017-04-15. Prieiga per internetą: http://www.spssanalize.lt/duomenu-sklaidos-charakteristikos/.

PRIEDAI

PAVAROS ID NUMERIO ĮRAŠYMO KODAS

```
void Dynamixel::setID(byte id, byte newId)
{
  int writeLength = 4;

  // Write standard header.
  Serial.write(0xFF);
  Serial.write(0xFF);
  Serial.write(id);
  Serial.write(writeLength);
  Serial.write(INST_WRITE);

  // Starting address of where the data is to be written.
  Serial.write(P_ID);

  // New ID.
  Serial.write(newId);

  // Check Sum.
  Serial.write((~(id + writeLength + INST_WRITE + P_ID + newId))&0xFF);
}
```

PAVAROS PADĖTIES ĮRAŠYMO KODAS

```
void Dynamixel::setPos(byte id, int pos, int vel)
{
 int writeLength = 7;
 byte pos_h = pos / 255;
 byte pos_l = pos % 255;
 byte vel_h = vel / 255;
 byte vel_l = vel % 255;

 // Write standard header.
 Serial.write(0xFF);
 Serial.write(0xFF);
 Serial.write(id);
 Serial.write(writeLength);
 Serial.write(INST_WRITE);

 // Starting address of where the data is to be written.
 Serial.write(P_GOAL_POSITION_L);

 // Write position low byte.
 Serial.write(pos_l);

 // Write position high byte.
 Serial.write(pos_h);

 // Write velocity low byte.
 Serial.write(vel_l);

 // Write velocity high byte.
 Serial.write(vel_h);

 // Check Sum.
 Serial.write((~(id + writeLength + INST_WRITE + P_GOAL_POSITION_L + pos_l +
pos_h + vel_l + vel_h))&0xFF);

 // Wait for instruction to be processed.
 delay(2);

 // Discard return data.
 while (Serial.read() >= 0){}
}
```

PAVAROS PADĖTIES UŽKLAUSOS IR GAVIMO KODAS

```
/* id = servo ID [0 - 255] */
int Dynamixel::getPos(byte id)
{
 int writeLength = 4;
 int readLength = 2;

 // Write standard header.
 Serial.write(0xFF);
 Serial.write(0xFF);
 Serial.write(id);
 Serial.write(writeLength);
 Serial.write(INST_READ);

 // Starting address of where the data is to be read.
 Serial.write(P_PRESENT_POSITION_L);

 // The length of data to read.
 Serial.write(readLength);

 // Check Sum.
 Serial.write((~(id + writeLength + INST_READ + P_PRESENT_POSITION_L + readLen
gth))&0xFF);

 // Wait for instruction to be processed.
 delay(2);

 // Discard extra data.
 for (int i = 0; i < 5; i++) Serial.read();

 // Read low byte.
 int low_Byte = Serial.read();

 // Read low byte.
 int high_byte = Serial.read();

 // Discard returned checksum.
 Serial.read();

 // Return position.
 return (int)high_byte << 8 | low_Byte;
}
```

MATLAB POZICIONAVIMO FUNKCIJOS KODAS

```
function [x,y]=lazerio_pozicionavimas(xas,yas,atstumas_iki_ekrano)

 startx_bits=512; %Pradinės nulinės x koordinatės išskaičiavimas
 startx=(startx_bits*300)/1024; %Pavertimas laipsniais
 starty_bits=512; %Pradinės nulinės y koordinatės išskaičiavimas
 starty=(starty_bits*300)/1024; %Pavertimas laipsniais
 ak=3.3; %Pirmos pavaros matmuo
 bk=5.5; %Antros pavaros matmuo

%Inversinės kinematikos lygtys kampo pasukimų skaičiavimui
x0=atstumas_iki_ekrano;
y0=yas;
fi=asin(bk/sqrt(x0.^2+y0.^2))-atan2(y0,x0);
fiL=fi*180/pi;
fi=fiL;
atstumas=ak+sqrt(x0.^2+y0.^2-bk.^2);
theta=atan2(xas,atstumas);
theta=theta*180/pi;
x_deg = startx+theta;
y_deg = starty+fi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Maksimalios ribos
if xas>50 || xas<-50
   x_deg=startx;
   disp('Maksimalios x koordinates nuo -50 iki 50 mm')
end

if yas>50 || yas<-50
   y_deg=starty;
   disp('Maksimalios y koordinates nuo -50 iki 50 mm')
end

%Posukio kampų pavertimas į kodą valdikliui
x=(x_deg*1024)/300;
x=round(x);
y=(y_deg*1024)/300;
y=round(y);
x=num2str(x);
y=num2str(y);
```

MATLAB KODAS DUOMENŲ SIUNTIMUI Į VALDIKLĮ

```
xas=0;%x koordinatë nuo -50 iki 50 mm
yas=0;%y koordinatë nuo -50 iki 50 mm
atstumas_iki_ekrano=155; %atstumas iki ekrano, mm
laser_value=0; %Lazerio pluoštelio intensyvumas

%Komunikacija su Arduino
s=instrfind;    %Ieškoti esam? nuosekli?j? komunikacij? (galima turëti tik vien?)
delete(s);      %... ir ištrinti.
s = serial('COM16'); %Pakeisti
set(s,'BaudRate',115200);
fopen(s);
pause(1);

%Kalibravimas
ijungti_kalibravima=1; % 0-išjungta, 1-?jungta
x_kal=0; %?vesti
x1_kal=0; %?vesti
y_kal=0; %?vesti
y1_kal=0; %?vesti
[x_kor,y_kor]=kalibravimas(x_kal, x1_kal, y_kal, y1_kal, xas, yas, atstumas_iki_ekrano,
ijungti_kalibravima);

%Pavar? valdymas
while true
[x_pos,y_pos]=lazerio_pozicionavimas(x_kor,y_kor,atstumas_iki_ekrano);
stringas=strcat('Servo:',x_pos,';',y_pos,';');%Komandos sudarymas
fprintf(s,'%s',stringas);%pavar? pasukimo komandos nusiuntimas
pause(0.05);
end

%Lazerio valdymas
while true
stringas=strcat('Laser:',laser_value,';');%Komandos sudarymas
fprintf(s,'%s',stringas);%lazerio intensyvumo nusiuntimas
pause(0.05);
end

%Komunikacijos pabaigimas
fclose(s);
delete(s);
clear s;
```

ARDUINO MIKROVALDIKLIO PROGRAMA

```
#include <Dynamixel.h>
#include <SoftwareSerial.h>
#include "Wire.h"
#include <math.h>

//Servo pavaros
float x = 512;
float y = 512;
//Lazeris
float laser_value = 0;

// Servo pavarų biblioteka
Dynamixel SERVO;

void setup(){

  // Servo komunikacijos inicializacija
  Serial3.begin(1000000);  // 1 MHz
  // Arduino komunikacijos inicializacija
  Serial.begin(115200);
  Serial.setTimeout(50);
  // Numatyta servo padėtis
  servo_control(512,512);
  // Numatyta lazerio būsena
  laser_control(0);
}

void loop(){
  // MatlLAB ->> Arduino duomenų perdavimas
  if (Serial.available()) {
    String str = Serial.readString();
    String command_header = str.substring(0,str.indexOf(':'));
    String command_values = str.substring(str.indexOf(':')+1,str.length());
    String values[5];
    int while_cycle = 1;

    while (command_values.indexOf(';') != -1) {
      values[while_cycle] = command_values.substring(0, command_values.indexOf(';'));
      command_values = command_values.substring(command_values.indexOf(';') + 1,
command_values.length());
      while_cycle += 1;
    }
    // Valdymas
    if (command_header == "Laser")  {
      laser_value = values[1].toFloat();
    }  else if (command_header == "Servo")  {
        x = values[1].toFloat();
        y = values[2].toFloat();
```

```
      }  else  {
          Serial.println("Komanda nezinoma");
        }
  }

  laser_control(laser_value);   // Lazerio valdymas 0 - OFF 255 - MAX
  servo_control(x, y); // Servo valdymas x ir y žingsniai
}

//servo pavarų valdymas
void servo_control (float x, float y)  {
  SERVO.setPos(0,x,150); // adresas, x - žingsniai, greitis
  SERVO.setPos(1,y,150); // adresas, y - žingsniai, greitis
}

//lazerio valdymas (TTL)
void laser_control (float power)  {
  analogWrite(9, power);
}
```

MATLAB KALIBRAVIMO FUNKCIJOS KODAS

```
function [x_kor,y_kor]=kalibravimas(x_kal, x1_kal, y_kal, y1_kal,xas, yas, atstumas_iki_ekrano, ijungti_kalibravima)

if ijungti_kalibravima>0
%-----------------------------------------------------------------------------------------------
% Pasvirimo kampas X ašimi
x=x_kal; %kalibravimo koordinat? x ašimi
x0=atstumas_iki_ekrano;
x1=x1_kal; %išmatuota koordinat? (didesn? už x)
x2=xas; %reikalinga koordinat?
theta_x=acos((x*(sqrt(x1^2*x^2+x1^2*x0^2-x^2*x0^2)+x0^2)/(x1*(x^2+x0^2))));
%pasvirimo kampas x ašimi
dx=roots([(sin(theta_x)^2*x2^2-x0^2) 2*cos(theta_x)*x2*x0^2 sin(theta_x)^2*x0^2*x2^2-x0^2*x2^2]);
if x>=0 && x1>=0
  if x2>0
    x_kor=dx(2);
  end
  if x2<=0
    x_kor=dx(1);
  end
end
if x<0 && x1<0
  if x2>0
    x_kor=dx(1);
  end
  if x2<=0
    x_kor=dx(2);
  end
end

%-----------------------------------------------------------------------------------------------
% Pasvirimo kampas Y ašimi
y=y_kal; %kalibravimo koordinat? y ašimi
x0=atstumas_iki_ekrano;
y1=y1_kal; %išmatuota koordinat? (didesn? už y)
y2=yas; %reikalinga koordinat?
theta_y=acos((y*(sqrt(y1^2*y^2+y1^2*x0^2-y^2*x0^2)+x0^2)/(y1*(y^2+x0^2))));
%pasvirimo kampas x ašimi
dy=roots([(sin(theta_y)^2*y2^2-x0^2) 2*cos(theta_y)*y2*x0^2 sin(theta_y)^2*x0^2*y2^2-x0^2*y2^2]);
if y>=0 && y1>=0
  if y2>0
    y_kor=dy(2);
  end
  if y2<=0
    y_kor=dy(1);
  end
```

```
end
if y<0 && y1<0
  if y2>0
    y_kor=dy(1);
  end
  if y2<=0
    y_kor=dy(2);
  end
end
end

if ijungti_kalibravima==0
    x_kor=xas;
    y_kor=yas;
end
```