KLAIPĖDOS UNIVERSITETAS

Jūrų technikos fakultetas

Informatikos inžinerijos katedra

ROBOTO MANIPULIATORIAUS NUOTOLINIO VALDYMO SISTEMOS KŪRIMAS ANDROID OPERACINĖJE SISTEMOJE

DEVELOPMENT OF REMOTE CONTROL SYSTEM FOR ROBOT MANIPULATOR USING ANDROID OPERATING SYSTEM

Magistro baigiamasis darbas

Autorius TII-11 gr. Aurimas Vaitekūnas

Vadovas Prof. dr. Arūnas Andziulis

Klaipėda, 2013

ANOTACIJA

Šiame darbe aprašoma roboto manipuliatoriaus prototipo nuotolinio valdymo sistema valdoma išmaniuoju telefonu. Išmaniojo telefono panaudojimas roboto manipuliatoriaus nuotoliniam valdymui suteikia visiškai naujų galimybių. Aprašomą inovatyvią nuotolinio valdymo sistemą sudaro Samsung Galaxy Nexus I9250 išmanusis telefonas su Android 4.2.1 Jelly Bean operacine sistema bei įrengtais akselerometriniu ir giroskopiniu jutiklais, sukurtas roboto manipuliatoriaus prototipas, sukurta specializuota išmaniojo telefono ir roboto manipuliatoriaus programinė įranga. Išmaniojo telefono programinė įranga sukurta taip, kad robotą manipuliatorių būtų galima valdyti tiek specialiais rankos gestais, tiek valdymo mygtukų pagalba. Specializuota roboto manipuliatoriaus programinė įranga vykdo visas gautas komandas iš išmaniojo telefono ir siunčia grįštamąją informaciją atgal į išmanųjį telefoną. Išmanusis telefonas ir robotas manipuliatorius tarpusavyje keičiasi informacija naudodami Bluetooth bevielį ryšį. Išmaniojo telefono panaudojimas nuotoliniam valdymui suteikia galimybę robotą manipuliatorių valdyti greitai ir patogiai, atliekant krovos bei kitus darbus.

RAKTINIAI ŽODŽIAI: nuotolinis valdymas, išmanusis telefonas, Android OS, robotas manipuliatorius, akselerometras, giroskopas.

SUMMARY

This work describes the robot manipulator prototype remote control system which is controlled by smartphone. Conventional remote control methods require considerable technical equipment: computer, joysticks and other accessories. Greater amount of equipment is harder to transport than working mobile work. Remotely operated robots are used for various works: working with hazardous chemicals, in the rescue operations after various natural disasters, in underground pipe installation and repair work, working on unstable surfaces and other works. Remote control techniques are used to manage industrial robots and robots that facilitate human life.

This paper describes the new robotic remote control mode, which is based on the smart phone with Android operating system. This new technology can replace conventional robot control techniques: computer, joysticks and other accessories. This work describes an innovative remote control system consists of the Samsung Galaxy Nexus I9250 smartphone with Android 4.2.1 Jelly Bean operating system and the installed accelerometer and gyroscope sensors, the robot manipulator, specialized smartphone and robot manipulator software. Smartphone software is designed so that the robot manipulator can be controlled using special hand gestures and control buttons. To identify human hand gestures are used accelerometer and gyroscope sensors. The accelerometer sensor is measuring static and dynamic acceleration. By measuring the amount of static acceleration due to gravity, smartphone software can get information about angle the smartphone is tilted at with respect to the earth. By sensing the amount of dynamic acceleration, smartphone software can get information about the way the smartphone is moving. The gyroscope sensor is measuring angular acceleration. Specialized smartphone software is using two control modes: robot manipulator control using only hand gestures and robot manipulator control using hand gestures and control buttons. Specialized smartphone software programmed to recognize eleven different hand gestures of operator. Using only hand gestures control mode, smartphone software recognize all eleven hand gestures to control robot manipulator. When enabled other control mode, smartphone software using five hand gestures and buttons to control robot manipulator. To transfer information between smartphone and robot manipulator used Bluetooth.

Tests carried out by the robot manipulator has been successfully operated remotely using the control buttons and human hand gesture recognition. Smartphone software successfully send management commands to robot manipulator, when operator made any of eleven hand gestures. Robot manipulator successfully accomplished all received control commands and send back information to smartphone.

KEYWORDS: remote control, smartphone, Android OS, robot manipulator, accelerometer, gyroscope.

PAVEIKSLŲ SĄRAŠAS

30 pav. Bluetooth įjungimo pranešimas išmaniojo telefono ekrane;

31 pav. Roboto manipuliatoriaus judėjimo sustabdymo gestas;

32 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu stumiamas į priekį (tolyn nuo operatoriaus);

33 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę;

34 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu stumiamas į priekį (tolyn nuo operatoriaus);

35 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę;

36 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus);

37 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas žemyn;

38 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus);

39 pav. Išmanusis telefonas sukamuoju judesiu pasukamas aukštyn;

40 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judėjimu pastumiamas į dešinę pusę;

41 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę;

42 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judėjimu pastumiamas į kairę pusę;

43 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę;

44 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę;

45 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę;

46 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas žemyn;

47 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas žemyn;

48 pav. Išmanusis telefonas sukamuoju judesiu pasukamas aukštyn;

49 pav. Išmanusis telefonas sukamuoju judesiu pasukamas aukštyn;

50 pav. Gestų atlikimo vidutinė trukmė, s

51 pav. Sukurtos specializuotos išmaniojo telefono programinės įrangos vaizdas išmaniojo telefono ekrane;

52 pav. Roboto manipuliatoriaus krano sukimasis pagal laikrodžio rodyklę;

53 pav. Vidutinis gestų atpažinimo patikimumas.

TERMINŲ IR SANTRAUKŲ ŽODYNĖLIS

A

Akselerometrinis jutiklis (*angl. Accelerometer sensor*) – jutiklis matuojantis statinę ir dinaminę įrenginio akseleraciją.

Android OS – Android operacinė sistema.

Android operacinė sistema (*angl. Android operating system*) – atviro kodo operacinė sistema, daugiausia naudojama išmaniuosiuose telefonuose bei planšetiniuose kompiuteriuose.

Arduino (*angl. Arduino*) – tai atviro kodo prototipavimo platforma.

AX.25 duomenų perdavimo protokolas (*angl. AX.25 data link layer protocol*) – duomenų perdavimo protokolas skirtas naudoti neprofesionaliems radijo operatoriams.

B

Bluetooth (*angl. Bluetooth*) – tai bevielio ryšio technologija.

D

Duomenų perdavimo protokolas (*angl. Transmission Control Protocol, TCP*) – tai vienas iš pagrindinių protokolų, esančių internetinių protokolų rinkinyje.

E

Ethernet (*angl. Ethernet*) – tai kompiuterinių tinklų technologija, skirta vietiniams kompiuterių tinklams.

F

FriendlyARM mini2440 (*angl. FriendlyARM mini2440*) – tai kompiuteris sumontuotas vienoje plokštėje, turintis Samsung S3C2440 mikroprocesorių bei naudojantis Linux operacinę sistemą.

I

Infraraudonųjų spindulių jutiklis (*angl. Infrared sensor*) – tai jutiklis naudojantis infraraudonuosius spindulius objektams aptikti.

Inklinometras (*angl. Inclinometer*) – tai įrenginys matuojantis objekto pakrypimo kampą, atsižvelgiant į žemės gravitaciją.

Interneto protokolas (*angl. Internet Protocol, IP*) – tai taisyklių visuma, apibrėžianti duomenų mainų būdą tarp dviejų kompiuterinių sistemų.

L

Linux (*angl. Linux*) – tai atvirojo kodo operacinė sistema.

M

Maršrutizatorius (*angl. Router*) – tai kompiuterių tinklus jungiantis įrenginys, atliekantis duomenų maršrutizavimo funkciją.

Microsoft Visual Basic (*angl. Microsoft Visual Basic*) – tai programinės įrangos kūrimo aplinka.

Mikrovaldiklis (*angl. Microcontroller*) – tai mažas kompiuteris esantis integruotoje mikroschemoje, turintis procesorių, atmintį, įvesties ir išvesties programuojamus išorinius įrenginius.

O

Orientacijos jutiklis (*angl. Orientation sensor*) – tai jutiklis matuojantis įrenginio orientaciją erdvėje. Pavyzdžiui įrenginio pasisukimą iš horizontalios į vertikalia padėtį

P

Posūkio kampo jutiklis (*angl. Rotary encoder*) – tai įrenginys matuojantis ašies pasisukimo kampą.

V

Val3 (*angl. Val3*) - tai specializuota programavimo kalba skirta robotų programavimui.

Vartotojo datagramų protokolas (*angl. User Datagram Protocol, UDP*) – tai TCP/IP naudojamas perdavimo protokolas. Šis protokolas yra alternatyva TCP protokolui.

VxWorks operacinė sistema (*angl. VxWorks operating system*) – tai realaus laiko operacinė sistema, turinti daugiaprogramino režimo branduolį, įvairių užduočių pirmumą, greitą reagavimą į procesų pertraukimus, paprastą naudojimą bei modernią failų sistemą.

W

Wi-Fi (*angl. Wi-Fi*) – tai bevielio ryšio technologija.

X

XBee (*angl. XBee*) – tai bevielio ryšio technologijos modulis.

TURINYS

ĮVADAS

Nuotoliniu būdu valdomi robotai – labai svarbi šiuolaikinio informacinių technologijų mokslo sritis [16]. Mokslo laimėjimai šioje srityje suteikia galimybę operatoriui valdyti robotą per atstumą žmogui nepasiekiamoje bei žmogaus sveikatai pavojingoje aplinkoje [16]. Nuotoliniu būdu valdomi robotai naudojami įvairiems darbams atlikti: dirbant su pavojingomis cheminėmis medžiagomis, atliekant gelbėjimo darbus po įvairių gamtos katastrofų, atliekant požeminių vamzdžių klojimo ir remonto darbus [3], dirbant ant nestabilių, galinčių įgriūti paviršių bei atliekant kitus darbus [16].

Sparčiai besivystant ir augant pasaulio ekonomikai, didėja ir pramonės gamybos apimtys. To pasėkoje daugėja ir nuotoliniu būdu valdomų pramoninių robotų. Tobulėjant robotų technologijoms neišvengiamai keičiasi ir žmones supanti aplinka. Žmonės po truputį pradeda nudoti įvairius nuotoliniu būdu valdomus robotus, kurie palengviną įprastą žmogaus kasdienybę [10], [12]. Daugėjant robotų kurie naudojami ne tik pramonėje ar pavojingiems darbams atlikti, taip pat atsiranda ir naujų nuotolinių valdymo būdų poreikis, kurie valdymą padarytų patogesnį ir lengvai prieinamą kiekvienam robotus naudojančiam žmogui [1], [14].

Šiame darbe aprašomas naujas robotų nuotolinio valdymo būdas – grindžiamas išmaniuoju telefonu su Android operacine sistema (Android OS). Ši nauja technologija gali pakeisti įprastus robotų valdymo metodus – roboto valdymą naudojant valdymo svirtį, stacionarų ar nešiojamąjį kompiuterį. Šios naujos technologijos dėka, visas roboto valdymo įrenginys telpa žmogaus delne. Tai ypač patogu, nes valdydamas robotą, operatorius gali laisvai judėti aplinkoje. Dėka naujausių išmaniojo telefono akselerometrinio ir giroskopinio jutiklių ir pažangios Android OS, operatorius gali valdyti robotą ne tik spausdamas valdymo mygtukus išmaniojo telefono ekrane, bet ir atlikdamas tam tikrus rankos gestus laikydamas išmanųjį telefoną delne. Atlikti gestai yra atpažįstami išmaniojo telefono programinės įrangos ir paverčiami roboto valdymo komandomis.

Darbo tikslas:

Sukurti ir ištirti roboto manipuliatoriaus nuotolinio valdymo sistemos prototipą Android operacinėje sistemoje su įdiegtais akselerometriniu ir giroskopiniu jutikliais.

Darbo uždaviniai:

1. Išanalizuoti robotų nuotolinio valdymo ypatumų mokslinę literatūrą.
2. Sukurti roboto manipuliatoriaus nuotolinio valdymo išmaniuoju telefonu sistemos modelį.
3. Sukurti roboto manipuliatoriaus prototipo programinę įrangą bei specializuotą programinę įrangą išmaniajam telefonui Android OS pagrindu.
4. Atlikti sukurtos nuotolinio valdymo sistemos verifikavimą.

I. ANALITINĖ DALIS

1.1. Robotų nuotolinis valdymas

1.1.1. Industrinių robotų manipuliatorių nuotolinis valdymas

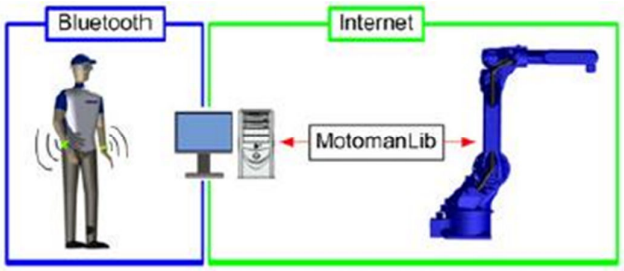
Šiuolaikinėje industrinėje gamyboje be įprastos žmogaus darbo jėgos, dažnai naudojami ir industriniai robotai manipuliatoriai [20]. Gamyboje naudojamų robotų skaičius svyruoja nuo kelių iki keliasdešimties vienu metu veikiančių industrinių robotų manipuliatorių ir daugiau, priklausomai nuo gamybos srities. Visi šie industriniai robotai manipuliatoriai gali atlikti iš anksto užprogramuotas užduotis arba būti valdomi operatoriaus nuotoliniu būdu priklausomai nuo gamybos poreikių. Dažniausiai industrinių robotų programavimas ar tiesiog valdymas pagal poreikius yra vykdomas naudojant nešiojamą valdymo įrenginį. Tai įrenginys, kuris suteikia galimybę nuotoliniu būdu programuoti, bei valdyti roboto judesius (1 pav.).

1 pav. Tipiškas nešiojamas industrinių robotų programavimo, valdymo įrenginys [1]

Tipiškas robotų valdymo įrenginys susideda iš klaviatūros valdymo komandoms įvesti. Klaviatūros dizainas kinta priklausomai nuo valdomo roboto rūšies. Taip pat įrenginyje yra įmontuotas ekranas, kuriama pateikiama informacija apie vykdomas komandas ir galiausiai saugumo sumetimais yra įmontuojamas didelis sustabdymo mygtukas, kad bet kuriuo metu būtų galima greitai išjungti valdomą industrinį robotą. Robotų programavimas bei nuotolinis valdymas reikalauja daug techninių žinių, be to roboto programavimo procesas naudojant šį nuotolinio valdymo įrenginį užtrunka ganėtinai ilgai [20]. Atsižvelgdami į tai mokslininkai nuolatos ieško naujų, intuityvių industrinių robotų programavimo bei valdymo būdų [20].

Pedro Neto ir kiti jo komandos mokslininkai šią problemą pabandė išspręsti, industrinių robotų programavimui ir valdymui, pritaikydami akselerometrinį jutiklį [20]. Savo eksperimente mokslininkai panaudojo du trijų ašių akselerometrinius bevielius jutiklius. Jutikliai buvo pritaisyti prie operatoriaus riešų. Tokiu būdu dviejų akselerometrinių jutiklių pagalba buvo fiksuojami rankų judesiai. Kairiosios rankos akselerometras buvo naudojamas tik sistemai įjungti ir išjungti, o visas roboto valdymo komandas fiksavo dešiniosios rankos akselerometrinis jutiklis [20]. Duomenys apie

rankų judesius iš jutiklių į kompiuterį buvo perduodami *Bluetooth* bevieliu ryšiu (2 pav.). Gauti duomenys iš jutiklių kompiuteryje buvo analizuojami panaudojant dirbtinį neuroninį tinklą. Dirbtinis neuroninis tinklas išanalizavęs duomenis nustatydavo kokį gestą atliko operatorius ir tokiu būdu programinė įranga žinodavo kokią valdymo komandą reikia perduoti į robotą. Tarp kompiuterio ir roboto duomenys buvo perduodami *Ethernet* tinklo ryšiu. Tam, kad duomenys sėkmingai būtų perduoti iš kompiuterio į robotą *Ethernet* ryšiu, mokslininkai sukūrė duomenų perdavimo biblioteką „*MotomanLib*“ [20].



2 pav. Duomenų perdavimo schema tarp operatoriaus, kompiuterio ir industrinio roboto [20]

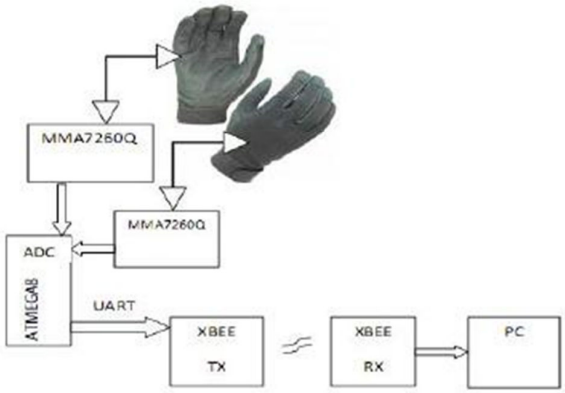
Atlikti tyrimai [20] parodė, kad akselerometro panaudojimas valdant industrinius robotus yra išties geras sprendimas. Lyginant su įprastu valdymo įrenginiu (1 pav.), akselerometriniu jutikliu paremtas valdymo būdas yra intuityvesnis ir paprastesnis. Net ir daug techninių žinių neturintis žmogus, panaudojus akselerometrinius jutiklius, gali valdyti robotą ganėtinai lengvai. Be to akselerometru paremtas valdymo metodas yra daug pigesnis. Sistemos gestų atpažinimo tikslumas yra 92% [20]. Tai reiškia, kad dalis operatoriaus komandų nebuvo įvykdytos. Pedro Neto, J. Noberto Pires ir A. Paulo Moreira ateityje planuoja padidinti gestų atpažinimo tikslumą, panaudodami daugiau akselerometrinių jutiklių pritvirtintų prie operatoriaus rankų [20].

Pedro Neto ir jo komandos mokslininkai akselerometrinius jutiklius pritvirtino prie žmogaus rankų riešų. Kitas galimas akselerometrinio jutiklio tvirtinimo metodas, kurį panaudojo Anala Pandit ir kiti jos komandos mokslininkai – akselerometrinio jutiklio pritaisymas prie pirštinių, kurias mūvi žmogus [23]. Anala Pandit, Dhairya Dand, Sisil Mehta, Shashank Sabesan, Ankit Daftery aprašė žmogaus ir kompiuterio sąsaja, t.y. kompiuterių, robotų ir kitų įrenginių valdymą panaudojant žmogaus rankos judesius, kuriuos fiksuoja prie pirštinės pritvirtintas akselerometrinis jutiklis. Anala Pandit, Dhairya Dand, Sisil Mehta, Shashank Sabesan, Ankit Daftery tikslas buvo sukurti naują, intuityvę, mobilę valdymo sistemą, kuri pakeistų įprastus kompiuterių ir kitų įrenginių valdymo būdus. Mokslininkų pasiūlytas sprendimas susideda iš dviejų pirštinių, kurias mūvi operatorius. Pirmoje pirštinėje yra įtaisytas 3 ašių akselerometras (MMA7260Q), jungiklis bei *XBee* bevielio ryšio modulis (1 priedas, 1 pav.). Antroje pirštinėje taip pat yra įtaisytas 3 ašių akselerometrinis jutiklis (MMA7260Q), jungiklis ir mikrovaldiklis atmega8 [23].

Įjungus žmogaus rankų gestų atpažinimą - vykdomi tokie veiksmai:

1. Akselerometriniai jutikliai siunčia analoginį signalą į mikrovaldiklį apie rankų padėtį erdvėje (3 pav.).
2. Atmega8 mikrovaldiklis gautą analoginį signalą pakeičia į 8 bitų skaitmeninį signalą.
3. Iš mikrovaldiklio per pirštinėse įtaisytą *XBee* bevielio ryši modulį informacija yra siunčiama į prie kompiuterio prijungtą *XBee* bevielio ryšio modulį.

Prie kompiuterio *XBee* modulis yra prijungtas USB jungtimi. Informacija iš mikrovaldiklio į kompiuterį siunčiama 32 bitų dydžio paketais. Paketas sudarytas iš 4 elementų po 8 bitus: X, Y, Z koordinates ir jungiklio padėties informacija.



3 pav. Bendra sistemos duomenų srautų diagrama [23]

Kompiuteryje į kurį siunčiama informacija iš akselerometrinių jutiklių įdiegta specializuota programinė įranga. Specializuotos programinės įrangos paskirtis – gautus duomenis iš akselerometrinių jutiklių paversti į reikiamas kompiuterio valdymo komandas. Šiame konkrečiame eksperimente pirštinės su akselerometriniais jutikliais buvo naudojamos kompiuterio valdymui vietoj įprastos kompiuterio pelės [23].

Kompiuterio pelės valdymas yra tik vienas iš daugelio šios technologijos pritaikymo būdų. Anot eksperimento autorių aptartą technologiją galima panaudoti dirbant su kompiuterine grafika, robotų valdymui, įvairių kitų įrenginių nuotoliniam valdymui. Taip pat šią technologiją galima pritaikyti žmonėms su negalia ir palengvinti negalią turinčių žmonių gyvenimą [23].

Atlikti tyrimai [23] patvirtina, kad akselerometriniai jutikliai gali būti panaudoti nuotoliniam industrinių robotų valdymui. Ateityje planuojama tobulinti šią technologiją ir jos galimybes dar labiau praplėsti [23]. Tolimesnis šios technologijos vystymas tik parodo, kad akselerometriniai jutikliai turi išties didelį pritaikymo potencialą valdant įrenginius nuotoliniu būdu.

Aptartuose industrinių robotų valdymo metoduose tiek operatorius tiek robotas, kurį valdo operatorius yra netoli vienas nuo kito, atstumas mažiau kaip 100 metrų. Žemiau nagrinėjamas atvejis, kada operatorių ir industrinį robotą skiria didelis atstumas, keliasdešimt ir daugiau

kilometrų. Mei Xuhong savo paskelbtame straipsnyje aprašo nuotolinę industrinio roboto valdymo sistemą kuri susideda iš tokių pagrindinių elementų: TX90 industrinis robotas, valdiklis-serveris ir kliento terminalas [31]. Atliktame eksperimente buvo panaudotas TX90 industrinis robotas, tačiau realiai galima naudoti ir kito modelio industrinį robotą. Robotas su valdikliu-serveriu yra sujungtas kabeliu, o valdiklį-serverį ir kliento terminalą jungia kompiuterinis tinklas (1 priedas, 2 pav.). TX90 industrinio roboto valdiklis naudoja *VxWorks* OS [31]. Valdiklio-serverio programinei įrangai sukurti buvo naudojama *VAL3* programavimo kalba. Kliento terminalui buvo panaudota FriendlyARM mini2440 platforma [31].

Kliento terminalo ir valdiklio-serverio susijungimas vyksta dviejų rankos paspaudimų principu (1 priedas, 3 ir 4 pav.). Jeigu kliento terminalas prisijungia sėkmingai, serveris į terminalą siunčia patvirtinimo pranešimą ir laukia kol terminalas atsakys. Kai terminalas atsako, serveris siunčia starto pranešimą į terminalą. Terminalas gavęs starto pranešimą siunčia į serverį roboto valdymo komandas [31].

Atlikti eksperimentai parodė, kad industrinį robotą galima valdyti nuotoliniu būdu panaudojant valdiklį-serverį ir nutolusį kliento terminalą. Ką tik aptarta nuotolinio valdymo sistema iš esmės skiriasi nuo ankščiau aptartų valdymo metodų, nes šiuo atveju naudojamas ne akselerometriniais jutikliais pagrįstas valdymas, o vietoj to, naudojamas kliento terminalas FriendlyARM mini2440.

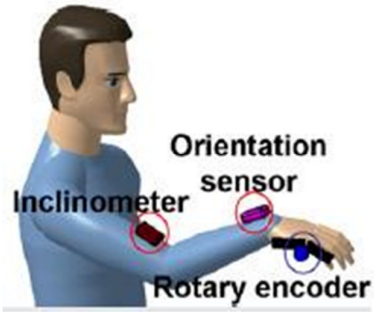
1.1.2. Statybai ir aplinkos tyrinėjimui naudojamų robotų nuotolinis valdymas

Šiuolaikiniame pasaulyje robotų pritaikymas išties platus. Be pramonės, nuotoliniu būdu valdomi robotai taip pat gali būti naudojami statybai bei aplinkos tyrinėjimui. Statybiniai robotai tai iš esmės žmonių valdomi statybiniai įrenginiai kaip pavyzdžiui ekskavatoriai, kurie buvo modifikuoti ir taip paversti nuotoliniu būdu valdomais robotais [9].

Dungmok Kim ir kiti mokslininkai aprašė metodą, kaip galima valdyti nedidelį ekskavatorių nuotoliniu būdu panaudojant žmogaus rankų judesius [13]. Nuotolinis ekskavatoriaus valdymas labai praverčia, kada reikia dirbti pavojingose vietovėse ar ant nestabilių paviršių. Dungmok Kim ir kitų mokslininkų pasiūlyta nuotolinio valdymo sistema susideda iš dviejų pagrindinių dalių. Pirmoji dalis – operatorius ant kurio rankos uždėti sensoriai, antroji – modifikuotas ekskavatorius.

Šiame eksperimente, kurį atliko mokslininkai, rankų judesiams atpažinti buvo panaudoti trys jutikliai: orientacijos jutiklis, inklinometras ir posūkio kampo jutiklis. Visi trys jutikliai išdėstyti ant dešiniosios operatoriaus rankos [13]. Posūkio kampo jutiklis pritvirtintas prie delno, orientacijos jutiklis pritvirtintas prie riešo išorinės pusės, o inklinometras pritvirtintas šiek tiek aukščiau alkūnės (4 pav.). Visi trys jutikliai veikia atskirai vienas nuo kito. Kompiuteris duomenis iš visų jutiklių nuskaito kas 100 ms. Atitinkami rankos judesiai valdo keturis ekskavatoriaus judesius:

ekskavatoriaus strėlės sukamąjį judesį, ekskavatoriaus strėlės pirmo ir antro šarnyrų judesius ir ekskavatoriaus kaušo judesius (1 priedas, 5 pav.).



4 pav. Operatoriaus ranka ant kurios pritvirtinti inklinometro, orientacijos bei posūkio kampo jutikliai [13]

Vienu metu yra valdoma tik viena ekskavatoriaus dalis. Pavyzdžiui jeigu valdomas ekskavatoriaus kaušas, tai likusios ekskavatoriaus dalys tuo metu nejuda. Visa valdymo sistema turi įjungimo ir išjungimo mechanizmą. Ekskavatoriaus valdymas įjungiamas arba išjungiamas pasukus riešą apie x ašį (1 priedas, 6 pav.). Paties ekskavatoriaus hidraulinė sistema yra modifikuota pridedant keturis papildomus vožtuvus, tam kad ekskavatorių būtų galima valdyti nuotoliniu būdu.

Ekskavatoriaus valdymo proceso eiga:

1. Kompiuteris nuskaito duomenis iš jutiklių.
2. Išanalizavęs gautus duomenis, kompiuteris per bevielį *Bluetooth* ryšį perduoda valdymo komandą į ekskavatoriuje esantį kompiuterį.
3. Ekskavatoriaus kompiuteris gavęs duomenis, atitinkamai įjungia vieną iš keturių valdymo vožtuvų, kurių pagalba valdomi ekskavatoriaus strėlės judesiai [13].

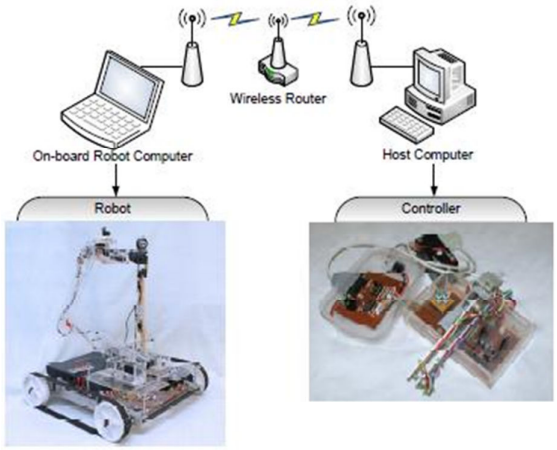
Atlikti eksperimentai parodė, kad ekskavatoriaus strėlę galima valdyti pasitelkiant žmogaus rankos judesius. Atlikti eksperimentai parodė, kad žmogaus rankos judesiams nustatyti gali būti panaudoti inklinometro, orientacijos bei kampo padėties jutikliai. Eksperimentų metu buvo nustatyta, kad ekskavatoriaus kaušas gali judėti horizontalia trajektorija su 80 mm paklaida. Šiame eksperimente buvo valdomi tik keturi ekskavatoriaus judėjimo laisvės laipsniai. Ateityje mokslininkai planuoja tobulinti šią technologiją, padidindami valdomų ekskavatoriaus laisvės laipsnių skaičių, bei patobulinti pačia valdymo sistemą.

Aptartas ekskavatoriaus nuotolinio valdymo metodas yra tik vienas iš galimų variantų. Mokslininkai T. Sasaki ir K. Kawashima pasirinko visai kitokį problemos sprendimo variantą. Mokslininkų sukurta nuotolinio valdymo sistema susideda iš pneumatinės roboto rankos (ekskavatoriaus strėlė) valdomos pneumatinių dirbtinių raumenų, nešiojamo kompiuterio, dviejų ekskavatoriaus valdymo svirčių, kameros, ekskavatoriaus valdymo dėžės, kurioje yra kompiuteris, maitinimo šaltinis ir bevielio ryšio plokštė [24]. Pneumatinių dirbtinių raumenų pagalba iš viso yra valdomi šeši ekskavatoriaus judėjimo laisvės laipsniai.

Šiame eksperimente ekskavatoriaus valdymui naudojami ne žmogaus rankos judesiai, bet dvi ekskavatoriaus valdymo svirtys, kurios yra prijungtos prie nešiojamo kompiuterio. Duomenis iš valdymo svirčių nuskaito nešiojamame kompiuteryje įdiegta specializuota programinė įranga. Gavusi duomenis iš valdymo svirčių specializuota programa per bevielį *Wi-Fi* (*IEEE* 802.11b) ryšį siunčia valdymo komandą į ekskavatoriuje įmontuotą kompiuterį. Ekskavatoriuje esantis valdymo kompiuteris gavęs duomenis atitinkamai siunčia valdymo komandas į pneumatinius dirbtinius raumenis, kurių pagalba valdomas ekskavatoriaus judėjimas (1 priedas, 7 ir 8 pav.). Taip pat į operatoriaus nešiojamą kompiuterį yra siunčiamas vaizdas iš ekskavatoriuje įmontuotos kameros. Kameros vaizdas padeda operatoriui geriau susiorientuoti, kokioje padėtyje yra valdomas ekskavatorius.

Atlikus eksperimentus nustatyta, kad nuotoliniu būdu valdomo ekskavatoriaus darbo našumas lyginant su tiesiogiai operatoriaus valdomu ekskavatoriumi siekia daugiau kaip 50% [24]. Atlikti eksperimentai parodė, kad nuotoliniu būdu valdomi robotai pavojingose vietovėse gali išties labai pasitarnauti ir padėti apsaugoti pavojingomis sąlygomis dirbančių žmonių sveikatą.

Nuotoliniu būdu valdomi robotai taip pat dažnai naudojami ir aplinkos tyrinėjimui, paieškos ar gelbėjimo darbams [16]. Šiems darbams skirti robotai dažniausiai valdomi iš didelio atstumo, kada tiesioginis roboto stebėjimas yra negalimas. Taip pat nuotoliniu būdu valdomi robotai naudojami ir kariniams tikslams. T. J. Leow ir K. S. Sim sukūrė nuotoliniu būdu valdomą robotą manipuliatorių, kuris gali būti panaudotas gelbėjimo, aplinkos tyrinėjimo darbams atlikti [16]. Mokslininkų sukurtą nuotolinio roboto manipuliatoriaus valdymo sistemą sudaro tokios pagrindinės dalys: a. robotas manipuliatorius ir b. roboto valdymo įranga, kuria naudojasi operatorius (5 pav.).



5 pav. Roboto manipuliatoriaus nuotolinio valdymo sistema [16]

a. Robotą manipuliatorių sudaro: roboto manipuliatoriaus ranka, mobilus roboto pagrindas ant kurio pritvirtinta manipuliatoriaus ranka, kompiuteris ir kita elektronika pritvirtinta prie roboto.

b. Roboto valdymo įrangą sudaro: valdymo kompiuteris, roboto valdymo mechanizmas.

Robotas manipuliatorius ir roboto valdymo įranga tarpusavyje bendrauja bevieliu *Wi-Fi* (*IEEE* 802.11) ryšiu.

Valdymo kompiuteris tarnauja kaip komunikacinis maršrutizatorius, taip pat kaip pagrindinis valdymo įrankis operatoriui, kurio pagalba yra valdomas bei stebimas robotas manipuliatorius. Prie valdymo kompiuterio yra prijungtas roboto manipuliatoriaus valdymo įtaisas. Šis valdymo įtaisas tai roboto manipuliatoriaus rankos sumažinta kopija. Operatorius valdydamas šią sumažintą roboto ranką valdo tikrą roboto manipuliatoriaus ranką [16]:

1. Sumažintoje manipuliatoriaus rankoje yra įtaisyti jutikliai, kurie fiksuoja rankos judesius. Duomenys iš šių daviklių patenka į valdymo kompiuteryje įdiegta programą, kuri apdoroja gautą informaciją.
2. Valdymo kompiuterio programa išanalizavusi duomenis perduoda valdymo komandą per *Wi-Fi* bevielį ryšį į roboto manipuliatoriaus kompiuterį.
3. Roboto manipuliatoriaus kompiuteris gavęs reikiamą komandą vykdo roboto judesių valdymą. Tikra roboto manipuliatoriaus ranka tiksliai atkartoja sumažintos roboto rankos judesius.

Roboto ranka ir kamera yra sumontuotos ant keturiais varikliais varomos vikšrinės platformos. Toks platformos modelis pasirinktas neatsitiktinai, o tam kad būtų didesnis roboto pravažumas [16].

Programinė įranga kuri naudojama valdant robotą buvo parašyta *Microsoft Visual Basic* programavimo kalba. Programa buvo sukurta naudojant *Microsoft Visual Basic 2008 Express Edition* programavimo platformą [16]. Valdymo kompiuteris tarnauja kaip komunikacinis serveris, kuris transliuoja informaciją į robotą bei gauna duomenis iš roboto. Valdymo kompiuteris nuolatos gauna vaizdą iš roboto kameros. Kameros pagalba operatorius gali valdyti robotą manipuliatorių nuotoliniu būdu, net ir tada, kada vizualiai akimis nebemato roboto manipuliatoriaus.

Aprašyta ir išbandyta nuotolinio valdymo sistema yra vienas iš galimų roboto manipuliatoriaus valdymo būdų. Sistema iš esmės yra tinkama ten, kur nėra reikalingas operatoriaus mobilumas, kadangi norint operatoriui pakeisti darbo vietą, reikia pernešti ne tik roboto valdymo kompiuterį bet ir sumažintą roboto rankos kopiją, kurios pagalba valdoma roboto manipuliatoriaus ranka. Ten kur operatoriaus mobilumas tikrai nėra būtinas, pasiūlytas technologinis sprendimas išties yra geras, kadangi roboto valdymui panaudota sumažinta roboto rankos kopija suteikia papildomo aiškumo operatoriui.

Kita mokslininkų grupė sukūrė nuotoliniu būdu valdomą paieškos ir gelbėjimo roboto manipuliatoriaus prototipą. Robotas manipuliatorius pritaikytas dirbti pavojingose vietovėse po įvairių žemės katastrofų [30].

Patį robotą manipuliatorių sudaro vikšrinė platforma, kuri labai padidina roboto pravažumą sudėtingomis sąlygomis. Ant platformos pritaisytas šešių laisvės laipsnių manipuliatorius, kuris gali stabiliai suimti ir pernešti daiktus. Manipuliatoriaus ranka objektus gali pasiekti aplink robotą apytiksliai 1,8 metro atstumu [30]. Manipuliatorius gali pakelti svorį iki 57 kg [30]. Taip pat robotas manipuliatorius turi tris kameras, kurios stebi aplinką. Galiausiai prie roboto pritaisytas mikrofonas, kuris perduodą aplinkos garsą į operatoriaus kompiuterį (1 priedas, 9 pav.).

Roboto programinė įranga suprojektuota taip, kad tinka ne tik šiam konkrečiam robotui, bet taip pat nesunkiai gali būti pritaikyta ir kitiems panašaus tipo robotams manipuliatoriams. Programinė įranga sukurta taip, kad būtų galima panaudoti jos modulius kituose robotuose. Ant roboto yra pritaisytas nešiojamas kompiuteris, kuris tarnauja kaip serveris. Prie nešiojamo kompiuterio prijungta roboto valdymo plokštė, bei kameros. Roboto nešiojamas kompiuteris su operatoriaus valdymo kompiuteriu bendrauja bevieliu *Wi-Fi* (*IEEE* 802.11a) ryšiu. Operatoriaus valdymo kompiuteryje yra įdiegta *Linux* OS. Kaip komunikaciniai protokolai naudojami *TCP* ir *UDP* protokolai [30]. Operatoriaus valdymo kompiuteryje įdiegta programinė įranga, kurios pagalba valdomas robotas manipuliatorius. Programos lange, kurį mato operatorius, pateikiami duomenys apie roboto sensorius ir vaizdas iš kamerų (1 priedas, 10 pav.). Robotas manipuliatorius valdomas mygtukais esančiais programiniame lange, bei panaudojant valdymo svirtį. Šiuo atveju valdymo svirtis naudojama roboto manipuliatoriaus judėjimui valdyti.

Išanalizavus statybai ir aplinkos tyrinėjimui naudojamų robotų nuotolinio valdymo technologijas, matyti, kad mokslininkai robotų nuotoliniam valdymui naudoja pačius įvairiausius metodus. Dažniausiai tokio tipo robotai valdomi pasitelkiant operatoriaus valdymo kompiuterį su specializuota programine įranga, bei panaudojant vieną ar dvi valdymo svirtis. Taip pat mokslininkai siekdami supaprastinti roboto valdymą tiek, kad jį nesunkiai galėtų valdyti net ir neapmokytas žmogus, bando panaudoti žmogaus rankų judesius. Toks sprendimo variantas roboto valdymui suteikia paprastumo ir patogumo. Be valdymo svirčių ar operatoriaus rankų judesių, manipuliatoriaus rankai valdyti dar naudojama ir sumažinta manipuliatoriaus rankos kopija. Šiuo atveju operatoriui valdant sumažintą manipuliatoriaus rankos kopija yra valdoma ir tikroji manipuliatoriaus ranka. Toks technologinis sprendimas suteikia operatoriui papildomo aiškumo, kadangi manipuliatoriaus ranka tiksliai atkartoja sumažintos rankos veiksmus. Aptarti mokslininkų nagrinėjami nuotolinio robotų valdymo metodai nuolatos yra tobulinami, todėl ateityje tūrėtų parodyti dar didesnį funkcionalumą.

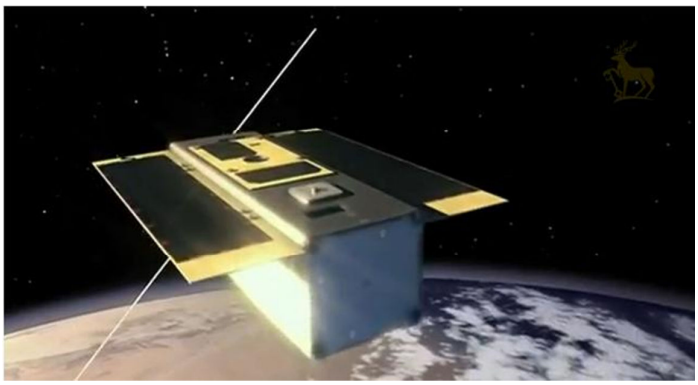
1.1.3. Nuotolinis robotų valdymas išmaniuoju telefonu

Per paskutiniuosius kelerius metus išmaniųjų telefonų technologijos labai stipriai pasistūmėjo į priekį. Įprastos telefono funkcijos kaip skambinimas ar trumpųjų žinučių rašymas, tapo tik

keliomis iš daugybės įvairiausių funkcijų, kurias gali atlikti išmanusis telefonas. Naujausiuose išmaniuosiuose telefonuose įdiegti net keturių branduolių procesoriai ir tai tikrai ne galutinis procesorių skaičius. Dėka kelėtos procesorių veikiančių vienu metu, išmaniojo telefono skaičiavimo galimybės pasidarė labai didelės. Naujausiuose išmaniuosiuose telefonuose įdiegti įvairūs jutikliai kaip akselerometras, giroskopas, barometras ir kt. Mikro-eletro-mechaniniai sensoriai (MEMS) išmaniajam telefonui suteikia visiškai naujų telefono pritaikymo galimybių.

Išmaniųjų telefonų pritaikymo galimybės išties labai plačios. Pirmiausia išmanusis telefonas tapo asmeniniu žmogaus pagalbininku kasdienėje žmogaus veikloje. Išmanieji telefonai naudojami namuose esančių įrenginių kaip televizorius ar muzikos grotuvas nuotoliniam valdymui. Taip pat išmanieji telefonai pradėti naudoti ir nuotoliniam robotų valdymui ir net panaudoti mažų dirbtinų žemės palydovų gamybai [28].

2013 metų vasario 25d. į kosmosą buvo iškeltas pirmasis nedidelis dirbtinis žemės palydovas, kurio pagrindas išmanusis telefonas su *Android* OS [28]. Dirbtinį žemės palydovą sukūrė mokslininkai iš Surrey universiteto. Dirbtinis palydovas sveria vos 4,3 kg, o jo dydis yra vos 10cm x 30cm. Palydovui kurti buvo panaudotas Google Nexus One išmanusis telefonas su *Android* OS (6 pav.). Dirbtinis palydovas valdomas iš Surrey universiteto kosminio centro. Dirbtiniame palydove įtaisytas radijo bangų įtaisas naudojantis AX.25 duomenų perdavimo protokolą [28]. Radijo bangų įtaisas naudoja 437,568 MHz radijo dažnį [28].



6 pav. Pirmasis dirbtinis žemės palydovas, kurio pagrindas išmanusis telefonas [28]

Prie išmaniojo telefono buvo pajungta kita reikalinga įranga, būtina palydovo funkcionavimui. Visas įrenginys buvo išbandomas įvairiausiomis sąlygomis, siekiant įsitikinti ar išmanusis telefonas sugebės tinkamai veikti 785km aukštyje [28]. Išmaniajame telefone yra įdiegtą daugybė įvairių programų, kurios valdys palydovo darbą.

Išmaniojo telefono panaudojimas kuriant miniatiūrinį dirbtinį žemės palydovą tik įrodo, kad išmanieji telefonai puikiai tinka kuriant miniatiūrinius dirbtinius žemės palydovus bei kitą robotų techniką.

Myeongjin Song, Baul Kim, Yunji Ryu, Yongdug Kim, Sangwook Kim sukūrė nuotolinio roboto valdymo sistemą grindžiamą išmaniuoju telefonu [26]. Ši sistema yra skirta universaliems robotams. Konkrečiame eksperimente buvo naudojamas robotas humanoidas. Mokslininkų atliktame darbe išmanusis telefonas naudojamas kaip pagrindinis roboto valdymo įrankis, kuris pakeičia įprastą stacionarų ar nešiojamą kompiuterį. Išmanusis telefonas su *Android* OS gali puikiai valdyti robotą nuotoliniu būdu, kadangi telefone yra įdiegti naudingi jutikliai [26]. Mokslininkų atliktame eksperimente robotas yra valdomas komandomis, kurias operatorius užduoda paspausdamas vieną ar kitą valdymo mygtuką išmaniojo telefono ekrane arba atlikdamas tam tikrus rankų gestus su išmaniuoju telefonu (1 priedas, 11 pav.). Roboto valdymo komandos ir kita reikalinga informacija iš išmaniojo telefono į robotą perduodama bevieliu ryšiu.

Roboto valdymo sistema grindžiama išmaniuoju telefonu su *Android* OS susideda iš keturių sluoksnių: *Linux* branduolio, *Android* OS bibliotekos, programos karkaso ir pačios programos (1 priedas, 12 pav.). *Linux* branduolys tarnauja kaip tarpininkas tarp techninės įrangos ir likusios programinės įrangos. *Android* OS bibliotekos, tai bibliotekos kurias naudoja kitos programos. Programos karkasas suteikia naudingą sąsają patogiam bibliotekų naudojimui [26]. Be įprastų *Android* OS aptartų dalių, mokslininkai papildomai sukūrė turinio valdymo modulį, kuris atsakingas už informacijos surinkimą, perdavimą. Pavyzdžiui jeigu operatorius atlieka tam tikrą gestą su išmaniuoju telefonu, informacija apie atliktą gesta būtent ir patenka į turinio valdymo modulį. Taip pat mokslininkai sukūrė ir roboto valdymo modulį, kuris pagal turinio valdymo modulio turimą informaciją sugeneruoja roboto valdymo instrukcijas. Roboto valdymo proceso eiga:

1. Informacija apie operatoriaus padarytus gestus ar paspaustas valdymo komandas išmaniojo telefono ekrane patenka į turinio valdymo modulį.
2. Turinio valdymo modulis kreipiasi į roboto valdymo modulį, kuris pagal gautus duomenis iš turinio valdymo modulio sugeneruoja roboto valdymo instrukcijas.
3. Sugeneruotos valdymo instrukcijos yra gražinamos atgal į turinio valdymo modulį.
4. Turinio valdymo modulis gautas valdymo instrukcijas bevieliu ryšiu išsiunčia į robotą.
5. Robotas gavęs atitinkamas instrukcijas įvykdo vieną ir kitą komandą [26].

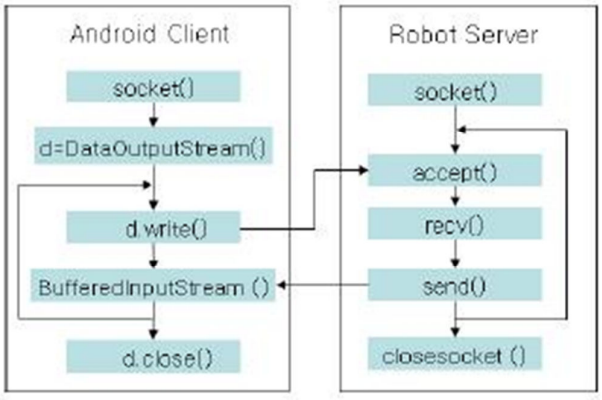
Roboto valdymas vyksta realiu laiku, t.y. operatorius stebėdamas roboto veiksmus, nusprendžia ir įvertina, kokią komandą reikia užduoti robotui.

Atlikti eksperimentai parodė, kad nuotoliniam robotų valdymui gali būti panaudoti šiuolaikiniai išmanieji telefonai su *Android* OS. Šiuo konkrečiu atveju buvo sukurti roboto valdymo moduliai, kurių dėka galima valdyti ir kitus standartinius robotus.

Y. H. Jeon ir H. Ahn sukūrė nuotolinio valdymo sistemą taip pat pritaikyta valdyti paslaugų robotui [10]. Mokslininkų sukurta nuotolinio valdymo sistema išsiskiria tuo, kad robotą nuotoliniu būdu galima valdyti ne tik vietiniame *Wi-Fi* bevieliame tinkle bet ir per internetą. Visa nuotolinio

valdymo sistema susideda iš išmaniojo telefono su Android OS ir standartinio paslaugų roboto (1 priedas, 13 pav.). Eksperimento metu mokslininkai naudojo standartinį paslaugų robotą. Robotą sudaro galva su įmontuota kamera, roboto važiuoklė, taip pat robotas turi dvi rankas, kurių pagalba gali sugriebti daiktus. Robote taip pat yra įtaisytas kompiuteris, kuris šiuo atveju naudojamas kaip serveris. Išmaniajame telefone įdiegta specializuota programa, kurios pagalba nuotoliniu būdu valdomas robotas.

Kadangi išmaniojo telefono ekrano dydis yra ribotas, todėl robotas valdomas trimis režimais [10]. Įjungus pirmąjį režimą yra valdomas roboto judėjimas. Šiuo atveju paspaudus mygtuką išmaniojo telefono ekrane, robotui išsiunčiama judėjimo komanda. Įjungus antrąjį režimą – valdomos roboto rankos bei galva. Komandos robotui yra išsiunčiamos paspaudus atitinkamą valdymo mygtuką išmaniojo telefono ekrane. Įjungus trečiąjį režimą, robotas valdomas operatoriaus balso komandomis. Šiuo atveju operatorius komandą pasako balsu. Balso komandą atpažįsta išmaniojo telefono programinė įranga ir atitinkamai pagal atpažintą komandą yra išsiunčiama valdymo komanda robotui. Roboto kompiuteryje taip pat įdiegta specializuota programinė įranga, kuri pagal gautas komandas valdo patį robotą. Bendravimo tarp roboto ir išmaniojo telefono eiga (7 pav.):



7 pav. Duomenų perdavimo eiliškumas tarp išmaniojo telefono ir roboto kompiuterio [10]

1. Roboto serveris sukuria tinklo prieigą (socket) ir laukia signalo iš išmaniojo telefono.
2. Išmaniojo telefono programa taip pat sukuria tinklo prieigą ir naudodama funkcijas DataOutputStream() ir d.write() siunčia signalą į serverį.
3. Serveris gavęs siganalą iš telefono, siunčia į telefoną nuotraukas iš kameros.
4. Vaizdinė medžiaga išmaniojo telefono programoje yra priimama panaudojant funkciją BufferedInputStream(). Jeigu išmaniojo telefono programa siunčia signalą į serverį apie mygtuko paspaudimą, serveris šią komandą priima naudodamas funkciją recv().

Į išmanųjį telefoną iš roboto yra siunčiamas tiek vaizdas iš roboto galvoje esančios kameros, tiek garsas. Taip pat robotas į išmanųjį telefoną siunčia informaciją apie vykdomą komandą.

Operatorius matydamas valdomą robotą tiek gyvai, tiek išmaniojo telefono ekrane gali lengviau orientuotis kokioje padėtyje yra valdomas robotas. Atlikti mokslininkų tyrimai parodė, kad *Android* OS puikiai tinka kurti roboto valdymo programas. Iš esmės programos sudėtingumas ir paskirtis priklauso tik nuo paties programuotojo ir žinoma nuo roboto, kurį reikia valdyti. Kadangi *Android* OS yra atvirojo kodo, programuotojas turi visas galimybes modifikuoti tiek pačią operacinę sistemą tiek kuriamą programinę įrangą.

Valdant robotus su išmaniuoju telefonu nuotoliniu būdu naudojamas *Bluetooth* ar *Wi-Fi* bevielis ryšys. H. Gulati ir kiti jo grupės nariai sukūrė nuotoliniu būdu valdomą gelbėjimo robotą [5]. Gelbėjimo robotų panaudojimas sparčiai auga, nes tokiu būdu yra apsaugomos, gelbėjimo darbus po įvairių gamtos katastrofų, atliekančių žmonių gyvybės. Robotai lyginant su žmonėmis ar specialiai apmokytais šunimis turi pranašumą, nes gali patekti ten kur pastarieji neturi galimybės.

H. Gulati ir jo komandos mokslininkų sukurtas robotas susideda iš platformos su keturiais ratais. Mokslininkai pirmenybę teikia vikšrinei važiuoklei dėl didesnio pravažumo, tačiau bandymų metu vietoj vikšrų buvo naudojami ratai (1 priedas, 14 pav.). Ant platformos yra sumontuotas mikrovaldiklis bei kamera. Šiuo atveju roboto kamerai panaudotas išmanusis telefonas. Taip pat robotas turi infraraudonųjų spindulių jutiklius, kurie reaguoja į arti esančius daiktus. Jutikliams aptikus arti esantį objektą, apie tai iškart informuojamas operatorius. Roboto veiksmai yra valdomi operatoriaus mobiliuoju telefonu (1 priedas, 15 pav.). Roboto valdymo programa buvo sukurta panaudojant java programavimo kalbą [5]. Robotas duomenims gauti ir siųsti gali naudoti tiek *Bluetooth*, tiek *Wi-Fi* bevielį ryšį. Priklausomai nuo poreikių yra naudojamas vienas arba kitas bevielis ryšys.

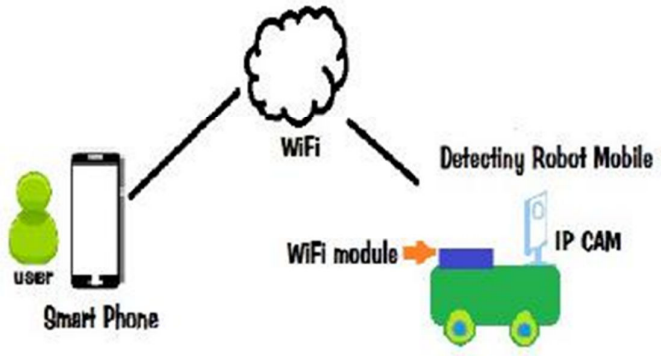
Bluetooth bevielio ryšio privalumai [5]:

- *Bluetooth* bevielis ryšys naudoja mažai elektros energijos;
- *Bluetooth* technologija naudoja nelicencijuotą radijo dažnį 2400-2483,5 MHz. Tokiu būdu yra išvengiama susimaišymo su kitomis radijo bangomis, kurias naudoja kiti robotai;
- *Bluetooth* technologija suteikia abipusio bendravimo galimybę tarp roboto ir operatoriaus. Tas pats *Bluetooth* modulis naudojamas tiek informacijai siųsti tiek gauti;
- *Bluetooth* technologija suteikia galimybę operatoriui vienu metu valdyti net septynis robotus. Operatorius gali vieną kartą išsiųsti komandą, kurią gaus visi septyni robotai. Nėra reikalo valdymo komandos siųsti kiekvienam robotui atskirai;
- *Bluetooth* bevielis ryšys yra labai paplitęs, visi šiuolaikiniai telefonai bei nešiojami kompiuteriai turi *Bluetooth* modulį. Didelis paplitimas suteikia galimybę naudotis jau įdiegta technologija, dėl to nereikia papildomų išlaidų;

- *Bluetooth* bevielio ryšio signalo stiprumas yra proporcingas atstumui tarp roboto ir operatoriaus. Todėl pagal signalo stiprumą apytiksliai galima spręsti apie atstumą skiriantį operatorių ir valdomą robotą.

Wi-Fi bevielis ryšys turi tuos pačius privalumus kaip ir *Bluetooth* bevielis ryšys, tačiau esminis skirtumas tarp šių dviejų technologijų yra tas, kad *Wi-Fi* bevielio ryšio duomenų perdavimo sparta yra 11 Mbps, o bluetooth ryšio 720 kbps [5]. Mokslininkų naudojamas *Wi-Fi* bevielio ryšio standartas yra *IEEE* 802.11b. Realiai galima būtų naudoti kitą *Wi-Fi* bevielio ryšio standartą *IEEE* 802.11n. Su šiuo standartu suderinamo įrenginio duomenų perdavimo sparta yra iki 300 Mbps. Mokslininkų atliktame eksperimente panaudotas *Bluetooth* įrenginys nėra pats sparčiausias. Eksperimente naudoto *Bluetooth* įrenginio versija yra 1.2. Naujausio ketvirtos versijos *Bluetooth* įrenginio duomenų perdavimo sparta yra iki 25 Mbps. Gulati ir kitų mokslininkų atliktame eksperimente *Bluetooth* ryšys naudojamas tada, kada nėra poreikio perduoti vaizdo iš roboto kameros. Jeigu būtina perduoti vaizdą iš roboto kameros – naudojamas *Wi-Fi* bevielis ryšys, nes *Bluetooth* yra per lėtas.

Panašus nuotoliniu būdu valdomas paieškos robotas aprašytas [15]. Pasiūlyta nuotolinio valdymo sistema susideda iš išmaniojo telefono ir valdomo roboto (8 pav.).



8 pav. Roboto nuotolinio valdymo sistema [15]

Pats robotas turi du variklius, kamerą, roboto valdymo mikroschemą, prie kurios prijungtas *Wi-Fi* bevielio ryšio modulis (1 priedas, 16 pav.). Išmaniajame telefone yra įdiegta specializuota programa, kuri atsakinga už roboto valdymą. Atidarius valdymo programą išmaniajame telefone pasirodo langas, kuriame yra išdėstyti roboto valdymo mygtukai bei dalis lango palikta kameros vaizdui. Robotui valdymo komanda gali būti užduodama paspaudžiant valdymo mygtuką telefono ekrane. Taip pat įjungus kitą valdymo režimą, komandos robotui siunčiamos atitinkamai pakeitus telefono padėtį erdvėje (1 priedas, 17 pav.). Šiuo atveju programa nuskaito išmaniojo telefono akselerometrinio jutiklio duomenis ir taip nusprendžia, kokią valdymo komandą reikia išsusti robotui [15]. Išmanusis telefonas ir robotas tarpusavyje bendrauja bevieliu *Wi-Fi* ryšiu.

Atliktas eksperimentas parodė, kad paieškos roboto nuotoliniam valdymui puikiai gali būti panaudotas išmanusis telefonas su akselerometriniu jutikliu. Ateityje mokslininkai planuoja dar tobulinti šią nuotolinio valdymo sistemą pritaikydami ją kitų robotų nuotoliniam valdymui.

Labai spartus išmaniųjų telefonų populiarėjimas atveria vis naujų galimybių. Išmaniuosius telefonus stengiamasi pritaikyti visur kur tik įmanoma. Išmaniaisiais telefonais valdoma ne tik namuose esanti įranga, pramoniniai robotai, bet ir daugelis kitų įvairiausių robotų, kuriuos gali sukurti žmogus. Įvardinti būtų galima mokymo tikslais, žmonių bendravimui palengvinti, sporto tikslais naudojamus robotus. Kas svarbiausia visi šie robotai valdomi nuotoliniu būdu pasitelkiant išmanųjį telefoną.

Mokymo robotų paskirtis yra padaryti mokymo procesą įdomesnį, siekiant labiau sudominti vaikus mokymo procesu. Mokslininkas B. D. Kang ir jo komandos nariai panaudojo išmanųjį telefoną su Android OS mokymo roboto nuotoliniam valdymui [12]. Šiuo atveju buvo panaudotas standartinis mokymo robotas iRobi-Q. Robotas turi galvą, rankas, monitoriaus ekraną, garso kolonėles ir važiuoklę. Pačiame robote įdiegta specializuota programinė įranga, kuri analizuoja gautas komandas iš išmaniojo telefono ir taip valdo roboto judėjimą. Išmaniajam telefonui taip pat buvo sukurta speciali programa, kurios pagalba nuotoliniu būdu valdomas robotas. Robotas ir išmanusis telefonas tarpusavyje bendrauja *Wi-Fi* bevieliu ryšiu. Robotą valdantis žmogus valdymo komandas į robotą perduoda paspausdamas vieną ar kitą valdymo mygtuką telefono ekrane. Taip pat išmaniojo telefono programa turi funkciją įrašyti robotą valdančio žmogaus balsą, o įrašytą balsą persiųsti robotui, kuris ištransliuos įrašą per kolonėles. Atlikti mokslininkų eksperimentai parodė, kad toks mokymo būdas išties gali labai sudominti vaikus, todėl ateityje jie dar planuoja tobulinti sukurtą nuotolinio valdymo sistemą. Atlikti eksperimentai parodė, kad išmanusis telefonas yra išties puiki priemonė nuotoliniam robotų valdymui.

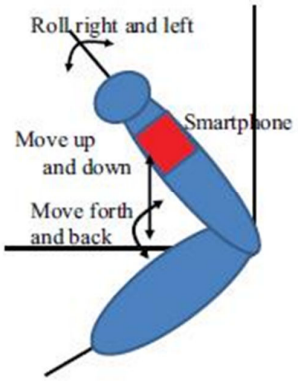
Panašiu principu veikia ir kitas robotas, kurį sukūrė D. W. Choi ir jo vadovaujama grupė. Šiuo atveju mokslininkai išmanųjį telefoną su *Android* OS panaudojo video konferencijų roboto nuotoliniam valdymui [4]. Patį robotą sudaro kamera, važiuoklė su dvejais ratais, bevielio ryšio *Wi-Fi* modulis, motorai bei kompiuteris su monitoriumi. Programinė įranga kuri valdo roboto veiksmus sukurta panaudojant *Microsoft Visual Studio* 2008 [4]. Eksperimentui buvo panaudotas Nexus S išmanusis telefonas su *Android* OS (1 priedas, 18 pav.). Programinė įranga išmaniajame telefone buvo sukurta naudojant java ir xml programavimo kalbas. Išmaniojo telefono programoje yra išdėstyti roboto valdymo mygtukai, bei išskirta vieta vaizdui iš roboto kameros peržiūrai (1 priedas, 19 pav.). Išmanusis telefonas ir robotas tarpusavyje bendrauja *Wi-Fi* bevieliu ryšiu. Atlikti mokslininkų eksperimentai parodė, kad išmanusis telefonas yra išties tinkama priemonė tokio tipo robotams valdyti. Nes šiuo atveju visos valdymo funkcijos telpa telefono ekrane. Be to dėka išmaniajame telefone įdiegtos priekinės kameros į robotą yra siunčiamas ir operatoriaus vaizdas.

Šiuo atveju išmaniajame telefone įdiegta techninė įranga suteikia visas galimybes video konferencijos robotui valdyti.

Išmanieji telefonai taip pat gali būti panaudoti ir sporto robotams valdyti. Mokslininkas K. Serafimov ir kiti jo komandos nariai panaudojo išmanųjį telefoną futbolo roboto nuotoliniam valdymui [25]. Mokslininkų pasiūlytas nuotolinis robotų valdymas remiasi žmogaus rankų judesiais. Operatorius laikydamas telefoną rankoje atlieka tam tikrą judesį ranka, o išmaniojo telefono programa atpažinusi vieną ar kitą judesį atitinkamai išsiunčia valdymo komanda į robotą. Žmogaus rankų judesiai yra fiksuojami akselerometrinio jutiklio įdiegto išmaniajame telefone.

Pirmiausia norėdami pasiekti savo tikslą mokslininkai turėjo sukurti žmogaus rankos gestų apmokymų duomenų bazę. Šiam tikslui pasiekti buvo pasitelkta 30 žmonių, kurių kiekvienas turėjo atlikti visus reikalingus rankos gestus. Visi mokymo metu gauti duomenys buvo įrašyti į duomenų bazę. Kai jau buvo sukurta duomenų bazė su kuria bus lyginami operatoriaus rankų gestai, buvo atliekami bandymai, kurių tikslas išsiaiškinti, kokiu tikslumu yra atpažįstami operatoriaus gestai. Eksperimento vykdymo metu kiekvienas gestas buvo kartojas dešimt kartų, o visus gestus atliko penki skirtingi žmonės. Surinkti rezultatai parodė, kad gestų atpažinimo tikslumas 94% [25]. Visiškas gestu atpažinimo tikslumas nebuvo pasiektas, kadangi gestus atliekant skirtingiems žmonėms, paties gesto atlikimo technika šiek tiek skiriasi.

Atlikti mokslininkų tyrimai parodė, kad laikant išmanųjį telefoną delne, žmogaus rankos gestų atpažinimo tikslumas siekia 94%. Mokslininkas Y. Murata ir jo komandos nariai panaudoto išmaniojo telefono teikiamas galimybes siekdami padėti rankos pirštų neturintiems žmonėms [19]. Mokslininkų pasiūlyta idėja ta, kad pritaisius išmanųjį telefoną prie žmogaus rankos riešo, galima taip pat puikiai atpažinti žmogaus rankos judesius, kaip ir laikant telefoną žmogaus delne. Mokslininkų sukurta nuotolinio valdymo sistema daugiau orientuota į įvairių prietaisų nuotolinį valdymą, kaip televizoriai, tačiau pats principas ir gestų atpažinimo technika gali būti nesunkiai pritaikyta ir robotų nuotoliniam valdymui. Išmaniojo telefono pritvirtinimo vieta ant žmogaus rankos (9 pav.).



9 pav. Išmaniojo telefono pritvirtinimas ant rankos [19]

Išmaniajame telefone specialiai sukurta programa nuskaito žmogaus daromus rankų judesius ir atitinkamai išsiunčia reikiamą komandą į valdomą prietaisą. Atlikti mokslininkų tyrimai parodė, kad išmanusis telefonas gali puikiai pasitarnauti ne tik sveikiems žmonėms, bet išnaudojus visas telefono galimybes gali padėti ir negalią turinčiam žmogui. Mokslininkų tyrimai tik įrodo, kad žmogaus rankų judesių atpažinimas valdant prietaisus, robotus, turi didžiulę reikšmę, siekiant sukurti paprastą ir lengvai valdomą nuotolinio valdymo sistemą.

Mokslininkai H. K. Oh ir I. C. Kim panaudojo išmanųjį telefoną kurdami namų apsaugos roboto nuotolinio valdymo sistemą [21]. Mokslininkų pasiūlyta nuotolinio valdymo sistema susideda iš roboto, serverio bei išmaniojo telefono. Robotą sudaro vikšrinė važiuoklė, kamera, sensorius objektams aptikti, bevielio ryšio modulis. Roboto valdymo proceso eiga:

1. Operatoriaus telefono ekrane yra išdėstyti roboto valdymo mygtukai, kuriuos paspaudus telefonas išsiunčia valdymo komandą į tarpinį serverį.
2. Tarpinis serveris gavęs komandą, perduoda gautą komandą į namuose esantį robotą.
3. Robotas gavęs komandą įvykdo atitinkamus veiksmus (1 priedas, 20 pav.).

Robotas taip pat į išmanųjį telefoną siunčia duomenis iš sensorių. Pats robotas gali savarankiškai išvengti kliūčių pasitelkdamas kliūčių aptikimo sensorių. Į operatoriaus išmanųjį telefoną taip pat yra perduodamas vaizdas iš roboto kameros, todėl operatorius robotą gali valdyti net ir tiesiogiai jo nematydamas.

Namų stebėjimo robotui valdyti išmanųjį telefoną taip pat panaudojo ir mokslininkai S. Y. Juang ir J. G. Juang. Mokslininkų sukurtą nuotolinio valdymo sistemą taip pat sudaro valdomas robotas su įtaisyta kamera, tarpinis kompiuteris ir išmanusis telefonas. Šiuo atveju duomenims perduoti tarp roboto ir kompiuterio naudojamas *ZigBee* bevielio ryšio modulis. Roboto valdymo eiga:

1. Operatorius išmaniojo telefono ekrane paspaudžia valdymo komandą.
2. Išmanusis telefonas komandą perduodą į tarpinį kompiuterį.
3. Tarpinis kompiuteris gautą komandą perduoda robotui, kuris įvykdo atitinkamą veiksmą.

Mokslininkų atlikti eksperimentai parodė, kad išmanusis telefonas yra puiki priemonė namų robotams valdyti. Kadangi išmanusis telefonas yra mažo dydžio, todėl žmogus gali jį nešiotis visur su savimi ir esant reikalui, patogiai valdyti namus prižiūrintį robotą nuotoliniu būdu.

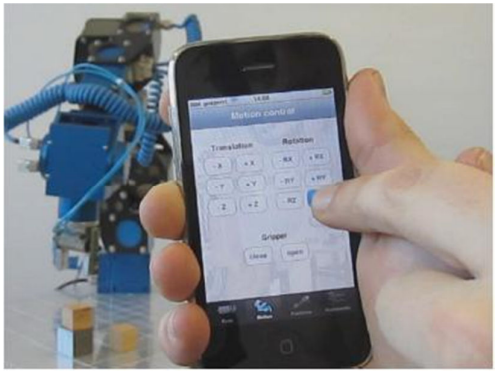
Kaip matyti iš atliktų mokslininkų eksperimentų išmanusis telefonas vis labiau populiarėja kaip valdymo priemonė robotams valdyti nuotoliniu būdu. Todėl daugelis mokslininkų bando pritaikyti išmaniojo telefono teikiamas galimybes savo atliekamuose eksperimentuose. Mokslininkai D. Hess ir C. Rohrig savo eksperimentuose taip pat pritaikė išmanųjį telefoną. Visas mokslininkų eksperimentas buvo daugiau orientuotas į patį duomenų perdavimo procesą tarp roboto ir išmaniojo telefono [6]. Anot mokslininkų ten kur atstumai nėra dideli ir nėra didelis duomenų

perdavimo kiekis, patogu naudoti *Bluetooth* bevielį ryšį, ten kur reikalinga didesnė duomenų perdavimo sparta, reikėtų naudoti *Wi-Fi* bevielį ryšį. Ryšio pasirinkimas priklauso nuo poreikių. Patį roboto valdymo procesą taip pat daugiau tyrinėja mokslininkas R. Tsuchiya ir jo komanda. Savo atliktame eksperimente roboto valdymui jie panaudojo ne tik valdymo mygtukus telefono ekrane ir akselerometrinį jutiklį, bet papildomai panaudojo ir išmaniojo telefono orientacijos jutiklį [27]. Mokslininkai A. M. Walker ir D. P. Miller savo atliktame eksperimente bando nustatyti, kaip patogiau valdyti robotą, kada vaizdas iš roboto yra perduodamas į išmaniojo telefono ekraną. Pirmiausia jie analizuoja valdymo patogumą, kada robotas yra valdomas mygtukais telefono ekrane, o vaizdas užima tik pusę telefono ekrano. Kitas variantas robotas valdomas akselerometrinio jutiklio esančio išmaniajame telefone pagalba, o visas telefono ekranas yra skiriamas vaizdui iš roboto kameros. Mokslininkas S. W. Moon ir jo komanda savo eksperimente išbandė robotą, kurio valdymo kompiuteryje įdiegta *Android* OS [18]. O pats robotas taip pat buvo valdomas išmaniojo telefono su *Android* OS. Eksperimentas buvo labiau orientuotas į *Android* OS išbandymą, kaip roboto operacinės sistemos ir į vaizdo perdavimą iš roboto kameros į išmanųjį telefoną. Atlikti eksperimentai buvo sėkmingi, mokslininkai nustatė, kad *Android* OS labai patogu naudoti ir kaip jau paties roboto operacinę sistemą. Anot mokslininkų ateityje tūrėtų pasirodyti, vis daugiau robotų, kurie naudos ne kokią kitą operacinę sistemą, bet naudos būtent *Android* OS. Iš esmės pats roboto nuotolinis valdymas yra lengviau įgyvendinamas, kada tiek išmanusis telefonas tiek valdomas robotas turi tą pačią *Android* OS.

Iš ankščiau nagrinėtų mokslininkų eksperimentų matyti, kad išmanieji telefonai naudojami valdyti patiems įvairiausiems robotams. Ne išimtis yra ir pramoniniai robotai, kurie taip valdomi nuotoliniu būdu panaudojant išmanųjį telefoną. Mokslininkas S. M. Abbas ir jo komanda panaudojo išmanųjį telefoną valdydami ir programuodami pasirinktą industrinį robotą. Anot mokslininkų iki šiol skirtingi robotų gamintojai sukurdavo skirtingus roboto valdymo ir programavimo prietaisus [1]. Net ir to paties gamintojo skirtingi robotai būdavo valdomi ir programuojami skirtingais valdymo prietaisais. Todėl mokslininkų tikslas yra sukurti daug bendresnę pramoninių robotų valdymo sistemą, kurią būtų galima pritaikyti daugeliui pramoninių robotų. Šiuo atveju į pagalbą ir ateina išmanusis telefonas, kaip standartinis pramoninio roboto valdymo ir programavimo įrenginys. Mokslininkų pasiūlyta nuotolinio valdymo sistema susideda iš pasirinkto pramoninio roboto, serverio prie kurio šis robotas prijungtas ir išmaniojo telefono. Išmaniajame telefone yra įdiegta specializuota programa specialiai sukurta pramoninio roboto valdymui ir programavimui. Išmanusis telefonas prie roboto serverio prisijungia naudodamas TCP/IP protokolą [1]. Robotui valdymo komandos yra siunčiamos išmaniojo telefono ekrane paspaudus valdymo mygtukus. Taip pat galima pasinaudoti ir išmaniajame telefone įdiegtu akselerometriniu jutikliu. Įjungus šį valdymo režimą pramoninio roboto judesiai valdomi keičiant išmaniojo telefono padėtį erdvėje. Robotui

valdymo komandas taip pat galima siųsti ir pasinaudojant tekstine įvestimi, t.y. komandos užduodamos ne tik paspaudus mygtuką telefono ekrane, bet reikiamą valdymo komandą galima tiesiog įvesti tam skirtoje vietoje telefono ekrane ir ši komanda bus išsiųsta į robotą. Išmaniojo telefono panaudojimas pramoninių robotų valdymui ir programavimui ateityje tūrėtų supaprastinti robotų valdymo ir programavimo procesą. Robotų gamintojai galės atsisakyti atskirų roboto valdymo įrenginių gamybos ir visą roboto valdymą sukoncentruoti išmaniajame telefone.

Mokslininkas J. Lambrecht ir jo komandos nariai taip pat panaudojo išmanųjį telefoną pramoninio roboto nuotoliniam valdymui [14]. Išmaniajam telefonui taip pat buvo sukurta specializuota programa, kurios pagalba valdomas pramoninis robotas. Operatorius valdymo komandas robotui užduodavo dviem būdais. Pirmiausia operatorius galėdavo paspausti roboto valdymo mygtukus esančius išmaniojo telefono ekrane (10 pav.). Antruoju būdu, operatorius galėdavo pasinaudoti išmaniajame telefone įdiegtu akselerometriniu jutikliu. Duomenų perdavimas tarp išmaniojo telefono ir roboto vykdomas *Wi-Fi* bevieliu ryšiu.



10 pav. Pramoninio roboto nuotolinis valdymas naudojant išmanųjį telefoną [14]

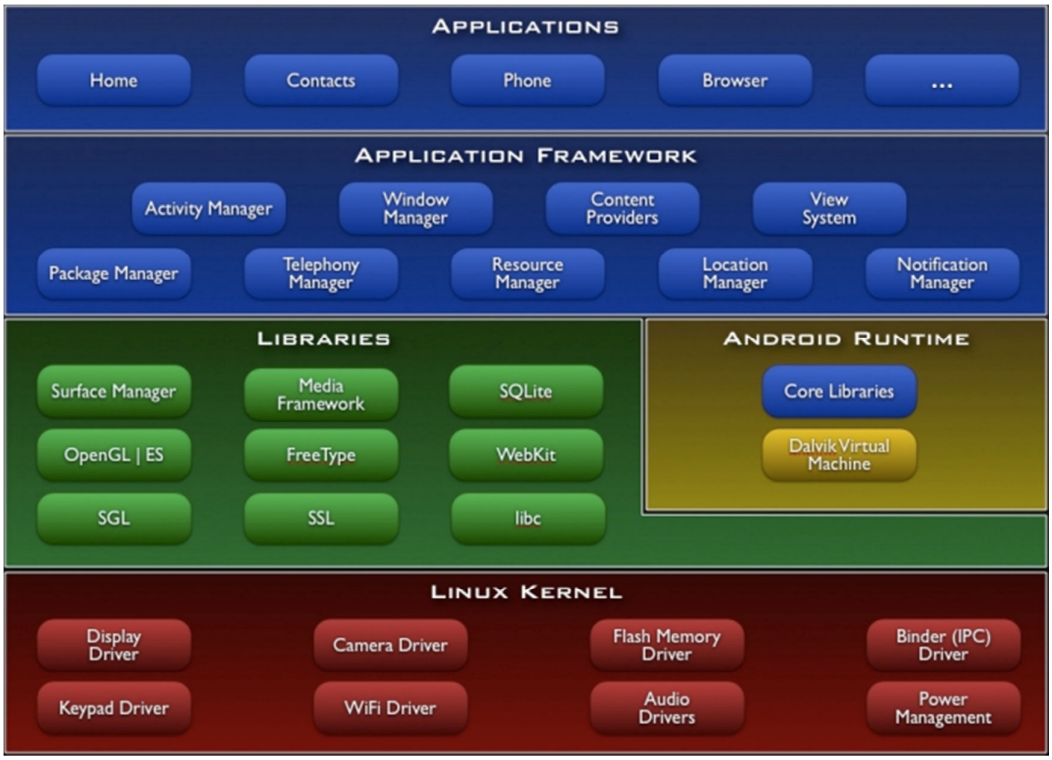
Mokslininkų teigimu šiuolaikinių išmaniųjų telefonų funkcionalumas suteikia galimybę atsisakyti įprastų roboto valdymo prietaisų, taip robotų valdymui panaudojant naujausius išmaniųjų telefonų technikos laimėjimus.

Kaip matyti iš nagrinėtų mokslininkų atliktų darbų išmanusis telefonas naudojamas labai įvairiems robotams valdyti. Tiek pradedant nuo paieškos, gelbėjimo robotų, paslaugų robotų, tiek baigiant pramoniniais robotais. Išnagrinėjus mokslininkų atliktus darbus galima teigti, kad paties roboto valdymo principas išmaniuoju telefonu yra panašus. Keičiantis robotų tipams, kartu keičiasi išmaniojo telefono programos struktūra. Iš esmės išmaniojo telefono programa yra priderinama prie valdomo roboto poreikių. Kuriant išmaniojo telefono programą yra atsižvelgiama į tai ką robotas gali daryti. Žinant roboto galimybes atitinkamai suprogramuojami roboto valdymo mygtukai, kurių pagalba bus užduodamos komandos robotui. Taip pat kuriant telefono programinę įrangą yra išnaudojamas ir telefone įdiegto akselerometrinio jutiklio galimybės. Akselerometro panaudojimas kuriant išmaniojo telefono programinę įrangą leidžia atsisakyti komandinių mygtukų. Tai leidžia

taupyti išmaniojo telefono ekrano vietą ir ją esant reikalui panaudoti kitiems tikslams, nes kaip bebūtų išmaniojo telefono ekrano įstrižainė yra maža ir ten telpa ribotas kiekis valdymo mygtukų. Taip pat akselerometrinio jutiklio panaudojimas roboto valdymą padaro paprastesnį, nes operatoriui nebereikia spaudyti valdymo mygtukų telefono ekrane. Duomenų perdavimui tarp išmaniojo telefono ir nuotoliniu būdu valdomo roboto yra naudojamas tiek *Bluetooth*, tiek *Wi-Fi* bevielis ryšis. Jeigu nėra reikalinga didelė duomenų sparta geriau rinktis *Bluetooth* bevielį ryšį. Jeigu reikalinga didelė duomenų perdavimo sparta tarp roboto ir išmaniojo telefono geriau naudoti *Wi-Fi* bevielį ryšį. Tačiau naujausi ketvirtos versijos *Bluetooth* bevielio ryšio moduliai palaiko net iki 25 Mbit/s duomenų perdavimo greitį. Todėl daugeliu atveju nėra jokio reikalo naudoti *Wi-Fi* bevielio ryšio, nes reikiamam rezultatui pasiekti užtenka panaudoti naujausia *Bluetooth* modulį. Vieną ar kitą duomenų perdavimo būdą reikia pasirinkti pagal poreikius.

1.2. Išmaniojo telefono Android operacinė sistema

Šiame darbe robotų nuotoliniam valdymui naudojamas *Samsung Galaxy Nexus I9250* išmanusis telefonas su *Android* 4.2.1 *Jelly Bean* operacine sistema. *Android* OS yra atviro kodo, daugiausia naudojama išmaniuosiuose telefonuose, planšetiniuose kompiuteriuose [22]. Be šių įrenginių operacinė sistema taip pat po truputį pradeda plisti ir kituose elektroniniuose prietaisuose. Visai neseniai pasirodė pirmasis fotoaparatas su *Android* OS. *Android* operacinės sistemos architektūros diagrama pateikta 11 pav.



11 pav. Android operacinės sistemos architektūra [2]

Visa *Android* OS sudaryta iš skirtingų sluoksnių, kurie atlieka tam tikras funkcijas. Svarbiausias operacinės sistemos sluoksnis – *Linux* branduolys [2]. Šis sluoksnis pažymėtas raudona spalva (11 pav.). *Android* OS pagrindą sudaro *Linux* branduolys, kurio architektūra dar šiek tiek modifikavo Google kompanija. *Linux* branduolys sąveikauja su visa išmaniojo telefono technine įranga ir turi visas techninės įrangos tvarkykles. Linux branduolys yra tarsi tarpinis sluoksnis tarp išmaniojo telefono techninės įrangos ir kitų operacinės sistemos sluoksnių. Šiame darbe naudojamas *Bluetooth* bevielis ryšys duomenų perdavimui tarp išmaniojo telefono ir roboto. *Linux* branduolys užtikrina tinkamą *Bluetooth* techninės įrangos darbą.

Kitas operacinės sistemos sluoksnis – bibliotekos [2]. Šis sluoksnis pažymėtas žalia spalva (11 pav.). Šis sluoksnis išmaniajam telefonui suteikia galimybę naudoti įvarius duomenų tipus. Dažniausiai naudojamos bibliotekos:

- *SQLite* biblioteka skirta darbui su išmaniojo telefono duomenų bazėmis;
- *WebKit* biblioteka skirta HTML turinio atvaizdavimui interneto naršyklėse;
- *OpenGL* biblioteka skirta 2D ir 3D grafikos atvaizdavimui ekrane;
- *Media Framework* biblioteka skirta groti įvairaus audio formato muziką bei ją įrašyti.

Android Runtime operacinės sistemos sluoksnis [2]. Šis sluoksnis pažymėtas geltona spalva (11 pav.). Sluoksnis susideda iš dviejų dalių:

a. *Dalvik* virtuali mašina. Ši dalis išmaniuosiuose telefonuose paleidžia ir vykdo išmaniojo telefono programas. Ši dalis sukurta taip, kad naudotų mažai išmaniojo telefono resursų. Skirtingai negu java kalbos virtuali mašina, *Dalvik* virtuali mašina vykdo dex failus, o ne class failus. Dex failų vykdymas vietoj, class failų, suteikia didesnę spartą, esant mažiems išmaniojo telefono resursams.
b. Bendrosios java kalbos bibliotekos.

Programų karkaso sluoksnis [2]. Šis sluoksnis pažymėtas mėlyna spalva (11 pav.). Šį sluoksnį sudaro tam tikri programiniai blokai, su kuriais tiesiogiai sąveikauja programa. Dažniausiai naudojami blokai:

- *Activity Manager* blokas tvarko išmaniojo telefono programų langų gyvavimo ciklą;
- *Content Provider* blokas tvarko duomenys tarp skirtingų išmaniojo telefono programų;
- *Telephony Manager* blokas tvarko išmaniojo telefono skambinimo funkcijas;
- *Resource Manager* blokas tvarko visus resursus, kurie naudojami išmaniojo telefono programoje.

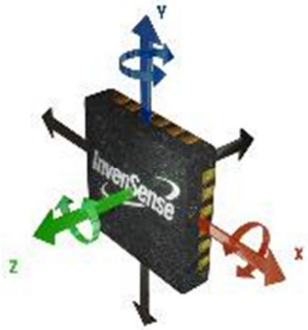
Programų sluoksnis [2]. Šis sluoksnis taip pat pažymėtas mėlyna spalva (11 pav.). Šis sluoksnis yra pats aukščiausias sluoksnis *Android* OS architektūroje. Būtent į šitą sluoksnį ir

patenka visos sukurtos išmaniojo telefono programos. Ne išimtis ir šiame darbe sukurta nuotolinio roboto valdymo programa, kuri taip pat patenka į šį sluoksnį.

Aptartą *Android* OS architektūrą sudaro įvairūs sluoksniai, kurių kiekvienas atlieka savitas funkcijas. Kadangi *Android* OS yra atvirojo kodo, tai programuotuojui suteikia labai plačias galimybes. Be to, kad programuotojas gali kurti naujas programas šiai operacinei sistemai, jis taip pat gali pakeisti ir pačios operacinės sistemos programas. Tai iš esmės programuotojui suteikia praktiškai neribotas kūrimo galimybes.

1.3. Išmaniojo telefono jutikliai

Šiuolaikiniai išmanieji telefonai turi labai daug galimybių, kurių neturi įprasti mobiliojo ryšio telefonai. Dalį šių naujų galimybių išmaniajam telefonui būtent ir suteikia įmontuoti įvairūs jutikliai. Išmaniuosiuose telefonuose naudojami mikro-elektro-mechaninės sistemos (MEMS) tipo jutikliai. MEMS technologijos dėka įprastos mechaninės arba elektro-mechaninės sistemos yra labai smarkiai sumažintos, tokiu būdu gali būti įdiegiamos tokiuose mažuose įrenginiuose kaip išmanusis telefonas [17]. MEMS įrenginių dydis gali svyruoti nuo mažiau kaip vieno mikrono iki kelių milimetrų. MEMS įrenginiai gali būti ganėtinai paprasti, neturintys jokių judančių dalių, taip pat gali būti labai sudėtingi, turintys judančias dalis, kurias valdo integruota mikroelektronika. Išmaniuosiuose telefonuose naudojami MEMS tipo jutikliai dažniausiai mechaninę energiją paverčia į elektrinį signalą [17]. MEMS tipo jutiklių yra pačių įvairiausių, tačiau šiame darbe yra naudojami du MEMS tipo jutikliai. Tai būtų akselerometrinis ir giroskopinis jutikliai. Akselerometrinis jutiklis matuoja linijinį įrenginio pagreitį bei įrenginio pasukimo kampą, t.y. statinę ir dinaminę išmaniojo telefono akceleraciją [8]. Giroskopinis jutiklis matuoja kampinį įrenginio pagreitį. Giroskopinio jutiklio privalumas tas, kad jo neveikia jokios išorinės jėgos, kaip kad gravitacijos jėga ar magnetinis laukas. Šiame darbe naudojamame išmaniajame telefone yra įmontuoti kompanijos InvenSense akselerometrinis ir giroskopinis jutikliai (12 pav.). Tiek akselerometrinis, tiek giroskopinis jutikliai yra trijų ašių.



12 pav. Akselerometrinis ir giroskopinis jutikliai [8]

Giroskopinis jutiklis:

- jutiklio versijos numeris – 1;
- jutiklio skiriamoji geba – 0,57246757 rad/s;
- jutiklio riba – 34,90656 rad/s.

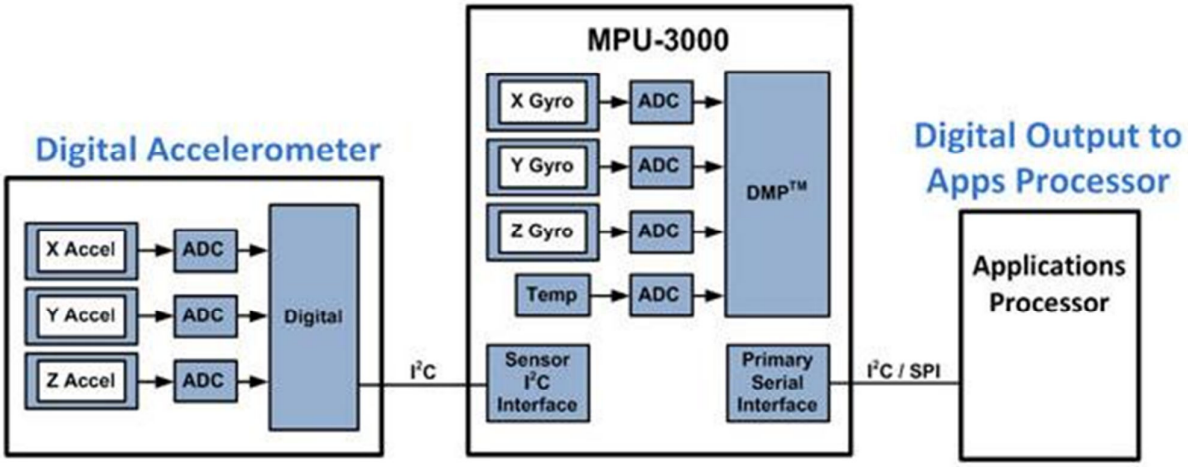
Akselerometrinis jutiklis (statinė akseleracija):

- jutiklio versijos numeris – 1;
- jutiklio skiriamoji geba – 0,038344003 m/s^2;
- jutiklio riba – 19,6133 m/s^2.

Akselerometrinis jutiklis (dinaminė akseleracija):

- jutiklio versijos numeris – 1;
- jutiklio skiriamoji geba – 1,0 m/s^2;
- jutiklio riba – 10240,0 m/s^2.

Šiuolaikiniai išmanieji telefonai privalo turėti mažo dydžio, mažai energijos naudojančius giroskopinius jutiklius. InvenSense yra pirmoji pasaulyje kompanija kuri sukūrė pirmąjį trijų ašių giroskopinį jutiklį MPU-3050^{TM} su įdiegtu skaitmeninių judesių procesoriumi, kitaip tariant technine įranga paremtu spartinimo varikliu [7]. Šiame darbe naudojamame išmaniajame telefone būtent ir įdiegtas šis pažangus giroskopinis jutiklis. Ši pažangi giroskopinio jutiklio technologija kartu apjungia ir akselerometrinį jutiklį taip sudarydama 6 ašių judesių atpažinimo įrenginį (13 pav.).



13 pav. 6 ašių judesių atpažinimo įrenginio schema [7]

Duomenys iš akselerometrinio jutiklio per I^2C sąsaja patenka į giroskopinio jutiklio bloką. Dėka skaitmeninio judesių procesoriaus visi duomenų skaičiavimai susiję su duomenų apdorojimu iš akselerometrinio ir giroskopinio jutiklio yra vykdomi MPU-3050^{TM} giroskopinio jutiklio bloke. Ši technologija žymiai pagreitina išmaniojo telefono darbą, nes skaičiavimais nėra apkranaunamas

pagrindinis išmaniojo telefono procesorius. MPU-3050^{TM} blokas apdorojęs duomenis persiunčia juos per I^2C/SPI sąsają reikiamai programai. Programa gavusi duomenis iš sensorių toliau juos naudoja tam, kam yra užprogramuota.

Darbo tikslas:

Sukurti ir ištirti roboto manipuliatoriaus nuotolinio valdymo sistemos prototipą Android operacinėje sistemoje su įdiegtais akselerometriniu ir giroskopiniu jutikliais.

Darbo uždaviniai:

1. Išanalizuoti robotų nuotolinio valdymo ypatumų mokslinę literatūrą.
2. Sukurti roboto manipuliatoriaus nuotolinio valdymo išmaniuoju telefonu sistemos modelį.
3. Sukurti roboto manipuliatoriaus prototipo programinę įrangą bei specializuotą programinę įrangą išmaniajam telefonui Android OS pagrindu.
4. Atlikti sukurtos nuotolinio valdymo sistemos verifikavimą.

II. MODELIAVIMO DALIS

2.1. Roboto manipuliatoriaus nuotolinio valdymo sistemos reikalavimų specifikavimas

Nuotolinio roboto manipuliatoriaus valdymo sistemos paskirtis:

Modeliuojama nuotolinio valdymo sistema grindžiama išmaniuoju telefonu su *Android* OS yra skirta roboto manipuliatoriaus nuotoliniam valdymui.

Roboto manipuliatoriaus nuotolinio valdymo sistemos tikslai:

a. Kuriamas roboto manipuliatoriaus prototipas privalo tinkamai atlikti bevieliu ryšiu gautas valdymo komandas iš operatoriaus išmaniojo telefono bei perduoti informaciją apie vykdomą komandą į operatoriaus išmanųjį telefoną.
b. Kuriama specializuota išmaniojo telefono programinė įranga privalo atpažinti operatoriaus užduodamas roboto valdymo komandas, kurias operatorius įveda paspausdamas atitinkamą mygtuką išmaniojo telefono ekrane arba atlikdamas tam tikrą gestą su telefonu rankoje.
c. Kuriama specializuota išmaniojo telefono programinė įranga privalo perduoti roboto valdymo komandą bevieliu ryšių į roboto manipuliatoriaus prototipą, taip pat priimti siunčiamą informaciją iš roboto manipuliatoriaus.
d. Kuriama specializuota išmaniojo telefono programinė įranga privalo pateikti visą reikiamą informaciją apie roboto manipuliatoriaus vykdomą komandą išmaniojo telefono ekrane.

Roboto manipuliatoriaus nuotolinio valdymo sistemos vartotojai:

Operatorius – tai žmogus, kuris valdo robotą manipuliatorių nuotoliniu būdu.

Suinteresuoti asmenys – tai žmonės, kurie pasitelkia nuotoliniu būdu valdomą robotą manipuliatorių, įvairiems darbams atlikti. Tai gali būti pavojingus darbus, paieškos, gelbėjimo, krovos darbus atliekantys asmenys bei kiti žmonės, kurių veikloje būtina panaudoti nuotoliniu būdu valdomą robotą manipuliatorių.

2.1.1. Reikalavimų sudarymas roboto manipuliatoriaus nuotolinio valdymo sistemai

Apribojimai sprendimui:

Apribojimai sprendimui sudaryti atsižvelgiant į darbo rašymo metu rinkoje esančių naujausių išmaniųjų telefonų techninius parametrus tam, kad būtų pasiektas maksimalus įmanomas nuotolinio valdymo sistemos gestų atpažinimo patikimumas. Taip pat įvertinant elementus, be kurių nuotolinio valdymo sistema negalėtų tinkamai funkcionuoti.

- Roboto manipuliatoriaus valdymui naudojamame išmaniajame telefone privalo būti įdiegta *Android* OS, kurios minimali versija 4.0 arba didesnė. *Android* OS pasirinkta, todėl, kad ji yra atvirojo kodo ir suteikia visus reikiamus resursus tinkamai programinei įrangai sukurti.

- Roboto manipuliatoriaus valdymui naudojamame išmaniajame telefone privalo būti įdiegtas dviejų branduolių procesorius, kurio skaičiavimo greitis būtu ne mažesnis kaip 1,2 GHz. Dviejų branduolių procesorius reikalingas tam, kad sukurta programinė įranga vienu metu galėtų vykdyti du lygiagrečius procesus.
- Roboto manipuliatoriaus valdymui naudojamame išmaniajame telefone privalo būti įdiegtas akselerometrinis bei giroskopinis jutikliai. Šie du davikliai būtini norint tinkamai atpažinti operatoriaus rankų gestus.
- Roboto manipuliatoriaus valdymui naudojamame išmaniajame telefone privalo būti įdiegtas lietimui jautrus ekranas, kurio minimali įstrižainė 3,5 coliai.
- Roboto manipuliatoriaus valdymui naudojamame išmaniajame telefone privalo būti įdiegta 1 GB arba daugiau operatyviosios atminties.
- Roboto manipuliatoriaus valdymui naudojamame išmaniajame telefone privalo būti įdiegtas bevielio ryšio *Bluetooth* modulis.
- Roboto manipuliatoriaus prototipe privalo būti įdiegtas bevielio ryšio *Bluetooth* modulis.
- Roboto manipuliatoriaus prototipe privalo būti įdiegtas mikrovaldiklis.
- Duomenys *Bluetooth* ryšiu tarp roboto manipuliatoriaus ir išmaniojo telefono privalo būti siunčiami ne mažesniu kaip 19200 bit/s greičiu.

2.1.2. Roboto manipuliatoriaus nuotolinio valdymo sistemos funkciniai reikalavimai

Funkciniai reikalavimai kuriamai specializuotai išmaniojo telefono programiniai įrangai:

1. Operatorius privalo prisijungti prie roboto manipuliatoriaus arba atsijungti nuo roboto manipuliatoriaus bevieliu *Bluetooth* ryšiu.
2. Operatorius privalo įjungti arba išjungti visą roboto manipuliatoriaus valdymą vieno mygtuko paspaudimu.
3. Išmaniojo telefono ekrane privalo būti pateikiama informacija apie roboto manipuliatoriaus vykdomą komandą.
4. Išmaniojo telefono ekrane privalo būti pateikiama informacija apie dabartinę programinės įrangos būseną.
5. Specializuota išmaniojo telefono programinė įranga privalo nuskaityti duomenis iš akselerometrinio ir giroskopinio jutiklių.
6. Specializuota išmaniojo telefono programinė įranga nuolatos privalo stebėti ar operatorius yra prilietęs ekraną pirštu ar ne.
7. Išmaniojo telefono ekrane privalo būti pateikiama informacija apie akselerometrinio ir giroskopinio jutiklių duomenis.

8. Specializuotoje išmaniojo telefono programinėje įrangoje privalo būti suprogramuoti du roboto manipuliatoriaus valdymo režimai: pirmasis gestų valdymas – kada roboto manipuliatoriaus valdymas vykdomas tik operatoriaus rankų gestais, antrasis mišrus valdymas – kada roboto manipuliatoriaus valdymas vykdomas tiek operatoriaus rankų gestais, tiek paspaudžiant valdymo mygtuką išmaniojo telefono ekrane.
9. Operatorius privalo perjungti valdymo režimą vieno mygtuko paspaudimu.
10. Specializuota išmaniojo telefono programinė įranga privalo atpažinti užprogramuotus operatoriaus rankų gestus.
11. Mišriame valdymo režime, kada valdymui naudojami ir mygtukai ir rankų gestai, operatoriui privalo būti suteikta galimybė įsijungti tik roboto platformos arba tik roboto krano valdymą.
12. Operatorius privalo įjungti arba išjungti roboto manipuliatoriaus magnetą vieno mygtuko paspaudimu.
13. Specializuota išmaniojo telefono programinė įranga veikimo metu privalo vykdyti du lygiagrečius procesus: pirmasis procesas yra atsakingas už informacijos iš roboto manipuliatoriaus priėmimą, tolesnį apdorojimą ir perdavimą pagrindiniam programos procesui, antrasis pagrindinis procesas yra atsakingas už visą likusią programos kontrolę.
14. Specializuota išmaniojo telefono programinė įranga privalo sustabdyti roboto manipuliatoriaus veiksmus, jeigu išmaniojo telefono ekranas nėra priliestas pirštu.

Funkciniai reikalavimai kuriamai specializuotai roboto manipuliatoriaus programinei įrangai:

15. Specializuota roboto manipuliatoriaus programinė įranga privalo priimti valdymo komandas iš išmaniojo telefono bevieliu *Bluetooth* ryšiu.
16. Specializuota roboto manipuliatoriaus programinė įranga privalo siųsti informaciją į išmanųjį telefoną apie vykdomą komandą bevieliu *Bluetooth* ryšiu.
17. Specializuota roboto manipuliatoriaus programinė įranga privalo valdyti visus roboto manipuliatoriaus variklius – tiek platformos, tiek krano, atitinkamai pagal gautą komandą.
18. Specializuota roboto manipuliatoriaus programinė įranga privalo įjungti arba išjungti magnetą, atitinkamai pagal gautą komandą.

2.1.3. Panaudojimo atvejų diagramos

Panaudojimo atvejai kuriamai specializuotai išmaniojo telefono programinei įrangai:

1. Prisijungti prie roboto manipuliatoriaus.
2. Atsijungti nuo roboto manipuliatoriaus.
3. Įjungti roboto manipuliatoriaus valdymą.

4. Išjungti roboto manipuliatoriaus valdumą.
5. Įjungti roboto manipuliatoriaus platformos valdymą.
6. Įjungti roboto manipuliatoriaus krano valdymą.
7. Įjungti valdymą tik rankų gestais.
8. Išjungti valdymą tik rankų gestais.
9. Įjungti magnetą.
10. Išjungti magnetą.
11. Nuskaityti akselerometrinio ir giroskopinio jutiklių duomenis.
12. Nuskaityti ekrano prilietimo duomenis.
13. Atvaizduoti akselerometrinio ir giroskopinio jutiklių informaciją telefono ekrane.
14. Atvaizduoti vykdomą komandą telefono ekrane.
15. Atvaizduoti programos būseną telefono ekrane.
16. Atpažinti operatoriaus rankos gestą.
17. Atpažinti paspaustą valdymo mygtuką telefono ekrane.
18. Išsiųsti valdymo komandą į robotą manipuliatorių.
19. Priimti informaciją iš roboto manipuliatoriaus.
20. Sustabdyti robotą manipuliatorių.

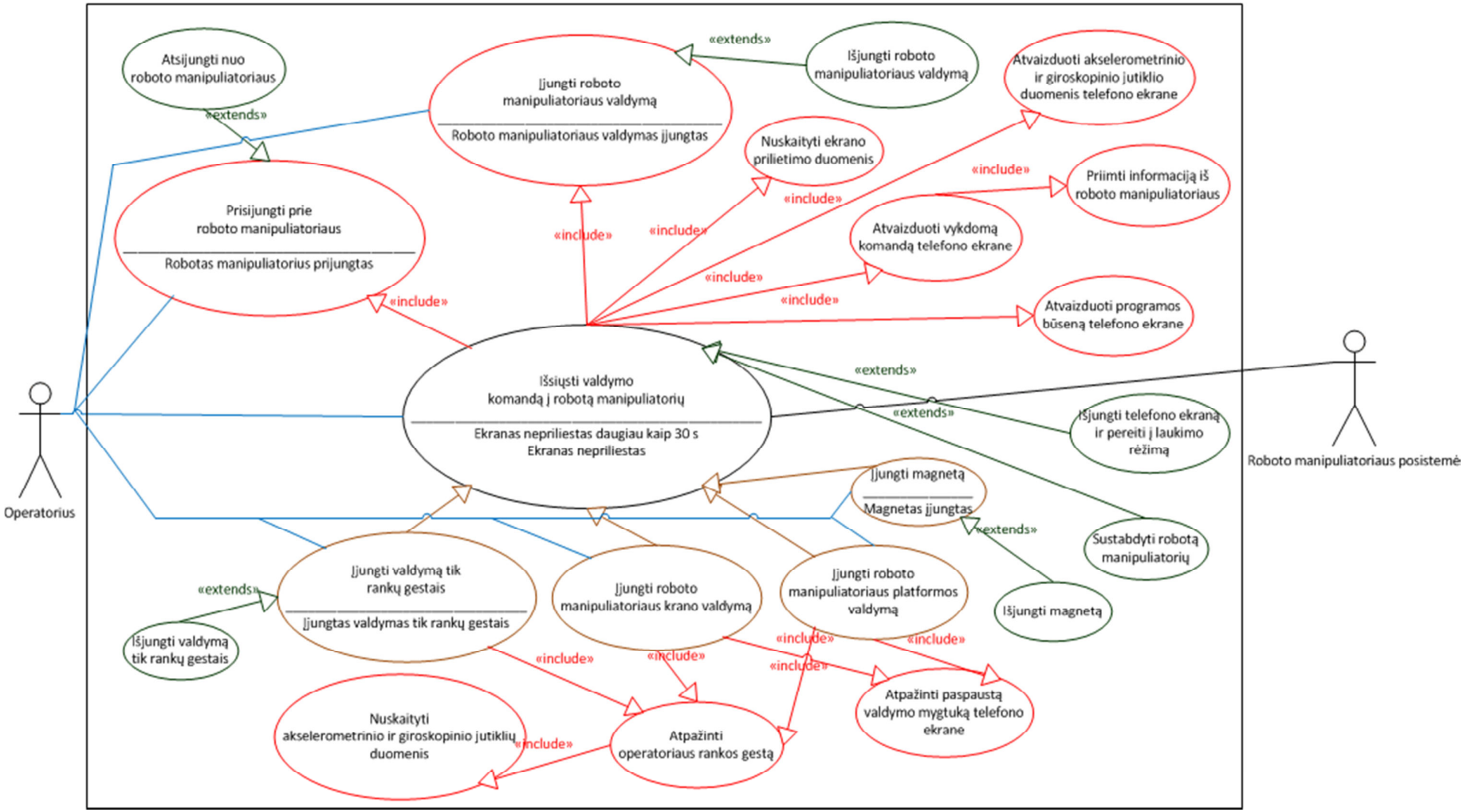
Panaudojimo atvejai kuriamai specializuotai roboto manipuliatoriaus programinei įrangai:

21. Priimti valdymo komandą iš išmaniojo telefono.
22. Atpažinti gautą valdymo komandą.
23. Siųsti informaciją apie vykdomą komandą į išmanųjį telefoną.
24. Valdyti reikiamus roboto manipuliatoriaus variklius.
25. Įjungti magneto elektros srovę.
26. Išjungti magneto elektros srovę.

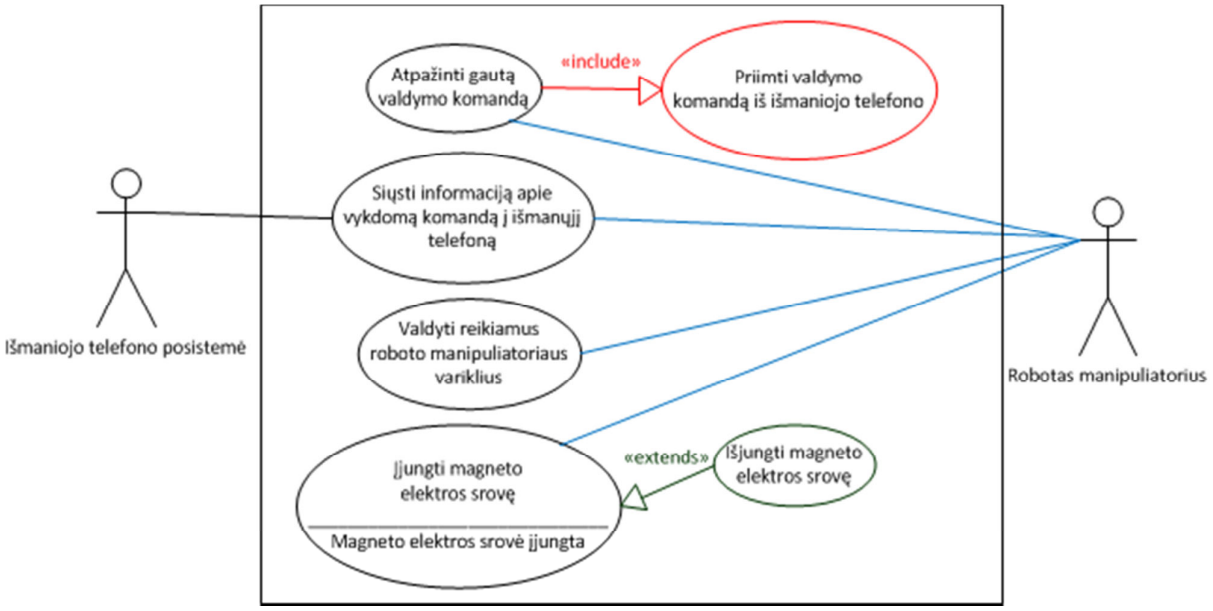
Panaudojimo atvejų ir funkcinių reikalavimų susietumo matrica pateikta 2 priede, 1 pav. Susietumo matrica sudaryta naudojant IBM Rational RequisitePro programinės įrangos paketą.

Kuriama nuotolinio valdymo sistema sudaryta iš dviejų dalių, kurios tarpusavyje keičiasi informacija. Pirmoji dalis – išmanusis telefonas, o antroji dalis robotas manipuliatorius (14 pav.). Kadangi kuriama nuotolinio valdymo sistema sudaryta iš dviejų dalių, panaudojimo atvejų diagramos yra pateikiamos kiekvienai sistemos daliai atskirai (15 ir 16 pav.).

14 pav. Kuriamos nuotolinio valdymo sistemos dalys



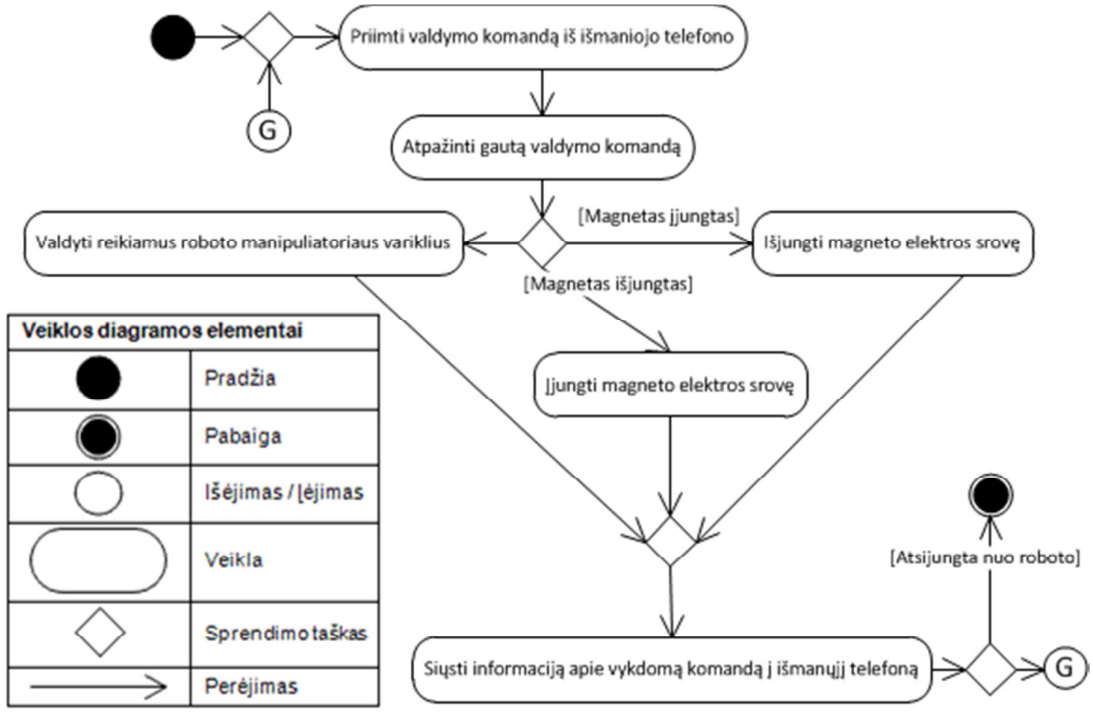
15 pav. Kuriamos specializuotos išmaniojo telefono programinės įrangos panaudojimo atvejų diagrama



16 pav. Kuriamos specializuotos roboto manipuliatoriaus programinės įrangos panaudojimo atvejų diagrama

2.1.4. Veiklos diagramos

Veiklos diagrama taip pat yra suskirstyta į kuriamos specializuotos išmaniojo telefono programinės įrangos veiklos diagramą (2 priedas, 2 pav.) ir kuriamos specializuotos roboto manipuliatoriaus programinės įrangos veiklos diagramą (17 pav.).



17 pav. Kuriamos specializuotos roboto manipuliatoriaus programinės įrangos veiklos diagrama

2.1.5. Sekos diagrama

Tam, kad būtų patogiau, sekos diagrama yra suskirstyta į atskirus veikimo etapus:

- Prisijungimas prie roboto manipuliatoriaus ir valdymo įjungimas (18 pav.);
- Roboto manipuliatoriaus valdymas tik gestais (19 pav.);
- Roboto manipuliatoriaus platformos valdymas mišriame valdymo režime (20 pav.);
- Roboto manipuliatoriaus krano valdymas mišriame valdymo režime (21 pav.);
- Atsijungimas nuo roboto manipuliatoriaus (22 pav.).

Sekos diagramoje naudojamos trys rodyklių rūšys:

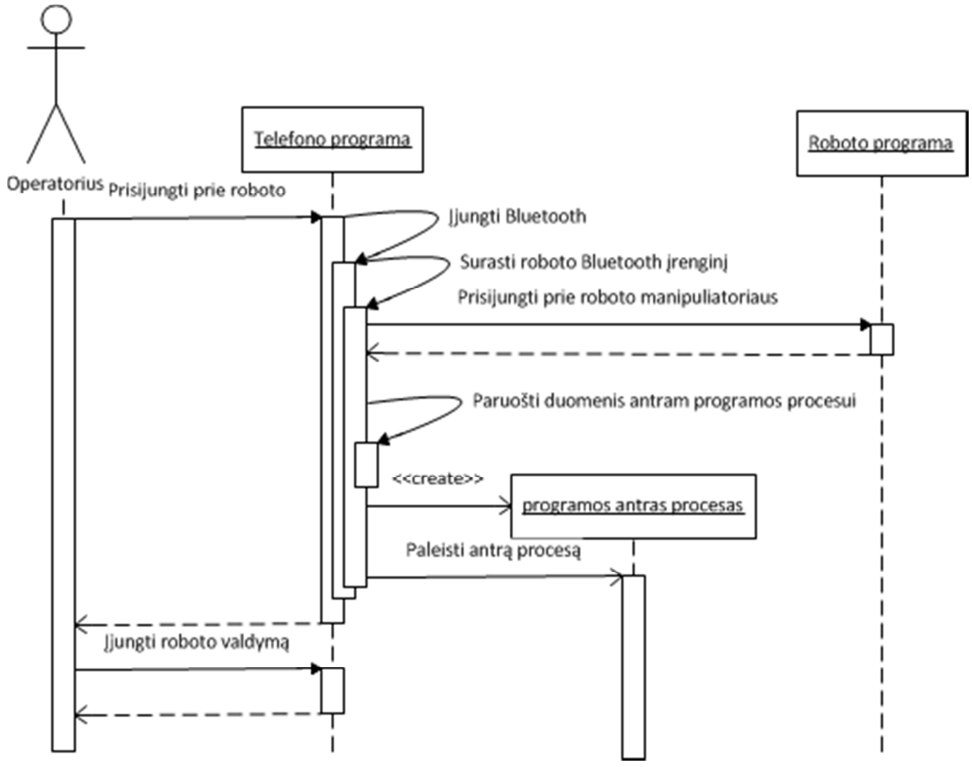
→ Sinchroninis kreipimasis;

→ Asinchroninis kreipimasis;

⇢ Grąžinimo rodyklė.

*Prisijungimo prie roboto manipuliatoriaus ir valdymo įjungimo sekos diagrama (*18 *pav.):*

Norėdamas valdyti robotą manipuliatorių operatorius pirmiausia turi prisijungti prie roboto manipuliatoriaus bevieliu *Bluetooth* ryšiu. Kai prisijungimas įvyksta, operatorius turi įjungti roboto manipuliatoriaus valdymą. Šie veiksmai įgyvendinami paspaudus atitinkamus mygtukus išmaniojo telefono ekrane.



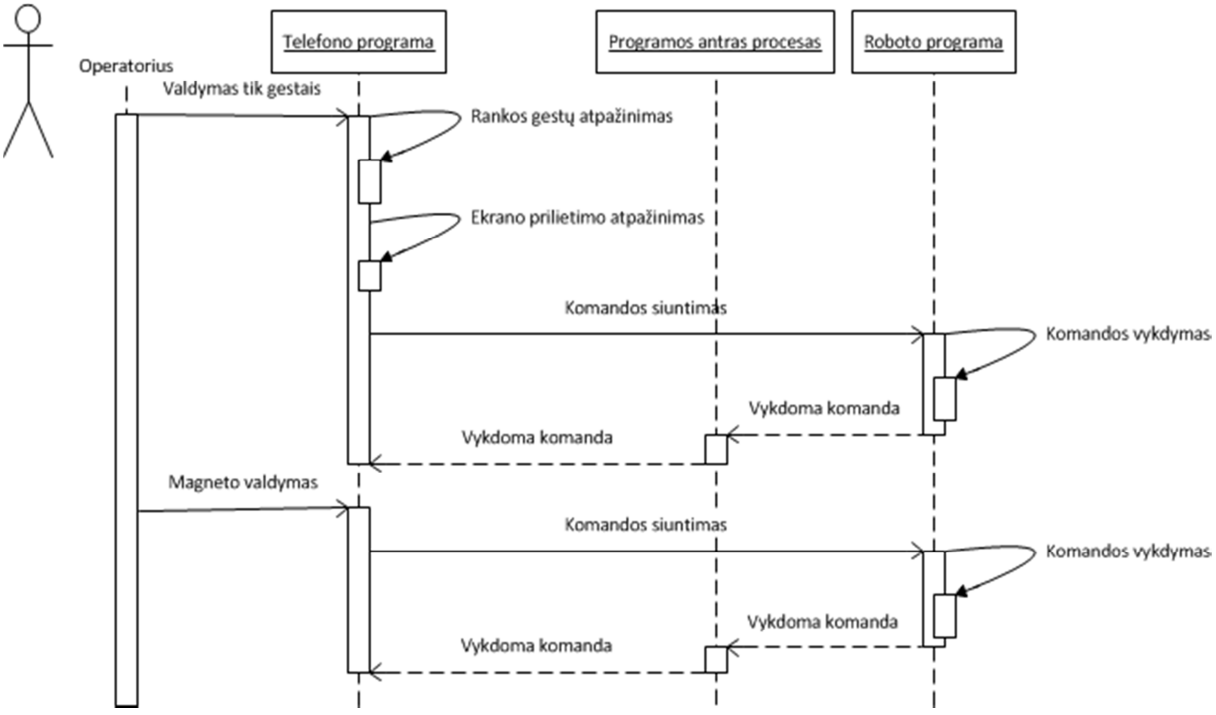
18 pav. Prisijungimo prie roboto manipuliatoriaus ir valdymo įjungimo sekos diagrama

Operatoriui paspaudus prisijungimo mygtuką išmaniojo telefono programa privalo aktyvuoti telefono *Bluetooth* bevielį ryšį bei surasti roboto manipuliatoriaus *Bluetooth* modulį. Išmaniojo

telefono programa aptikusi roboto manipuliatoriaus *Bluetooth* modulį privalo prisijungti prie įrenginio. Išmaniojo telefono programa atlikusi prisijungimą privalo paruošti duomenis antro programos proceso sukūrimui. Programa tai atlikusi privalo paleisti antrą programos procesą, kuris programos veikimo metu yra atsakingas už informacijos priėmimą iš roboto manipuliatoriaus per *Bluetooth* bevielį ryšį. Išmaniojo telefono programai prisijungus prie roboto manipuliatoriaus, operatorius privalo įjungti visą roboto manipuliatoriaus valdymą, paspausdamas atitinkamą mygtuką išmaniojo telefono ekrane, tam kad galėtų pasirinkti norimą valdymo režimą ir sėkmingai valdyti robotą manipuliatorių.

*Roboto manipuliatoriaus valdymo tik gestais sekos diagrama (*19 *pav.):*

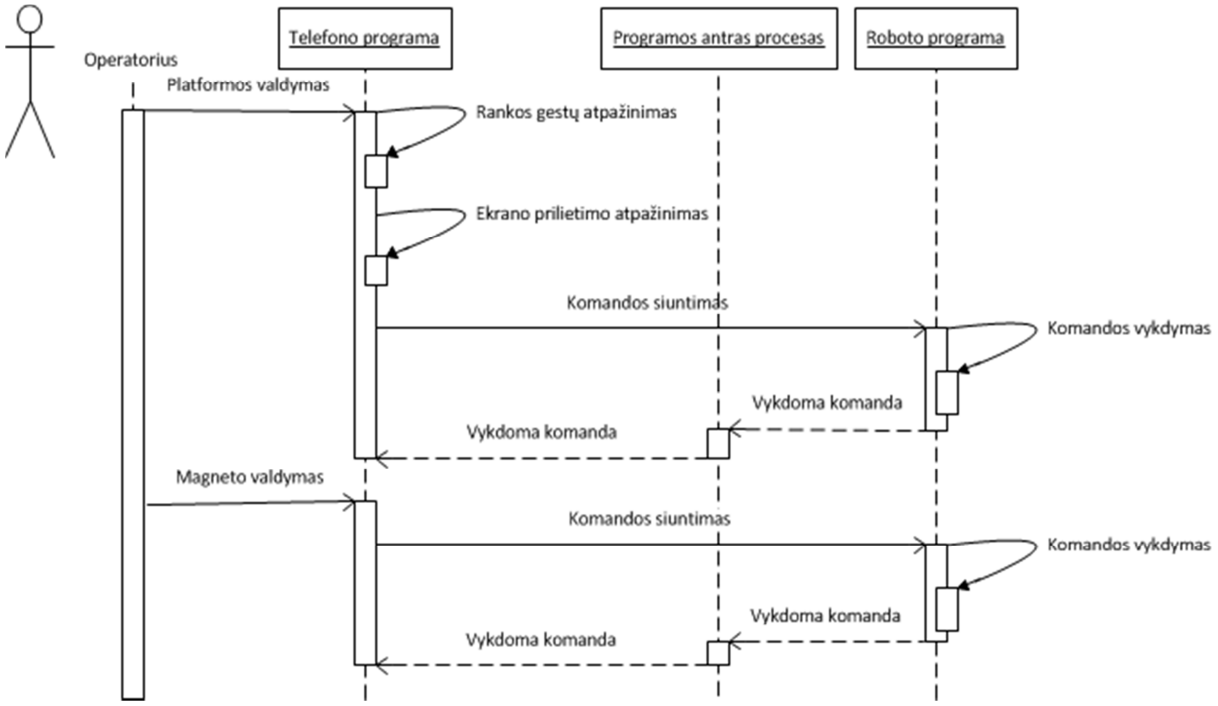
Šitame valdymo režime operatorius robotą manipuliatorių privalo valdyti naudodamas tik rankos gestus. Kiekvieną komandą atitinka tam tikras gestas, kurį privalo atlikti operatorius norėdamas robotui manipuliatoriui perduoti reikiamą komandą.



19 pav. Roboto manipuliatoriaus valdymo tik rankų gestais sekos diagrama

*Roboto manipuliatoriaus platformos valdymo sekos diagrama (*20 *pav.):*

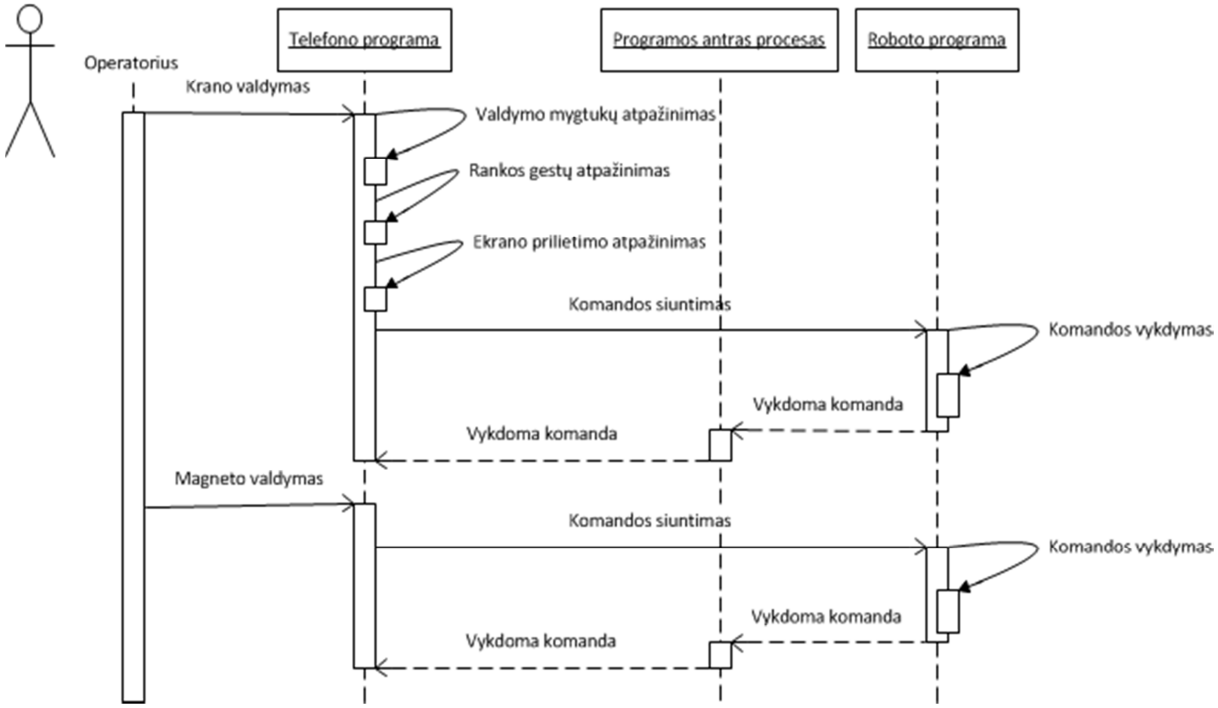
Operatorius mišriame valdymo režime privalo pasirinkti roboto manipuliatoriaus platformos arba krano valdymą.



20 pav. Roboto manipuliatoriaus platformos valdymo mišriame valdymo režime sekos diagrama

*Roboto manipuliatoriaus krano valdymo sekos diagrama (*21 *pav.):*

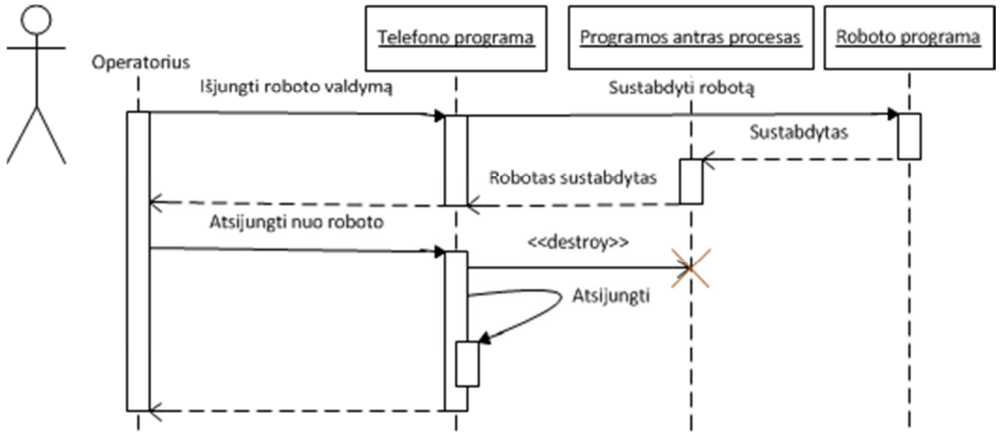
Operatorius valdydamas kraną mišriame valdymo režime turi naudoti tiek rankos gestus, tiek valdymo mygtukus.



21 pav. Roboto manipuliatoriaus krano valdymo mišriame valdymo režime sekos diagrama

Atsijungimo nuo roboto manipuliatoriaus sekos diagrama (22 pav.):

Baigęs dirbti su robotu manipuliatoriumi operatorius privalo atsijungti nuo roboto manipuliatoriaus.



22 pav. Atsijungimo nuo roboto manipuliatoriaus sekos diagrama

2.1.6. Roboto manipuliatoriaus nuotolinio valdymo sistemos nefunkciniai reikalavimai

Nefunkciniai reikalavimai:

Kuriamai specializuotai išmaniojo telefono programinei įrangai:

1. Specializuota išmaniojo telefono programinė įranga privalo būti sukurta taip, kad visi valdymo mygtukai tilptų viename lange.
2. Specializuota išmaniojo telefono programinė įranga privalo būti sukurta taip, kad operatoriui būtų patogu naudotis visomis funkcijomis, t.y. visi valdymo mygtukai būtu išdėstyti patogiai.
3. Specializuota išmaniojo telefono programinė įranga privalo būti sukurta taip, kad bet kuri roboto manipuliatoriaus valdymo komanda būtų vykdoma tik tada, kai operatorius liečia išmaniojo telefono ekraną pirštu. Tai yra būtinas saugumo reikalavimas.

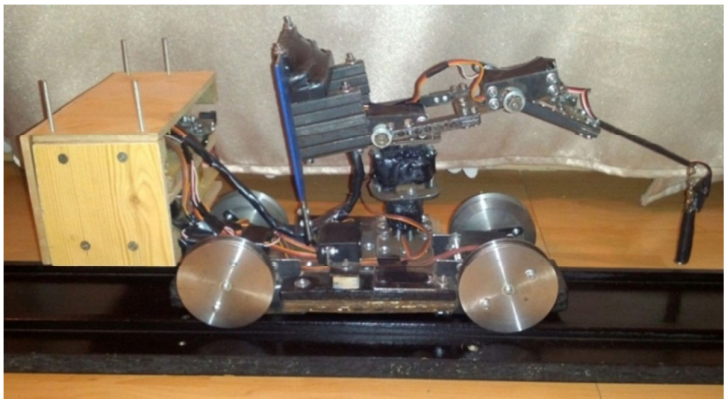
Kuriamam roboto manipuliatoriaus prototipui:

4. Roboto manipuliatoriaus platforma privalo būti varoma visais keturiais ratais.
5. Roboto manipuliatoriaus platformoje privalo būtį įmontuota pozicionavimo svirtis vagonų sąstatui perstumti.
6. Robote manipuliatoriuje privalo būti sumontuotas dviejų ašių kranas, prie kurio galo pritvirtintas magnetas metalo laužui krauti.

III. PRAKTINĖ DALIS

3.1. Roboto manipuliatoriaus prototipas ir jo programinė įranga

Šiame darbe buvo sukurtas roboto manipuliatoriaus prototipas, skirtas metalo laužo krovimui iš traukinių vagonų ir į juos bei traukinių vagonų pozicionavimui (23 pav.).



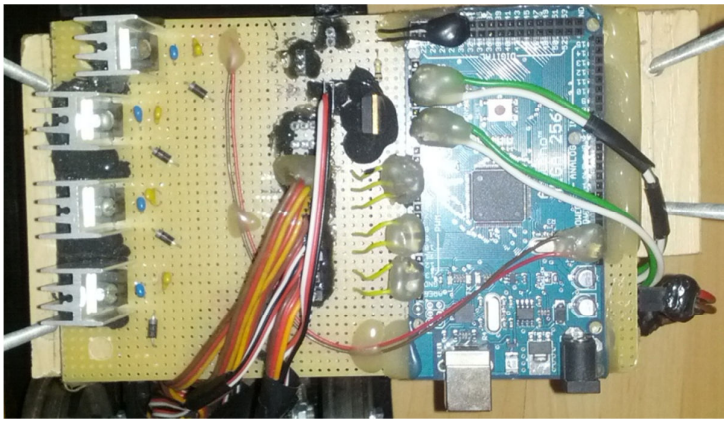
23 pav. Roboto manipuliatoriaus prototipas

Roboto manipuliatoriaus prototipas sumontuotas ant platformos, kuri yra varoma visais keturiais ratais. Kiekvienas ratas yra varomas atskiru modifikuotu servo varikliu. Pats robotas manipuliatorius gali judėti tik ten kur yra nutiesti bėgiai. Ant roboto manipuliatoriaus platformos yra sumontuota traukinių vagonų pozicionavimo svirtis, skirta traukinio sąstato pozicionavimui. Svirtis yra valdoma servo variklio. Taip pat ant roboto platformos yra sumontuotas dviejų ašių kranas, skirtas metalo laužo krovimui. Patį kraną valdo keturi servo varikliai. Prie krano galo yra pritvirtintas magnetas skirtas metalo laužo krovimui. Žemiau nagrinėjamos roboto manipuliatoriaus judėjimo galimybės pažymėtos numeriais nuo 1 iki 5 (24 pav.).

24 pav. Roboto manipuliatoriaus prototipas: 1 – slenkamasis judėjimas; 2 – sukamasis judėjimas; 3 – sukamasis judėjimas; 4 – sukamasis judėjimas; 5 – sukamasis judėjimas

Roboto manipuliatoriaus platforma gali judėti bėgiais pirmyn ir atgal (24 pav. Nr. 1). Pozicionavimo svirtis gali būti arba pakelta arba nuleista. Svirtis gali judėti sukamuoju judėjimu (24 pav. Nr. 5). Roboto manipuliatoriaus kranas gali suktis aplink savo ašį sukamuoju judėjimu (24 pav. Nr. 4), o dvi krano ašys gali judėti aukštyn arba žemyn taip pat sukamuoju judėjimu (24 pav. Nr. 2, 3).

Visus roboto manipuliatoriaus veiksmus valdo *Arduino Mega 2560* valdymo blokas, kuriame įmontuotas *Atmega 2560* mikrovaldiklis. *Arduino Mega 2560* valdymo blokas sumontuotas ant specialios montažinės plokštės, tam, kad būtų galima prijungti roboto manipuliatoriaus variklius (25 pav.).



25 pav. Roboto manipuliatoriaus valdymo blokas su Atmega 2560 mikrovaldikliu

Be pagrindinio mikrovaldiklio naudojamas dar ir papildomas *Atmega 328* mikrovaldiklis, kuris yra atsakingas už dviejų roboto platformos variklių valdymą. Abu mikrovaldikliai tarpusavyje bendrauja serijiniu rx, tx ryšiu. Tam, kad robotas manipuliatorius galėtų priimti ir siųsti valdymo komandas iš ir į išmanųjį telefoną, prie mikrovaldiklio *Atmega 2560* yra prijungtas *Bluetooth* modulis BTM 222. *Bluetooth* modulis prie mikrovaldiklio taip pat prijungtas naudojant serijinį rx, tx ryšį.

Sukurto roboto manipuliatoriaus prototipo programinė įranga:

Programinis kodas roboto manipuliatorio mikrovaldikliams yra parašytas naudojant *Arduino* atvirojo kodo programinę įrangą. Programuojant mikrovadiklį su *Arduino* programinę įranga pirmiausia yra aprašomi visi kintamieji, kurie bus naudojami tolimesmėje programos eigoje. Taip pat pradžioje yra įtraukiamos reikalingos bibliotekos. Šiuo atveju į kodą yra įtraukiama už servo variklių valdymą atsakinga biblioteka („*#include <Servo.h>*“). Aprašius visus reikalingus kintamuosius yra nustatomi pagrindiniai programos parametrai:

```
1. void setup(){
2.       Serial1.begin(19200);
3.       Serial3.begin(19200);
4.       kranas_variklis1.attach(8);
```

```
5.      kranas_variklis2.attach(9);
6.      kranas_variklis3.attach(12);
7.      kranas_variklis4.attach(13);
8.      platforma_variklis3.attach(2);
9.      platforma_variklis4.attach(3);
10.     platforma_variklis5.attach(4);
11.     pinMode(magnet, OUTPUT);
12.}
```

Antroje ir trečioje kodo eilutėje yra inicijuojami serijiniai ryšio kanalai. Vienas ryšio kanalas reikalingas tam, kad mikrovaldiklis galėtų bendrauti su bluetooth moduliu, kitas ryšio kanalas reikalingas tam, kad mikrovaldiklis galėtų bendrauti su papildomu mikrovaldikliu. Toliau nuo ketvirtos iki dešimtos kodo eilutės yra inicijuojamos ir priskiriamos atitinkamos mikrovaldiklio kojelės servo variklių valdymui. Vienuoliktoje kodo eilutėje vienas mikrovaldiklių išvadų yra pažymimas kaip išeinantis. Vėliau šis išvadas bus naudojamas magneto valdymui.

Baigus inicijuoti pagrindinius programos nustatymus, prasideda pagrindinis programos ciklas. Žemiau nagrinėjamos esminės ciklo dalis. Pirmiausia ciklo pradžioje programa priima valdymo komadas iš sukurtos specializuotos išmaniojo telefono programinės įrangos:

```
1. if (Serial3.available() > 0) {
2.      IncomingByte = Serial3.read();
3.      serial_gauta_komanda = true;
4.      if ((IncomingByte == 'a' || IncomingByte == 'b') || IncomingByte == 's'){
5.           Serial1.println(IncomingByte);
6.      }
7. }
```

Pirma kodo eilutė patikrina ar gauta valdymo komanda. Jeigu duomenys gauti – nuskaitoma gauta valdymo komanda (antra kodo eilutė). Ketvirtoje eilutėje yra tikrinama dar viena sąlyga. Jeigu sąlyga teisinga valdymo komanda yra persiunčiama į papildomą mikrovaldiklį.

Kai valdymo komanda jau yra gauta toliau yra vykdomas atitinkamas valdymo komandos programinis kodas. Visos valdymo komandos suprogramuotos panašiu principu, todėl plačiau nagrinėjamos tik keletę iš jų, o visos likusios pateiktos 3 priede kartu su visu programiniu kodu. Žemiau yra nagrinėjama valdymo komanda, kada roboto platforma važiuoja į kairę pusę;

```
1. if (IncomingByte == 'a'){
2.      platforma_value3 = 91;
3.      platforma_value4 = 92;
4.      if (serial_gauta_komanda){
5.           vykdoma_komanda = 'a';
6.           serial_send_to_phone = true;
7.      }
8. }
```

Kiekviena valdymo komanda yra pažymėta atitinkama raide. Šiuo atveju raidė „a“ nurodo, kad roboto manipuliatoriaus platforma turi važiuoti į kairę pusę. Antra ir trečia kodo eilutės nurodo, kad du platformos servo varikliai turi lėtai suktis į kairę pusę. Likusius du platformos servo variklius valdo papildomas mikrovaldiklis, kuriam jau buvo perduota komanda, kad platforma turi

važiuoti į kairę pusę. Penkta ir šešta eilutės paruošia duomenis siuntimui į išmanųjį telefoną. Roboto manipuliatoriaus platformos sustabdymas ir judėjimas į dešinę pusę apsirašo labai panašiai. Žemiau nagrinėjamas platformos svirties nuleidimo kodo fragmentas:

```
1. if (IncomingByte == 'c'){
2.       Serial1.println('s');
3.       platforma_value3 = 93;
4.       platforma_value4 = 90;
5.       if (platforma_value5 == 116){
6.            nuleidziam_svirti = true;
7.       }
8. }
9. if (nuleidziam_svirti == true){
10.       if (platforma_value5 > 24){
11.            platforma_value5 = platforma_value5 - 1;
12.            delay(50);
13.       }
14.       vykdoma_komanda = 'c';
15.       serial_send_to_phone = true;
16.       if (platforma_value5 == 24){
17.            nuleidziam_svirti = false;
18.            vykdoma_komanda = 's';
19.            serial_send_to_phone = true;
20.       }
21.}
```

Kodo pradžioje yra sustabdomas platformos judėjimas. Toliau yra patikrinama ar svirtis yra pakelta. Jeigu svirtis pakelta pradeda svirties nuleidimo procedūra. Kadangi svirtį valdo servo variklis, tai norint lėtai nuleisti svirtį reikia į variklį valdymo signalą paduoti kas tam tikrą laiko tarpą, ką ir atlieka tolimesnis kodas, tol kol yra pasiekiama servo variklio sustabdymo riba. Paskutinėse kodo fragmento eilutėse yra paruošiami duomenys išsiuntimui per *Bluetooth* į išmanųjį telefoną. Svirties pakėlimą aprašo labai panšus kodo fragmentas, kaip ką tik nagrinėtas.

Svarbu paminėti magneto valdymą. Jeigu magnetas yra išjungtas ir gaunama magneto įjungimo komanda, tada mikrovaldilis paleidžia elektros srovę per magnetą. Tam veiksmui atlikti yra specialai išskirtas vienas mikrovaldiklio išvadas, kuris jau buvo paminėtas šiek tiek ankščiau. Jeigu gaunama magneto išjungimo komanda tada mikrovaldiklis tiesiog sustabdo elektros srovės tekėjimą per magnetą.

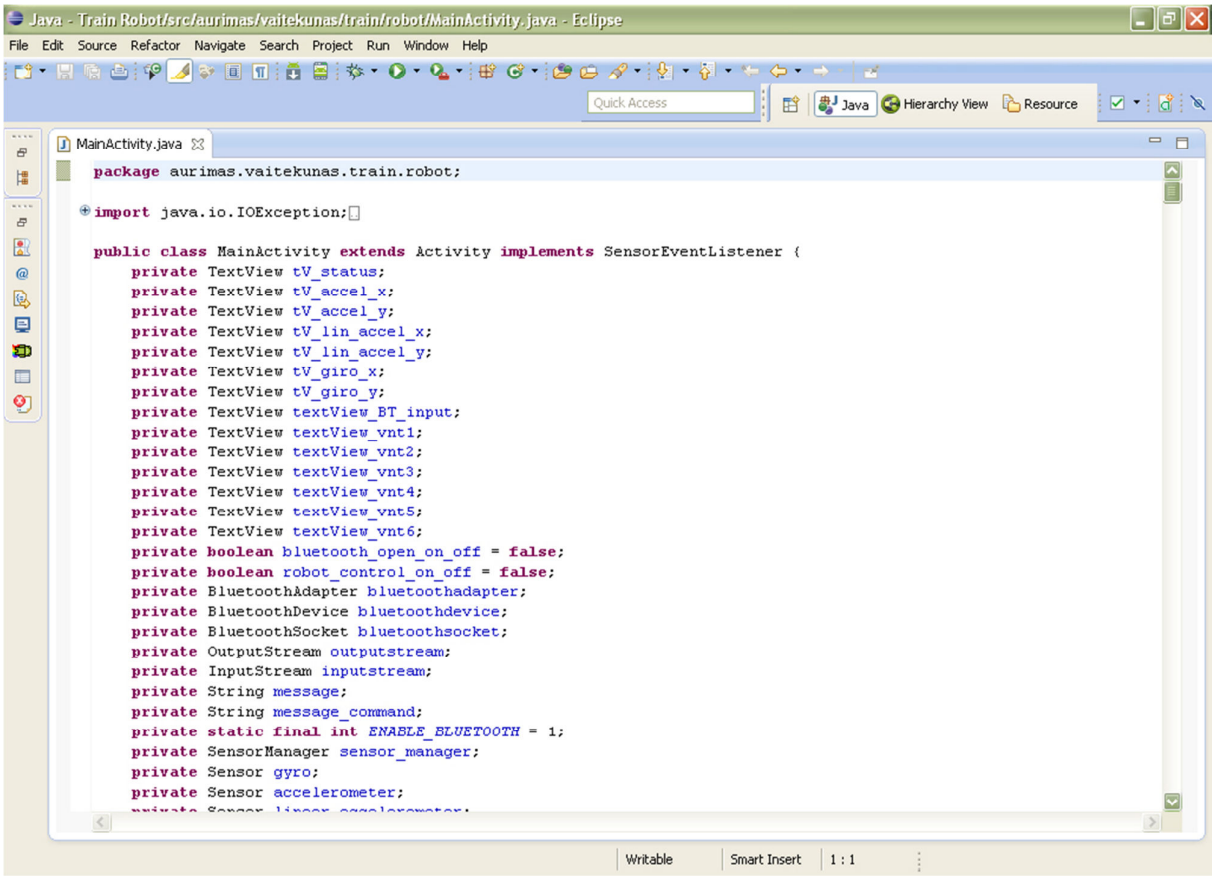
Krano varikliai valdomi labai panašiu principu kaip ir svirties variklis. Gavus krano valdymo komandą į krano variklius yra siunčiamas atitinkamas signalas, pagal kurį krano variklis juda reikiam kryptimi. Atitinkamai pagal gautą komandą programinė įranga išsiunčia valdymo signalą į reikiamą roboto manipuliatoriaus variklį. Kiekvienas servo variklis gavęs tik jam skirtą valdymo signalą pajuda reikiama kryptimi. Visų variklių greitis reguliuojamas arba keičiant paduodamą valdymo signalą arba valdymo signalas yra paduodamas į variklius kas tam tikrą laiko tarpą.

Galiausiai roboto manipuliatoriaus programa išsiunčia informaciją apie šiuo metu vykdomą komandą sukurtai specializuotai išmaniojo telefono programinei įrangai per *Bluetooth* bevielio ryšio modulį. Visas abiejų mikrovaldiklių programinis kodas pateiktas 3 priede.

3.2. Specializuota išmaniojo telefono programinė įranga

3.2.1. Programinės įrangos kūrimo aplinka – Eclipse

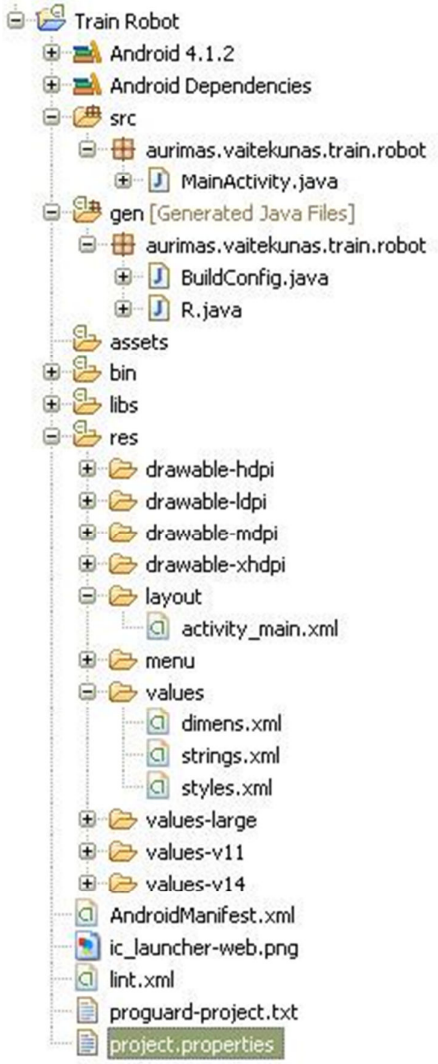
Šiame darbe programinė įranga *Android* OS sukurta naudojant *Eclipse* programinės įrangos kūrimo aplinką. *Eclipse* yra atvirojo kodo integruota kūrimo aplinka. *Eclipse* susideda iš daugelio įskiepiu, kurių kiekvienas gali turėti savo langus, savo meniu punktus, savo nepriklausomą nuotolinio atnaujinimo sistemą. Programa skirta nuotoliniam roboto manipuliatoriaus valdymui, sukurta naudojant šiuo metu naujausią *Eclipse Juno* versiją, skirtą išmaniųjų telefonų programuotojams. Tam, kad su *Eclipse* būtų galima kurti programas *Android* OS, į *Eclipse* reikėjo papildomai įdiegti specialų įskiepį – *Android* kūrimo įrankius. Įdiegus įskiepį, taip pat buvo būtina dar įdiegti *Android* programinės įrangos kūrimo rinkinį. Padarius paminėtus veiksmus *Eclipse* buvo pilnai paruošta kurti programas skirtas *Android* OS (26 pav.).



26 pav. Eclipse programinės įrangos kūrimo aplinkos langas

3.2.2. Išmaniojo telefono roboto manipuliatoriaus valdymo programa

Kiekviena *Android* OS programa turi tam tikrą bendrą programos struktūrą (27 pav.).



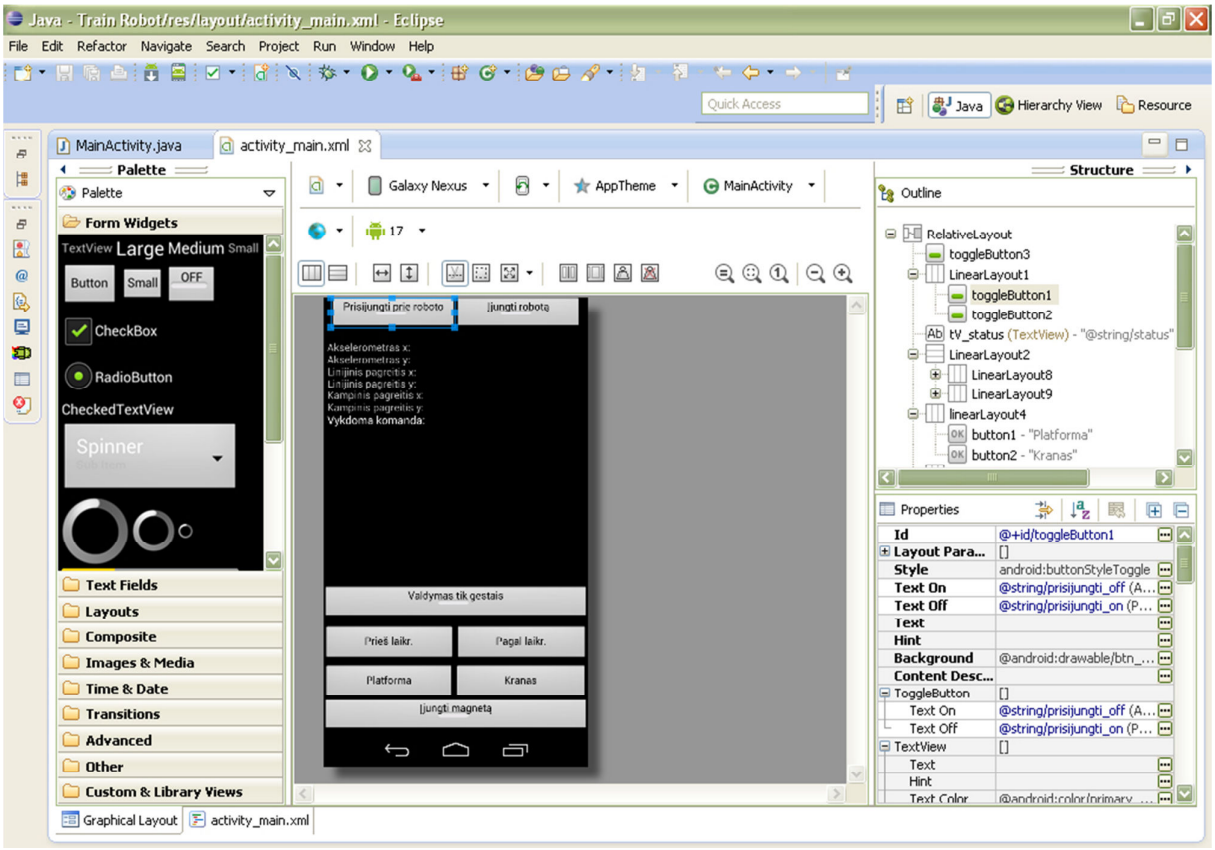
27 pav. Android programos struktūra

Svarbiausios struktūros dalys yra *src*, *res* aplankalai bei *AndroidManifest.xml* failas. *Src* aplankale yra talpinamos visos programos projekto klasės. Šiuo atveju pagrindinė programos klasė yra *MainActivity.java*. *Res* aplankale yra talpinama informacija apie programos grafinę sąsają, programoje naudojamas nuotraukas, statinius žodžius, spalvas ir kitą panašią informaciją. *Res* aplankas yra ta vieta kur saugomi visi programoje naudojami resursai. *AndroidManifest.xml* failas yra pagrindinis programos nustatymų failas. Šiame faile apsirašo visa būtina informaciją susyjusi su programos nustatymais. Pavyzdžiui leidimų suteikimas programai naudoti tam tikrus operacinės sistemos resursus. *Gen* ir *bin* aplankai yra automatiškai sukuriami programos kompiliavimo metu. Šiuose aplankuose saugoma informacija apie jau sukompiliuotą programą. *Android 4.1.2*, *Android Dependencies*, *libs* aplankalai savyje saugo įvairias klases reikalingas programos kūrimui. *Assets*

aplankalas skirtas talpinti papildomus failus ar nuotraukas reikalingas programos veikimui. Pavyzdžiui jeigu būtina programoje naudoti tekstinius failus, tada tokie failai bus talpinami būtent į *assets* aplankalą. *Proguard-project.txt* ir *Project.properties* failuose apsirašo papildoma informacija apie programos savybes bei programos saugumo nustatymai. *Lint.xml* failas yra susijęs su programos klaidų tikrinimu.

Sukurtos išmaniojo telefono programinės įrangos vartotojo grafinės sąsajos failas activity_main.xml:

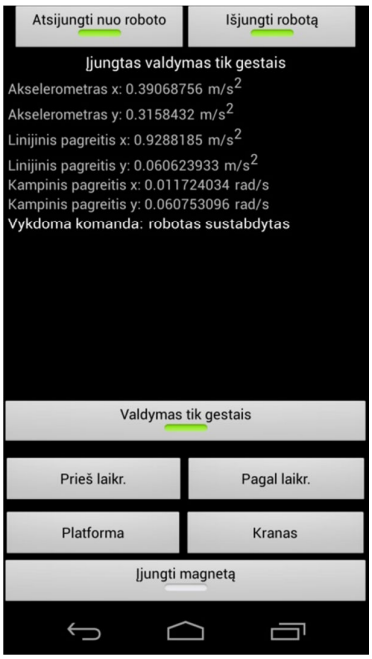
Šiame faile yra aprašoma visa informacija, susijusi su programos grafine sąsaja (28 pav.).



28 pav. Activity_main.xml failo langas Eclipse programoje

Kairėje lango pusėje (*Palette*) yra išdėstyti įvairūs komponentai, kuriuos galima panaudoti savo programoje. Šiuo atveju kuriant šitą programą buvo panaudoti teksto, mygtukų, išdėstymo schemos komponentai. Viršutinėje dešinėje lango pusėje (*Outline*) yra išdėstyti programoje naudojami komponentai. Tai programoje naudojamų komponentų medis. Dešinėje apatinėje lango pusėje (*Properties*) yra aprašomos kiekvieno naudojamo komponento savybės, parametrai. *Eclipse* programa yra patogi todėl, kad išmaniojo telefono programai vartotojo grafinę sąsają galima kurti tiek naudojant *Eclipse* grafinę sąsaja (28 pav.), tiek tiesiogiai rašant programos kodą. Priklausomai nuo poreikių išmaniojo telefono programos vartotojo grafinei sąsajai sukurti naudojamas tiek

Eclipse grafinės sąsajos langas, tiek tiesioginis kodo rašymas. Žemiau pateiktas vaizdas, kaip atrodo sukurta specializuota išmaniojo telefono programa (29 pav.).



29 pav. Sukurtos specializuotos išmaniojo telefono programos lango vaizdas

Kuriant šios programos grafinę sąsają buvo naudojamos trys pagrindinės komponentų išdėstymo schemos. Kaip pagrindinė programos išdėstymo schema buvo pasirinkta reliatyvi schema. Tai reiškia, kad pasirinktus programos komponentus, kaip kad tekstą ar mygtukus galima laisvai dėlioti išmaniojo telefono ekrane. Toliau yra pateiktas šios išdėstymo schemos fragmentas:

```
1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2. xmlns:tools=http://schemas.android.com/tools
3. android:layout_width="match_parent"
4. android:layout_height="match_parent" >
5. </RelativeLayout>
```

Pirma ir antra kodo eilutės nurodo kokius specifinius xml dokumento elementus naudoja programa, o trečia ir ketvirta kodo eilutės pažymi, kad reliatyvi komponentų išdėstymo schema naudojama per visą išmaniojo telefono ekraną.

Be reliatyvaus išdėstymo, taip pat naudojamas vertikalus bei horizontalus komponentų išdėstymas. Žemiau pateiktas vienas iš naudojamų vertikalaus programos komponentų išdėstymo kodo fragmentų:

```
1. <LinearLayout
2. android:id="@+id/LinearLayout2"
3. android:layout_width="match_parent"
4. android:layout_height="wrap_content"
5. android:layout_alignParentLeft="true"
6. android:layout_below="@+id/tV_status"
7. android:gravity="left"
8. android:orientation="vertical" >
```

Šiame kodo fragmente antra eilutė aprašo unikalų šio išdėstymo id numerį. Šis unikalus id numeris reikalingas tam, kad esant poreikiui šį kodo fragmentą būtų galima naudoti programuojant java klasės failą. Šiuo atveju programuojant *MainActivity.java* failą. Kiekvienas programos grafinėje sąsajoje naudojamas komponentas turi savo unikalų id numerį, nesvarbu ar tai mygtukas, ar teksto komponentas ar išdėstymo komponentas. Trečia kodo eilutė nurodo, kad šis vertikalus išdėstymo komponentas eina per visą ekrano plotį. O ketvirta eilutė pažymi, kad šis išdėstymo komponentas apgaubia jame esantį turį, pavyzdžiui mygtuką. Penkta, šešta, septinta eilutės parodo kaip šis komponentas yra išdėstytas telefono ekrane ir galiausiai aštunta eilutė nurodo, kad tai yra būtent vertikalaus išdėstymo komponentas. Horizontalaus išdėstymo komponentas yra analogiškas vertikaliam, tik šiuo atveju visi komponentai telefono ekrane išdėstomi, ne vertikaliai bet horizontaliai. Kartu maišant tiek vertikalaus, tiek horizontalaus išdėstymo stilius, pasiekiamas reikiamas rezultatas.

Sukurtoje programoje naudojami tiek paprasti mygtukai, tiek perjungimo tipo mygtukai. Žemiau pateiktas vienas iš kelių naudojamų perjungimo tipo mygtukų kodo fragmentas:

```
1. <ToggleButton
2. android:id="@+id/toggleButton1"
3. android:layout_width="170dp"
4. android:layout_height="wrap_content"
5. android:gravity="center_vertical|center_horizontal"
6. android:onClick="PrisijungtiPrieRoboto"
7. android:textOff="@string/prisijungti_on"
8. android:textOn="@string/prisijungti_off" />
```

Kodo fragmento pradžioje vėlgi nurodytas komponento id numeris, mygtuko plotis ir aukštis (trečia, ketvirta eilutės), mygtuko vieta telefono ekrane (penkta eilutė). Šešta eilutė nurodo, kokį metodą turi iškviesti programa, kada mygtuką paspaudžia operatorius. Šiuo atveju paspaudus mygtuką programa iškvies „PrisjungtiPrieRoboto“ metodą, kuris atliks tam tikras operacijas. Septinta ir aštunta eilutės nurodo, kas turi būti parašyta ant mygtuko. Šiuo atveju mygtukas turi dvi skirtingas fazes – įjungtas ir išjungtas, tad yra pateikiami ir du skirtingi mygtuko užrašai. Visi programos pradiniai tekstai naudojami ant mygtukų ar tiesiog telefono ekrane yra saugomi atskirame resursų faile *strings.xml*. Septintoje ir aštuntoje eilutėse būtent ir yra pateikiamos nuorodos į šį teksto resursų failą, kuriame yra aprašomi tekstiniai sakiniai, žodžiai naudojami programoje (1 priedas, 21 pav.). Programos vykdymo metu visus tekstus galima keisti pagal poreikį, tad *strings.xml* faile yra aprašomos pradinės teksto išraiškos, kurios panaudojamos programos paleidimo metu.

Be jau aptarto perjungimo tipo mygtuko, sukurtoje programoje taip pat naudojami paprasti mygtukai. Šie mygtukai apsirašo labai panašiu kodu kaip ir perjungimo tipo mygtukai.

Galiausiai paskutinis vartotojo grafinės sąsajos elementas, kuris naudojamas sukurtoje programoje yra tekstiniai komponentai. Žemiau pateiktas vienas iš kelėtos naudojamų tekstinių komponentų kodo fragmentų:

```
1. <TextView
2. android:id="@+id/tV_accel_x"
3. android:layout_width="wrap_content"
4. android:layout_height="wrap_content"
5. android:paddingLeft="5dp"
6. android:paddingRight="0dp"
7. android:text="@string/akselerometras_x"
8. android:textAppearance="?android:attr/textAppearanceSmall" />
```

Pirmose kodo eilutėse vėl gi aprašytas unikalus tekstinio komponento id numeris. Toliau nurodytas tekstinio komponento plotis ir aukštis. Šiuo atveju tekstinis komponentas apgaubia ir prisitaiko prie jame įrašyto teksto. Penktoje ir šeštoje eilutėje aprašytas komponento atitraukimas nuo kitų programos komponentų iš kairės ir dešinės pusės. Septintoje eilutėje nurodytas naudojamas tekstinis elementas iš strings.xml failo. Aštuntoje eilutėje aprašyti teksto nustatymai, šiuo atveju pažymėta, kad teksto dydis yra mažas.

Išanalizuoti pagrindiniai sukurtos specializuotos išmaniojo telefono programos grafinės sąsajos komponentai, kurie buvo panaudoti kuriant programos vartotojo grafinę sąsają. Visas sukurtos išmaniojo telefono programos vartotojo grafinės sąsajos *activity_main.xml* failo programinis kodas pateiktas 4 priede.

Sukurtos specializuotos išmaniojo telefono programos pagrindinės klasės failas MainActivity.java:

Šiame faile yra aprašoma pagrindinė sukurtos specializuotos programos logika. Žemiau pateiktos sukurtos programos naudojamos papildomos klasės ir sąsajos:

- *java.io.IOException* – ši klasė yra atsakinga už išimčių apdorojimą atsirandančių dėl signalo įėjimo ar išėjimo klaidų.
- *java.io.InputStream* – ši klasė yra susijusi su baitų įvesties srautu. Naudojama priimant duomenis per *Bluetooth*.
- *java.io.OutputStream* – ši klasė yra susijusi su baitų išvesties srautu. Naudojama siunčiant duomenis per *Bluetooth*.
- *java.util.Set* – ši sąsaja atsakinga už tai, kad tam tikra kolekcija elementų netūrėtų pasikartojančių elementų. Naudojama programuojant *Bluetooth* įrenginį.
- *java.util.UUID* – ši klasė reprezentuoja nekintamą universalų unikalų identifikatorių. Šis unikalus identifikatorius naudojamas programuojant *Bluetooth* įrenginį.
- *android.app.Activity* – tai klasė atsakinga už vartotojui matomo lango telefono ekrane sukūrimą ir palaikymą.

- *android.bluetooth.BluetoothAdapter* - tai klasė atsakinga už telefono *Bluetooth* įrenginio valdymą.
- *android.bluetooth.BluetoothDevice* - tai klasė atsakinga už rastų kitų *Bluetooth* įrenginių valdymą. Pavyzdžiui gali pateikti informaciją apie rastą kitą *Bluetooth* įrenginį.
- *android.bluetooth.BluetoothSocket* - tai klasė atsakinga už *Bluetooth* jungties sukūrimą tarp dviejų bluetooth įrenginių.
- *android.content.Intent* - tai klasė padedanti atlikti tam tikras operacijas, perduoti tam tikrą informaciją. Šiuo atveju naudojama programuojant *Bluetooth* įrenginį.
- *android.hardware.Sensor* - tai klasė atsakinga už išmaniojo telefono jutiklius.
- *android.hardware.SensorEvent* - tai klasė atsakinga už jutiklių informacijos pateikimą.
- *android.hardware.SensorEventListener* - tai klasė atsakinga už pranešimų gavimą, kada pasikeičia tam tikro jutiklio reikšmė.
- *android.hardware.SensorManager* - tai klasė, kuri leidžia programai naudoti išmaniojo telefono jutiklius.
- *android.os.Bundle* – ši klasė naudojama programuojant programos pradinį „*onCreate*“ sukūrimo metodą.
- *android.os.Handler* – ši klasė naudojama programuojant papildomus programos procesus.
- *android.text.Html* – ši klasė naudojama programuojant teksto stilių.
- *android.view.MotionEvent* – ši klasė yra atsakinga už išmaniojo telefono ekrano prilietimo įvykių fiksavimą.
- *android.view.View* – ši klasė yra susijusi su matomų ekrano komponentų programavimu. Pavyzdžiui mygtukų programavimu.
- *android.widget.TextView* – ši klasė susijusi su teksto atvaizdavimu išmaniojo telefono ekrane.

Kiekviena *Android* programa turi pagrindinį „*onCreate*“ metodą. Šio metodo pagalba yra inicializuojami visi svarbūs programos elementai, kurie bus vėliau naudojami programoje. Žemiau pateiktas šio metodo kodo fragmentas:

```
1. @Override
2. public void onCreate(Bundle savedInstanceState) {
3.       super.onCreate(savedInstanceState);
4.       setContentView(R.layout.activity_main);
5.       handler = new Handler();
6.       tV_status = (TextView) findViewById(R.id.tV_status);
7.       tV_accel_x = (TextView) findViewById(R.id.tV_accel_x);
8.       tV_accel_y = (TextView) findViewById(R.id.tV_accel_y);
9.       tV_lin_accel_x = (TextView) findViewById(R.id.tV_lin_accel_x);
10.      tV_lin_accel_y = (TextView) findViewById(R.id.tV_lin_accel_y);
11.      tV_giro_x = (TextView) findViewById(R.id.tV_giro_x);
12.      tV_giro_y = (TextView) findViewById(R.id.tV_giro_y);
```

```
13.      textView_BT_input = (TextView) findViewById(R.id.textView_BT_input);
14.      textView_vnt1 = (TextView) findViewById(R.id.textView_vnt1);
15.      textView_vnt2 = (TextView) findViewById(R.id.textView_vnt2);
16.      textView_vnt3 = (TextView) findViewById(R.id.textView_vnt3);
17.      textView_vnt4 = (TextView) findViewById(R.id.textView_vnt4);
18.      textView_vnt5 = (TextView) findViewById(R.id.textView_vnt5);
19.      textView_vnt6 = (TextView) findViewById(R.id.textView_vnt6);
20.      bluetoothadapter = BluetoothAdapter.getDefaultAdapter();
21.      sensor_manager = (SensorManager) getSystemService(SENSOR_SERVICE);
22.      gyro = sensor_manager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
23.      accelerometer = sensor_manager
24.           .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
25.      linear_accelerometer = sensor_manager
26.           .getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
27. }
```

Antroje eilutėje metodo argumente yra panaudotas žodis „*savedInstanceState*" – tai reškia, kad sukurta programa veikimo metu gali būti sustabdyta ir jos dabartinė būsena bus išsaugota. Pavyzdžiui jeigu programos langas bus nuleistas žemyn ir bus iškviesta kita programa. Grįžus atgal į programos langą, programos arbas bus ten, kur buvo sustabdytas. Ketvirtoje eilutėje yra inicijuojami programos išvaizdos išdėstymo schema. Šiuo atveju yra kreipiamasi į activity_main.xml failą. Penktoje eilutėje sukuriamas kintamasis dėl antro programos proceso sukūrimo. Nuo šeštos iki devynioliktos eilutės yra inicijuojami programos išvaizdos komponentai. Dvidešimtoje eilutėje inicijuojamas kintamasis susijęs su *Bluetooth* įrenginio valdymu. Visos likusios kodo eilutės susijusios su išmaniojo telefono jutiklių iniciavimu.

Žemiau nagrinėjami metodai vykdomi sukurtos programos veikimo metu. Pirmiausia operatoriui paspaudus mygtuką prisijungti prie roboto yra vykdomi sekantys metodai:

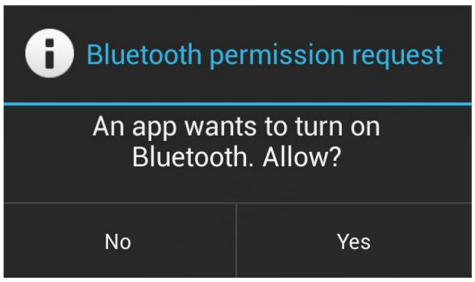
```
1. public void PrisijungtiPrieRoboto(View view) {
2.       if (bluetooth_open_on_off == false) {
3.            enable_bluetooth();
4.            return;
5.       }
6.       if (bluetooth_open_on_off == true) {
7.            try {
8.            close_bluetooth();
9.            } catch (IOException e) {
10.                e.printStackTrace();
11.           }
12.      }
13. }
```

Pirmiausia metodas tikrina ar yra įjungtas *Bluetooth* bevielis ryšys. Jeigu *Bluetooth* neįjungtas tada iškviečiamas *Bluetooth* įjungimo metodas (3 eilutė), jeigu *Bluetooth* ryšys įjungtas iškviečiamas *Bluetooth* išjungimo metodas (8 eilutė). Žemiau pateiktas *Bluetooth* įjungimo metodas:

```
1. private void enable_bluetooth() {
2.       if (!bluetoothadapter.isEnabled()) {
3.            Intent bluetoothIntent = new Intent(
4.            BluetoothAdapter.ACTION_REQUEST_ENABLE);
```

```
5.            startActivityForResult(bluetoothIntent, ENABLE_BLUETOOTH);
6.       } else {
7.            tV_status.setText("Bluetooth įjungtas");
8.            find_bluetooth();
9.       }
10.}
```

Pirmoji kodo sąlyga patikrina ar išmaniojo telefono *Bluetooth* įrenginys yra įjungtas. Jeigu *Bluetooth* yra išjungtas, tada iškviečiama *Bluetooth* įjungimo paprogramė ir telefono ekrane pasirodo pranešimas dėl *Bluetooth* įrenginio įjungimo. Pasirodžius pranešimui, operatorius gali įjungti arba ne *Bluetooth* įrenginį (30 pav.).



30 pav. Bluetooth įjungimo pranešimas išmaniojo telefono ekrane

Jeigu *Bluetooth* įrenginys jau buvo įjungtas sukurta programa pradeda roboto manipuliatoriaus *Bluetooth* įrenginio paiešką:

```
1. private void find_bluetooth() {
2.       Set<BluetoothDevice> pairedDevice = bluetoothadapter.getBondedDevices();
3.       if (pairedDevice.size() > 0) {
4.            for (BluetoothDevice device : pairedDevice) {
5.                 if (device.getName().equals("Serial Adaptor")) {
6.                      bluetoothdevice = device;
7.                      tV_status.setText("Bluetooth rastas");
8.                      break;
9.                 } else
10.                     tV_status.setText("Bluetooth nerastas");
11.           }
12.           if (bluetoothdevice.getName().equals("Serial Adaptor")) {
13.               try {
14.               connect_bluetooth();
15.               } catch (IOException ex) {
16.               }
17.           }
18.      }
19.}
```

Kodo eilutės nuo antros iki vienuoliktos yra atsakingos už *Bluetooth* įrenginio paiešką. *Bluetooth* įrenginys yra ieškomas išmaniojo telefono atmintyje, kadangi išmanusis telefonas ir robotas manipuliatorius jau buvo vieną kartą suporuoti pasinaudojant *Android* OS *Bluetooth* programa. Sukurta programa peržiūrėjusi visą atmintyje likusių *Bluetooth* įrenginių sąrašą ir radusi įrenginį vardu „*Serial Adaptor*“ (taip vadinasi roboto manipuliatoriaus *Bluetooth* įrenginys) pradeda vykdyti prisijungimą prie rasto *Bluetooth* įrenginio. Šį veiksmą aprašo kodo dalis nuo dvyliktos iki

septynioliktos eilutės. Keturioliktoje eilutėje yra iškviečiamas prisijungimo metodas. Žemiau pateiktas šio metodo programinis kodas:

```
1. private void connect_bluetooth() throws IOException {
2.       tV_status.setText("Jungiamasi...");
3.       UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
4.       bluetoothsocket = bluetoothdevice
5.            .createRfcommSocketToServiceRecord(uuid);
6.       bluetoothsocket.connect();
7.       outputstream = bluetoothsocket.getOutputStream();
8.       inputstream = bluetoothsocket.getInputStream();
9.       tV_status.setText("Prisijungta sėkmingai");
10.      bluetooth_open_on_off = true;
11.      bluetooth_input_listening();
12.      startThread();
13. }
```

Trečioje kodo eilutėje yra nurodomas nekintamo unikalaus universalaus identifikatoriaus kodas. Šis kodas parinktas specialiai naudojamiems *Bluetooth* įrenginiams. Ketvirtoje ir penktoje kodo eilutėse yra vykdomas prisijungimas prie roboto *Bluetooth* įrenginio. Septinta ir aštunta eilutės sukuria duomenų įvesties ir išvesties srautus. Vienuolikta eilutė iškviečia metodą, kuris parengia duomenis antro programos proceso sukūrimui ir galiausiai dvylikta eilutė paleidžia antrą programos procesą atsakingą už duomenų priėmimą iš *Bluetooth* įrenginio. Šioje vietoje operatorius jau yra sėkmingai prisijungęs prie roboto manipuliatoriaus.

Sėkmingai prisijungus ir operatoriui paspaudus roboto valdymo įjungimo mygtuką yra vykdomas šis metodas:

```
1. public void IjungtiRobota(View view) {
2.       if (bluetooth_open_on_off == true) {
3.            if (robot_control_on_off == false) {
4.                 robot_control_on_off = true;
5.                 tV_status.setText("Roboto valdymas įjungtas");
6.                 if (platforma == true) {
7.                      tV_status.setText("Įjungtas platformos valdymas");
8.                 }
9.                if (kranas == true) {
10.                    tV_status.setText("Įjungtas krano valdymas");
11.               }
12.               textView_BT_input.setText("robotas sustabdytas");
13.               return;
14.          }
15.          if (robot_control_on_off == true) {
16.               robot_control_on_off = false;
17.               tV_status.setText("Roboto valdymas išjungtas");
18.               if (message_command != "s") {
19.                    message_command = "s";
20.                    try {
21.                         send_command_bluetooth();
22.                    } catch (IOException ex) {
23.                    }
24.                } else {
25.                    textView_BT_input.setText("robotas sustabdytas");
26.                }
27. }}}
```

Antroje kodo eilutėje patikrinama ar sukurta specializuota išmaniojo telefono programinė įranga yra prisijungusi prie roboto manipuliatoriaus. Tam naudojamas loginio tipo kintamasis. Jeigu programa prisijungusi prie roboto manipuliatoriaus, tada tikrinama ar roboto valdymas yra įjungtas ar ne. Tam vėl gi naudojamas loginio tipo kintamasis. Jeigu valdymas išjungtas loginio tipo kintamajam „*robot_control_on_off*" priskiriama true reikšmė bei atvaizduojamas tam tikras tekstas išmaniojo telefono ekrane. Dėka loginio tipo kintamojo tolimesnis programos kodas žinos, kada operatorius yra įjungęs roboto manipuliatoriaus valdymą. Jeigu roboto manipuliatoriaus valdymas buvo jau įjungtas, yra vykdomos kodo eilutės nuo penkioliktos iki dvidešimt septintos. Pirmiausia loginiam kintamajam yra priskiriama false reikšmė. Tai pažymi, kad valdymas yra išjungtas. Po to yra išsiunčiama roboto manipuliatoriaus sustabdymo komanda, kuri sustabdo visus roboto manipuliatoriaus veiksmus. Komandos siuntimui yra naudojamas metodas „*send_command_bluetooth()*". Visom komandom, kurios siunčiamos į robotą manipuliatorių, naudojamas „*send_command_bluetooth()*" metodas. Žemiau pateiktas šio metodo programinis kodas:

```
1. private void send_command_bluetooth() throws IOException {
2.       message = message_command;
3.       outputstream.write(message.getBytes());
4. }
```

Šiame metode svarbiausia kodo eilutė yra trečioji, nes ji išsiunčia valdymo komandą į robotą manipuliatorių. Įjungus roboto valdymą operatorius gali pasirinkti kaip valdys robotą: ar valdys tik gestais, ar valdys tik platformą ar tik kraną. Sukurtoje programoje kiekvienas valdymo tipas įjungiamas atskiru metodu. Žemiau nagrinėjamas valdymo tik gestais įjungimo, išjungimo metodas:

```
1. public void ValdymasTikGestais(View view) {
2.       if (robot_control_on_off == true && bluetooth_open_on_off == true) {
3.           if (valdymas_gestais == false) {
4.                valdymas_gestais = true;
5.                tV_status.setText("Įjungtas valdymas tik gestais");
6.                if (message_command != "s") {
7.                     message_command = "s";
8.                     try {
9.                          send_command_bluetooth();
10.                    } catch (IOException ex) {
11.                    }
12.               } else {
13.                    textView_BT_input.setText("robotas sustabdytas");
14.               }
15.               return;
16.          }
17.          if (valdymas_gestais == true) {
18.               valdymas_gestais = false;
19.               tV_status.setText("Išjungtas valdymas tik gestais");
20.               if (message_command != "s") {
21.                    message_command = "s";
22.                    try {
23.                         send_command_bluetooth();
24.                    } catch (IOException ex) {
```

```
                    }
                } else {
                    textView_BT_input.setText("robotas sustabdytas");
                }
            }
        }
}
```

Šio metodo paskirtis įjungti arba išjungti tik gestų valdymą. Tai padaroma atitinkamai priskiriant loginiam kintamajam true arba false reikšmes. Ir tiek valdymo įjungimo, tiek išjungimo atveju robotui yra išsiunčiama sustabdymo komanda bei atvaizduojama atitinkama informacija išmaniojo telefono ekrane.

Vien tik roboto manipuliatoriaus platformos ar krano įjungimo metodai labai panašūs, kaip ką tik aptartas metodas. Metodo vykdymo metu vėl gi yra naudojami loginio tipo kintamieji, kuriems priskiriamos true arba false reikšmės. Pagal tai programa žino, koks valdymas yra įjungtas. Be to kiekvienas taip pat išsiunčia ir roboto manipuliatoriaus sustabdymo komandą.

Bet kurio valdymo režimo metu yra vykdomas gestų atpažinimo metodas. Visas metodas yra labai ilgas, todėl bus paminėtos tik svarbiausios metodo dalys. Visą programos kodą galima rasti 5 priede. Gestų atpažinimo metodas programoje vadinasi „*onSensorChanged(SensorEvent event)*“. Metodo veikimo pradžioje yra nuskaitoma informacija iš akselerometrinio ir giroskopinio jutiklių. Metodas tris jutiklių reikšmes. Pirmiausia iš akselerometro yra gaunami duomenys apie statinę įrenginio akceleraciją. Dėka šių duomenų galima nustatyti, kokiu kampu yra pasviręs išmanusis telefonas žemės atžvilgiu. Taip pat iš akselerometro yra gaunami duomenys ir apie dinaminę įrenginio akseleraciją (linijinį pagreitį). Žinant šiuos duomenis, galima nustatyti kokia kryptimi juda išmanusis telefonas. Galiausiai yra nuskaitomi giroskopinio jutiklio duomenys. Giroskopinis jutiklis suteikia informacijos apie kampinį pagreitį. Duomenys apie jutiklių informaciją yra pateikiami išmaniojo telefono ekrane. Žemiau pateiktas kodo fragmentas atsakingas už jutiklių duomenų nuskaitymą:

```
sensor = event.sensor;
if (sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
    linijinis_akselerometras_x = event.values[0];
    linijinis_akselerometras_y = event.values[1];
    tV_lin_accel_x.setText("Linijinis pagreitis x: "
    + linijinis_akselerometras_x);
    textView_vnt3.setText(Html.fromHtml("m/s<sup>2</sup>"));
    tV_lin_accel_y.setText("Linijinis pagreitis y: "
    + linijinis_akselerometras_y);
    textView_vnt4.setText(Html.fromHtml("m/s<sup>2</sup>"));
}
if (sensor.getType() == Sensor.TYPE_GYROSCOPE) {
    giroskopas_x = event.values[0];
    giroskopas_y = event.values[1];
    tV_giro_x.setText("Kampinis pagreitis x: " + giroskopas_x);
    textView_vnt5.setText("rad/s");
    tV_giro_y.setText("Kampinis pagreitis y: " + giroskopas_y);
    textView_vnt6.setText("rad/s");
```

```
19. }
20. if (sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
21.      akselerometras_x = event.values[0];
22.      akselerometras_y = event.values[1];
23.      tV_accel_x.setText("Akselerometras x: " + akselerometras_x);
24.      textView_vnt1.setText(Html.fromHtml("m/s<sup>2</sup>"));
25.      tV_accel_y.setText("Akselerometras y: " + akselerometras_y);
26.      textView_vnt2.setText(Html.fromHtml("m/s<sup>2</sup>"));
27. }
```

Kodo eilutės nuo antros iki vienuoliktos yra susijusios su akselerometrinio jutiklio duomenų nuskaitymu. Šiuo atveju nuskaitoma dinaminė akceleracija (linijinis pagreitis). Kodo eilutės nuo dvyliktos iki devynioliktos susijusios su giroskopinio jutiklio duomenų nuskaitymu. Likusios kodo eilutės atsakingos už akselerometrinio jutiklio duomenų nuskaitymą. Šiuo atveju nuskaitoma statinė akseleracija.

Šie trijų rūšių duomenys iš jutiklių naudojami operatoriaus rankos gestų atpažinimui. Gestams atpažinti yra naudojama visų trijų rūšių duomenų informacija. Priklausomai nuo gesto rūšies naudojamai vieni ar kiti jutiklių duomenys ar duomenų kombinacija. Kiekvienas gestas atitinka skirtingas jutiklių duomenų reikšmes arba tam tikrą atliekamą veiksmų seką. Žemiau pateiktas vieno gesto atpažinimo kodo fragmentą:

```
1.  if (galima_vykdyti_komanda == true
2.  && akselerometras_y > 4.0 && giroskopas_x > 1.5) {
3.       galima_vykdyti_komanda = false;
4.       if (message_command != "j" && ekranas_priliestas == true) {
5.            message_command = "j";
6.            try {
7.                 send_command_bluetooth();
8.            } catch (IOException ex) {
9.            }
10.      }
11. return;
12. }
```

Šį kodo fragmentą reikėtų interpretuoti taip: operatorius laikydamas išmanųjį telefoną pasuka riešą aukštyn, link operatoriaus (48 pav.). Šiuo atveju gestui atpažinti panaudojami akselerometrinio ir giroskopinio jutiklių duomenys. Operatoriui atlikus šį gestą, kada įjungtas valdymo tik gestais režimas, yra valdomas vienas iš roboto manipuliatoriaus krano variklių. Iš viso sukurta programa gali atpažinti vienuolika skirtingų operatoriaus rankos gestų.

Kitų gestų atpažinimo programinis kodas yra panašus, kaip ir prieš tai aptartas kodo fragmentas. Tačiau skiriasi kiekvieno gesto veiksmų seką, kurią operatorius turi atlikti, tad kiekvienu atvejų yra reikalingos skirtingos jutiklių duomenų reikšmės. Visą sukurtą programinį kodą galima rasti 5 priede.

Be gestų mišriame valdymo režime taip pat naudojami ir keli mygtukai robotui manipuliatoriui valdyti. Atitinkamai paspaudus vieną ar kitą valdymo mygtuką yra išsiunčiama atitinkama valdymo komanda robotui manipuliatoriui.

Tiek viename, tiek kitame valdymo režime operatorius bet kada gali įjungti arba išjungti magnetą. Už magento valdymą yra atsakinagas *MagnetoValdymas(View view)* metodas. Metodo vykdymo metu robotui manipuliatoriui yra išsiunčiama atitinkamai magneto įjungimo arba išjungimo komanda ir taip pat loginio tipo kintamajam priskiriama true arba false reikšmė. Loginio tipo kintamasis naudojamas tam, kad likusi programa žinotų ar magnetas yra įjungtas ar ne.

Be jau aptartų kodo dalių labai svarbu paminėti ekrano prilietimo atpažinimo metodą „*onTouchEvent(MotionEvent event)*“. Šis metodas atsakingas už tai, kad nei viena komanda nebūtų vykdoma jeigu ekranas nėra priliestas pirštu. Šis metodas atsakingas ir už tai, kad robotas manipuliatorius būtų iškart sustabdytas jeigu operatorius patraukia pirštą nuo ekrano. Tai yra būtinas saugumo reikalavimas.

Žemiau nagrinėjama paskutinė esminė kodo dalis, kuri yra atsakinga už informacijos iš roboto manipuliatoriaus priėmimą. Už šią kodo dalį yra atsakingi šie metodai: *bluetooth_input_listening()*, *startThread()* ir *VykdomosKomandosNustatymas(String vykdoma_komanda)*. Pirmieji du metodai yra skirti paleisti ir vykdyti antrą programos procesą atsakingą už informacijos priėmimą iš roboto manipuliatoriaus. Programinis kodas esantis šiuose metoduose nuolatos tikrina ar nėra gauta informacija per *Bluetooth* iš roboto manipuliatoriaus. Jeigu gauta informacija buvo aptikta, tada ši informacija yra perduodama trečiajam metodui *VykdomosKomandosNustatymas(String vykdoma_komanda)*, kuris iškoduoja gautą informaciją ir apie tai praneša išmaniojo telefono ekrane. Dėka šių trijų metodų operatorius žino kokia valdymo komandą šiuo metu vykdo robotas manipuliatorius.

Išnagrinėtos pačias svarbiausios *MainActivity.java* failo dalys, kurios suteikia visą sukurtos specializuotos programos vykdymo funkcionalumą. Visas pagrindinio failo *MainActivity.java* programinis kodas yra pateiktas 5 priede.

Sukurtos specializuotos išmaniojo telefono programos pagrindinių nustatymų AndroidManifest.xml failas:

Šiame faile yra pateikiama informacija apie pagrindinius sukurtos programos nustatymus. Šiame faile nurodoma, kokia *Android* OS versija galės naudotis sukurta programa. Šiame faile nurodyti visi leidimai reikalingi prieiti prie sistemos resursų. Šiuo atveju sukurtai programai yra suteiktas leidimas naudoti ir visiškai valdyti išmaniojo telefono *Bluetooth* modulį. Taip pat yra pateikiama informacija apie programos langus. Koks programos langas turi atsidaryti paleidus sukurtą programą. Kadangi programoje naudojamas tik vienas langas, jis ir nurodomas kaip pagrindinis programinis langas. Be šių svarbių dalykų dar pateikiama informacija apie sukurtos

specializuotos programos versiją. Versijos numeris yra labai svarbus, jeigu programą reikėtų platinti viešai. Taip pat šiame faile nurodomas sukurtos programos paketo pavadinimas. Visas *AndroidManifest.xml* failo kodas yra pateiktas 6 priede.

3.2.3 Sukurtos išmaniojo telefono programos atpažįstami rankos gestai

Specializuota išmaniojo telefono programa sukurta taip, kad atpažintų vienuolika skirtingų operatoriaus rankos gestų. Valdant robotą manipuliatorių naudojant tik gestų valdymą, roboto manipuliatoriaus valdymui yra naudojami visi vienuolika gestų. Valdant robotą manipuliatorių naudojant mišrų valdymo režimą su valdymo mygtukais, roboto manipuliatoriaus valdymui yra naudojami tik penki iš vienuolikos gestų. Bandymų metu operatoriui atlikus bet kurį iš vienuolikos gestų, sukurta specializuota išmaniojo telefono programinė įrangą sėkmingai išsiųsdavo valdymo komandą į robotą manipuliatorių. Žemiau aprašytas kiekvienas gestas, gesto atliko technika bei su konkrečiu gestu susieta roboto manipuliatoriaus valdymo komanda.

1. Išmanusis telefonas laikomas horizontalioje padėtyje (31 pav.). Atlikus šį gestą, robotui manipuliatoriui išsiunčiama sustabdymo komanda. Ši komanda naudojama norint visiškai sustabdyti roboto manipuliatoriaus judėjimą. Šis gestas naudojamas valdant robotą abiejuose valdymo režimuose.

31 pav. Roboto manipuliatoriaus judėjimo sustabdymo gestas

2. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus) ir delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę (32 ir 33 pav.). Šis gestas naudojamas tik gestų valdymo režime. Atlikus šį gestą robotui manipuliatoriui yra išsiunčiama komanda, kad roboto manipuliatoriaus platforma bėgiais važiuotų į kairę pusę.

32 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu stumiamas į priekį (tolyn nuo operatoriaus)

33 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę

3. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus) ir delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę (34 ir 35 pav.). Šis gestas naudojamas tik gestų valdymo režime. Atlikus šį gestą robotui manipuliatoriui yra išsiunčiama komanda, kad roboto manipuliatoriaus platforma bėgiais važiuotų į dešinę pusę.

34 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu stumiamas į priekį (tolyn nuo operatoriaus)

35 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę

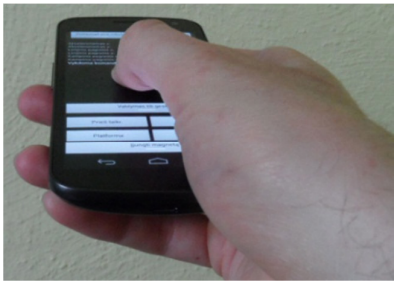
4. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus) ir delne sukamuoju judesiu pasukamas žemyn (išmaniojo telefono ekranas juda tolyn nuo operatoriaus) (36 ir 37 pav.). Šis gestas naudojamas tik gestų valdymo režime. Atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – nuleisti traukinių vagonų pozicionavimo svirtį.

36 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus)

37 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas žemyn

5. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus) ir delne sukamuoju judesiu pasukamas aukštyn (išmaniojo telefono ekranas juda artyn link operatoriaus) (38 ir 39 pav.). Šis gestas naudojamas tik gestų valdymo režime. Atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – pakelti traukinių vagonų pozicionavimo svirtį.

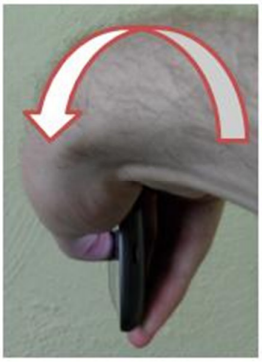
38 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judesiu pastumiamas į priekį (tolyn nuo operatoriaus)



39 pav. Išmanusis telefonas sukamuoju judesiu pasukamas aukštyn

6. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judėjimu pastumiamas į dešinę pusę ir delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę (40 ir 41 pav.). Šis gestas naudojamas tik gestų valdymo režime. Atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – roboto manipuliatoriaus kranui suktis prieš laikrodžio rodyklę.

40 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judėjimu pastumiamas į dešinę pusę



41 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę

7. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judėjimu pastumiamas į kairę pusę ir delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę (42 ir 43 pav.). Šis gestas naudojamas tik gestų valdymo režime. Atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – roboto manipuliatoriaus kranui suktis pagal laikrodžio rodyklę.

42 pav. Išmanusis telefonas horizontalioje padėtyje slenkamuoju judėjimu pastumiamas į kairę pusę

43 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę

8. Išmanusis telefonas laikomas horizontalioje padėtyje ir delne sukamuoju judesiu pasukamas į kairę pusę prieš laikrodžio rodyklę (44 pav.). Šis gestas naudojamas abiejuose valdymo režimuose. Tik gestų valdymo režime atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – leisti pirmąją krano ašį žemyn (24 pav. 2 nr.). Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus platformos valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda – roboto manipuliatoriaus platformai bėgiais važiuoti į kairę pusę. Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus krano valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda - leisti pirmąją krano ašį žemyn (24 pav. 2 nr.).

44 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas prieš laikrodžio rodyklę

9. Išmanusis telefonas laikomas horizontalioje padėtyje ir delne sukamuoju judesiu pasukamas į dešinę pusę pagal laikrodžio rodyklę (45 pav.). Šis gestas naudojamas abiejuose valdymo režimuose. Tik gestų valdymo režime atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – kelti pirmąją krano ašį aukštyn (24 pav. 2 nr.). Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus platformos valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda – roboto manipuliatoriaus platformai bėgiais važiuoti į dešinę pusę. Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus krano valdymas, atlikus

gestą, robotui manipuliatoriui išsiunčiama komanda - kelti pirmąją krano ašį aukštyn (24 pav. 2 nr.).

45 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas pagal laikrodžio rodyklę

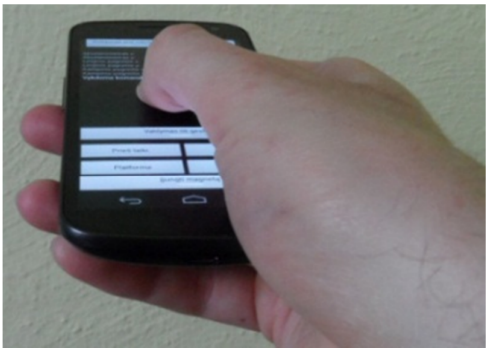
10. Išmanusis telefonas laikomas horizontalioje padėtyje ir delne sukamuoju judesiu pasukamas žemyn (išmaniojo telefono ekranas juda tolyn nuo operatoriaus) (46 ir 47 pav.). Šis gestas naudojamas abiejuose valdymo režimuose. Tik gestų valdymo režime atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – leisti antrąją krano ašį žemyn (24 pav. 3 nr.). Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus platformos valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda – nuleisti traukinių vagonų pozicionavimo svirtį. Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus krano valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda - leisti antrąją krano ašį žemyn (24 pav. 3 nr.).

46 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas žemyn

47 pav. Išmanusis telefonas delne sukamuoju judesiu pasukamas žemyn

11. Išmanusis telefonas laikomas horizontalioje padėtyje ir delne sukamuoju judesiu pasukamas aukštyn (išmaniojo telefono ekranas juda artyn link operatoriaus) (48 ir 49 pav.). Šis gestas naudojamas abiejuose valdymo režimuose. Tik gestų valdymo režime atlikus šį gestą robotui manipuliatoriui išsiunčiama komanda – kelti antrąją krano ašį aukštyn (24 pav. 3 nr.). Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus platformos valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda – pakelti traukinių vagonų pozicionavimo svirtį. Mišriame valdymo režime, kada įjungtas roboto manipuliatoriaus krano valdymas, atlikus gestą, robotui manipuliatoriui išsiunčiama komanda - kelti antrąją krano ašį aukštyn (24 pav. 3 nr.).

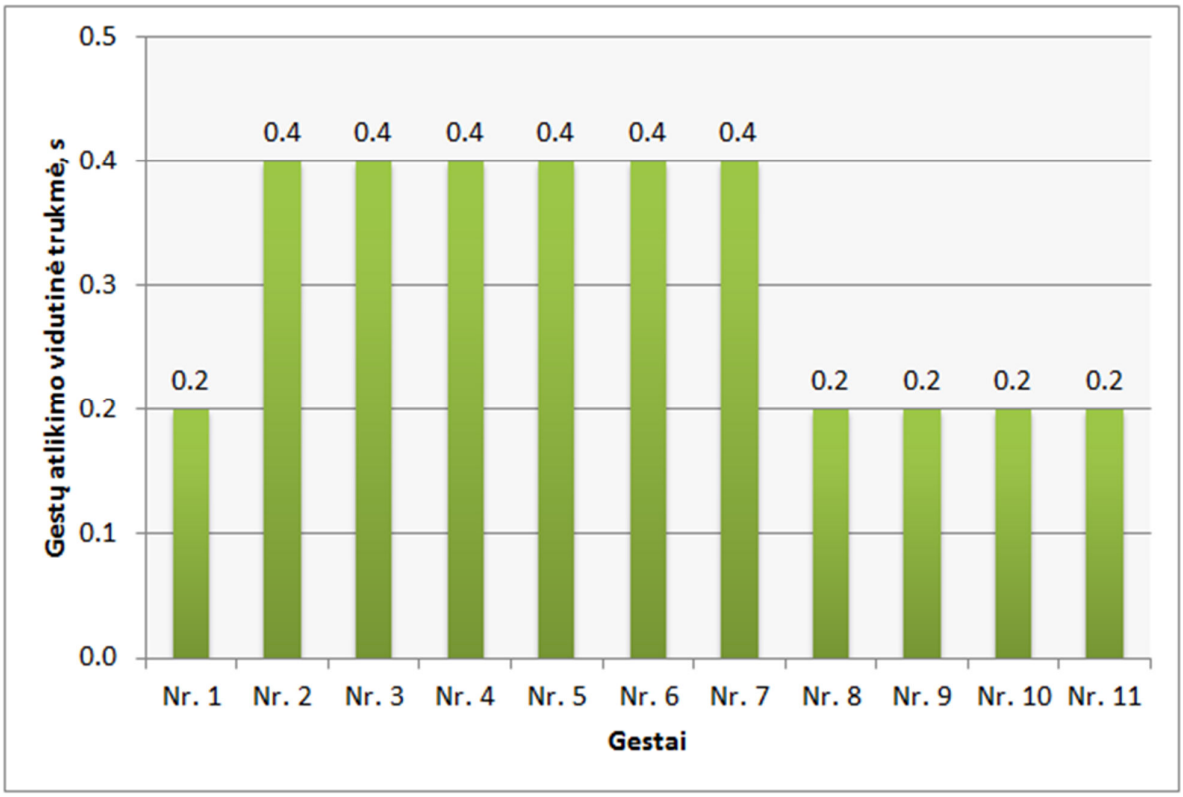
48 pav. Išmanusis telefonas sukamuoju judesiu pasukamas aukštyn



49 pav. Išmanusis telefonas sukamuoju judesiu pasukamas aukštyn

Vidutinė gestų atlikimo trukmė:

Žemiau pateikta diagrama, kurioje pateikti duomenys apie vidutinę kiekvieno gesto atlikimo trukmę sekundėmis (50 pav.). Su kiekvienu gestu buvo atlikta po 100 bandymų ir nustatyta kiekvieno gesto vidutinė atlikimo trukmė. Gauti duomenys (50 pav.) parodė, kad gestai, kuriems atlikti tereikia vieno judesio rankos riešu, atliekami apytiksliai du kartus greičiau, negu gestai susidedantys iš dviejų rankos judesių. Pagrindiniai penki vieno judesio rankos riešu gestai (Nr. 1, Nr. 8, Nr. 9, Nr. 10. Nr. 11) naudojami abiejuose valdymo režimuose. O likę gestai, kuriems atlikti reikia dviejų rankos judesių naudojami tik gestų valdymo režime.



50 pav. Gestų atlikimo vidutinė trukmė, s

IV. VERIFIKAVIMO DALIS

4.1. Sukurtos nuotolinio valdymo sistemos reikalavimų atitikimas

Roboto manipuliatoriaus nuotolinio valdymo sistema buvo sukurta laikanatis antrame skyriuje aprašytų sistemos funkcinių, nefunkcinių reikalavimų bei apribojimų. Robotui manipuliatoriui valdyti buvo panaudotas išmanusis telefonas su Android 4.2.1 operacine sistema. Išmaniajame telefone įmontuotas 1,2 GHz dviejų branduolių procesorius, akselerometrinis bei giroskopinis jutikliai, *Bluetooth* modulis. Išmaniajame telefone įdiegta 1 GB operatyviosios atminties. Išmaniojo telefono ekrano įstrižainės skersmuo yra 4,65 coliai.

Roboto manipuliatoriaus valdymui naudojama Arduino Mega 2560 plokštė, kurioje yra įmontuotas atmega 2560 mikrovaldiklis. Prie mikrovaldiklio taip pat yra prijungtas *Bluetooth* modulis, kurio duomenų perdavimo sparta yra 19200 bit/s.

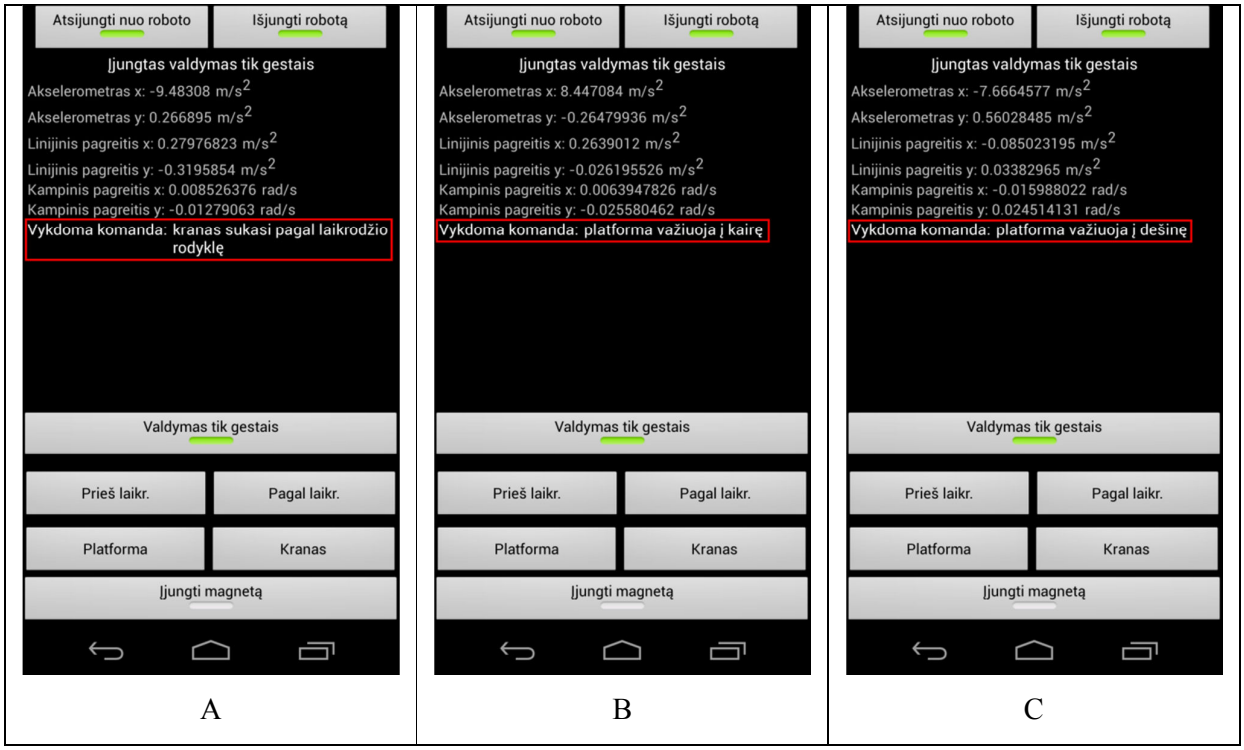
Buvo sėkmingai įgyvendini sistemos reikalavimuose apsibrėžti tikslai:

a. Sukurtas roboto manipuliatoriaus prototipas bandymų metu atpažino gautas valdymo komandas iš sukurtos specializuotos išmaniojo telefono programinės įrangos ir įvykdė operatoriaus užduotus veiksmus. Be įvykdytų komandų, sukurta roboto manipuliatoriaus prototipo programinė įranga taip pat išsiųsdavo informaciją, apie šiuo metu roboto manipuliatoriaus vykdomą komandą, sukurtai išmaniojo telefono programinei įrangai.
b. Specializuota išmaniojo telefono programinė įranga sukurta laikantis visų apsibrėžtų reikalavimų. Specializuota išmaniojo telefono programinė įranga sukurta taip, kad galėtų atpažinti 11 operatoriaus rankos gestų t.y. 11 skirtingų komandų, kurias gali užduoti operatorius. Sukurtoje programoje buvo suprogramuoti ir valdymo mygtukai, kurių pagalba taip pat valdomas robotas manipuliatorius.
c. Bandymų metu informacijos perdavimui tarp išmaniojo telefono ir roboto manipuliatoriaus prototipo buvo sėkmingai naudojamas bluetooth bevielis ryšis.
d. Sukurta specializuota išmaniojo telefono programinė įranga bandymų metu atvaizdavo visą reikiamą informaciją susijusę su roboto manipuliatoriaus valdymu.

4.2. Roboto manipuliatoriaus vykdomų komandų patvirtinimas

Specializuota išmaniojo telefono programinė įranga sukurta taip, kad galėtų priimti informaciją iš roboto manipuliatoriaus prototipo. Ši funkcija reikalinga tam, kad operatorius tiksliai žinotų, kokią komandą vykdo robototas manipuliatorius. Ši informacija yra pateikiama išmaniojo telefono ekrane. Sukurta specializuota roboto manipuliatoriaus programinė įranga gali siųsti trylika informacinių pranešimų apie vykdomas komandas sukurtai specializuotai išmaniojo telefono programinei įrangai. Bandymų metu buvo išbandytos visos įmanomos roboto manipuliatoriaus

valdymo komandos, kurias robotas manipuliatorius įvykdė ir informaciją apie vykdomą komandą išsiuntė sukurtai specializuotai išmaniojo telefono programinei įrangai. Sukurta išmaniojo telefono programinė įranga atvaizdavo gautą informaciją apie vykdomas komandas išmaniojo telefono ekrane. Žemiau yra pateikti trys pavyzdžiai, kaip atrodo gauta informacija apie vykdomą roboto manipuliatoriaus komandą išmaniojo telefono ekrane (51 pav.).

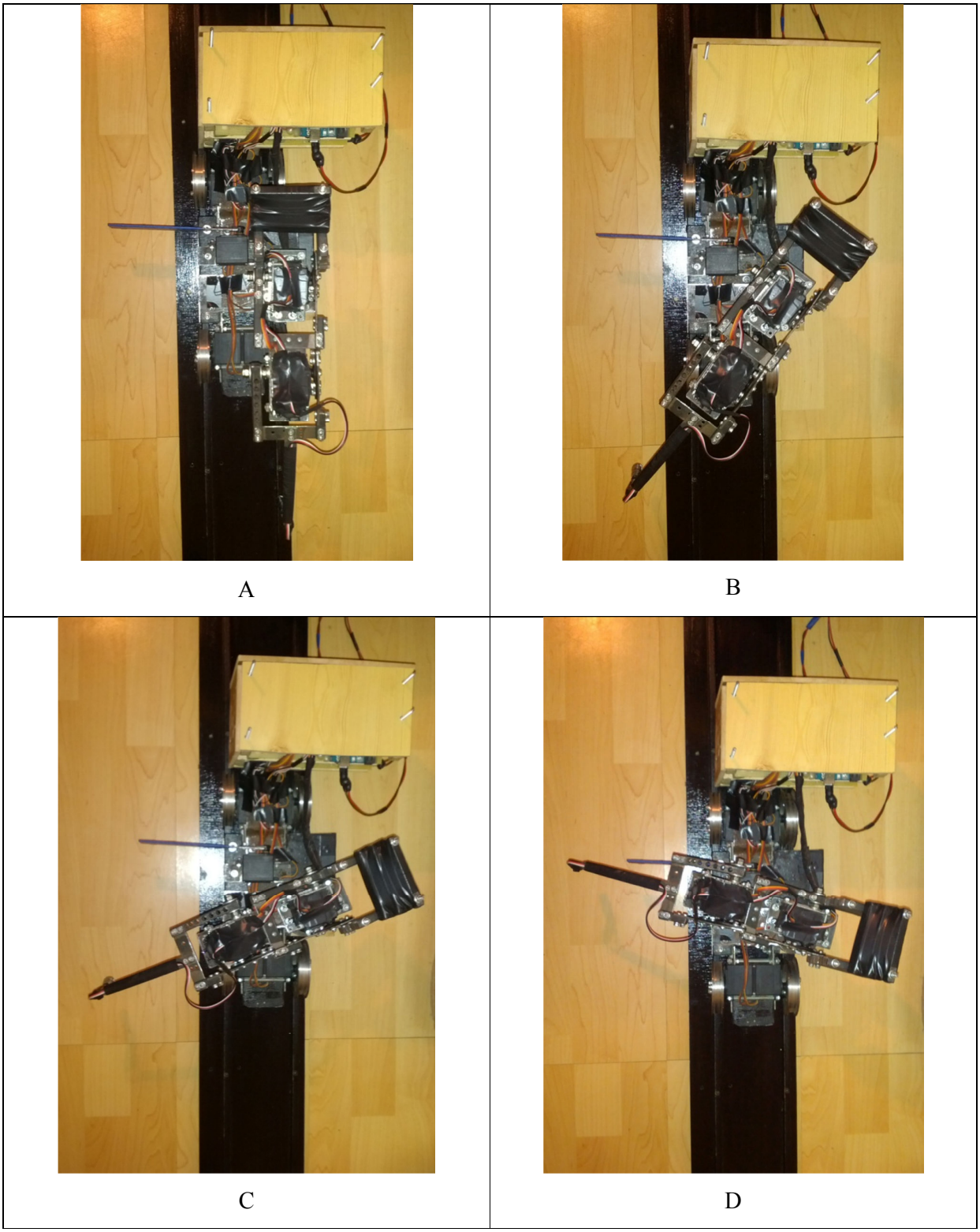


51 pav. Sukurtos specializuotos išmaniojo telefono programinės įrangos vaizdas išmaniojo telefono ekrane

Informacija apie roboto manipuliatoriaus vykdomą komandą apibraukta raudona linija (51 pav.). A dalyje (51 pav.) pranešimas teigia, kad roboto manipuliatoriaus kranas sukasi pagal laikrodžio rodyklę. B dalyje (51 pav.) pranešimas teigia, kad roboto manipuliatoriaus platforma bėgiais važiuoja į kairę pusę. C dalyje (51 pav.) pranešimas teigia, kad roboto manipuliatoriaus platforma bėgiais važiuoja į dešinę pusę.

Atitinkamai bandymų metu robotas manipuliatorius sukurtai specializuotai išmaniojo telefono programinei įrangai išsiuntė ir kitų vykdomų komandų informacinius pranešimus, kuriuos sukurta išmaniojo telefono programinė įranga atvaizdavo išmaniojo telefono ekrane.

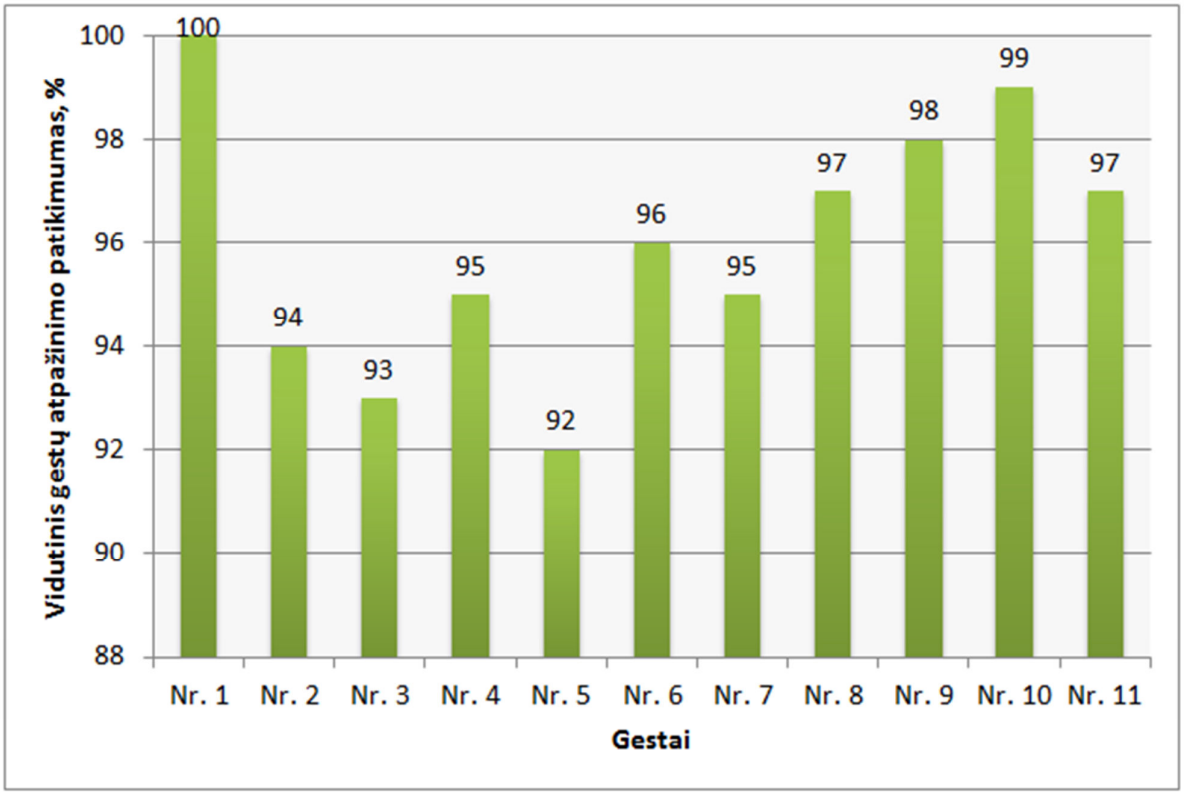
Žemiau pateikta nuotraukų seka, kaip atrodo viena iš roboto manipuliatoriaus vykdomų komandų (52 pav.). Nuotraukų sekoje pavaizduota roboto manipuliatoriaus krano sukimo pagal laikrodžio rodyklę komanda.



52 pav. Roboto manipuliatoriaus krano sukimasis pagal laikrodžio rodyklę

Pavaizduotoje nuotraukų sekoje (52 pav.) robotas manipuliatorius sėkmingai įvykdė užduotą komandą. Atitinkamai atliktų bandymų metu robotas manipuliatorius įvykdė visas operatoriaus užduotas komandas.

Iš viso su kiekvienu gestu buvo atlikta po 100 bandymų. Gauti duomenys (53 pav.) parodė, kad vidutinis gestų atpažinimo patikimumas siekia 96%. Tai reiškia, kad iš šimto vidutiniškai keturi gestai nėra atpažinami. Jeigu operatorius neteisingai atliko pasirinktą gestą, sistema tokio gesto neatpažįsta. Neatpažintus gestus sistema blokuoja, ir juos būtina pakartoti iš naujo.



53 pav. Vidutinis gestų atpažinimo patikimumas

IŠVADOS

1. Išanalizavus mokslinę literatūrą apie robotų nuotolinį valdymą nustatyta, kad robotų valdymui pasitelkiamų žmogaus rankos gestų atpažinimui naudojami tiek įvairūs prietaisai dedami ant žmogaus rankų, tiek išmanieji telefonai.
2. Sukurtas intelektinio transporto krovos manipuliatoriaus prototipas, skirtas traukinių vagonų pozicionavimui ir metalo laužo krovos darbams atlikti, su nuotoline valdymo sistema per išmanųjį telefoną su Android OS ir su įdiegtais akselerometriniu ir giroskopiniu jutikliais.
3. Sukurtoje Android OS programinės įrangos aplikacijoje buvo realizuoti du valdymo režimai: valdymas sekantis žmogaus rankos gestus ir valdymas naudojantis žmogaus rankos gestus kartu su valdymo mygtukais mobiliame įrenginyje. Sukurta nuotolinio valdymo sistema atpažįsta vienuolika skirtingų žmogaus rankos gestų. Penki pagrindiniai gestai atliekami vienu rankos judesiu, likusieji atliekami dvejais rankos judesiais. Bandymų metu nustatyta, kad vienu rankos judesiu atliekami gestai įvykdomi du kartus greičiau, negu gestai atliekami dvejais rankos judesiais.
4. Darbe parodyta, kad išmaniojo telefono panaudojimas nuotoliniam valdymui suteikia naują galimybę atsisakyti roboto manipuliatoriaus mikrovaldiklio valdymo svirčių. Sukurtą nuotolinio valdymo sistemą galima panaudoti robotų manipuliatorių, kranų nuotoliniam valdymui, bei situacijose, kada būtinas operatoriaus mobilumas ir būtinybė dėmesį sukoncentruoti į valdomą objektą.
5. Su kiekvienu gestu buvo atlikta po 100 verifikavimo testų. Gauti duomenys parodė, kad vidutinis gestų atpažinimo patikimumas siekia 96%. Tai reiškia, kad iš šimto vidutiniškai keturi gestai nėra atpažinami. Neatpažintus gestus sistema blokuoja, ir juos būtina pakartoti iš naujo. Sėkmingai atpažinusi pasirinktą gestą, sukurta Android OS programinės įrangos aplikacija išsiųsdavo valdymo komandą roboto manipuliatoriaus prototipui, kuris įvykdydavo gautą komandą.

Mokslinių tyrimų rezultatai buvo publikuoti mokslinėje konferencijoje „Technologijos mokslo darbai vakarų Lietuvoje VIII“.

LITERATŪRA

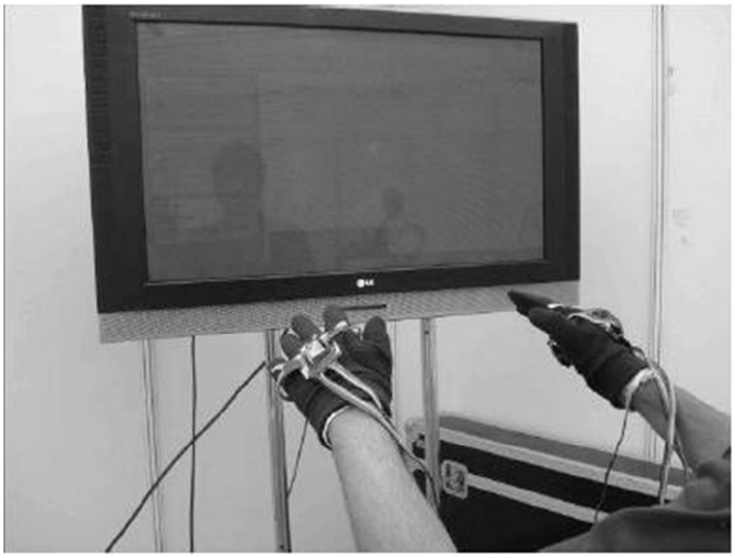
1. Abbas S. M., Hassan S., Jongwon Y. 2012. Augmented reality based teaching pendant for industrial robot. Control, Automation and Systems, 2012 12th International Conference, p. 2210-2213.
2. Android developers. 2013. Android Architecture [interaktyvus]. [žiūrėta 2013 m. kovo 10 d.]. Prieiga per Internetą: < http://developer.android.com/about/versions/index.html#>.
3. Bernold L. E. 2007. Control schemes for tele-robotic pipe installation. *Automation in Construction*, vol. 16, No. 4, p. 518-524.
4. Choi D. W., Song T. H., Jeong S. M. et al. 2011. Design the video conferencing robot for one-to-many communication using smartphone. Control, Automation and Systems, 2011 11th International Conference, p. 671-675.
5. Gulati H., Vaishya S., Veeramachaneni S. 2011. Bluetooth and Wi-Fi controlled rescue robots. India Conference, 2011 Annual IEEE, p. 1-5.
6. Hess D., Rohrig C. 2009. Remote controlling of technical systems using mobile devices. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2009 IEEE International Workshop, p. 625-628.
7. InvenSense. 2013. MPU-3050 Triple Axis Gyroscope with Embedded Digital Motion Processor™ [interaktyvus]. [žiūrėta 2013 m. kovo 12 d.]. Prieiga per Internetą: < http://www.invensense.com/mems/gyro/mpu3050.html>.
8. InvenSense. 2013. Principles of Motion Sensing [interaktyvus]. [žiūrėta 2013 m. kovo 12 d.]. Prieiga per Internetą: < http://www.invensense.com/mems/principles.html>.
9. Yoon J., Manurung A. 2010. Development of an intuitive user interface for a hydraulic backhoe. *Automation in Construction*, vol. 19, No. 6, p. 779-790.
10. Jeon Y. H., Ahn H. 2011. A multimodal ubiquitous interface system using smart phone for human-robot interaction. Ubiquitous Robots and Ambient Intelligence, 2011 8th International Conference, p. 764-767.
11. Juang S. Y., Juang J. G. 2012. Real-time indoor surveillance based on smartphone and mobile robot. Industrial Informatics, 2012 10th IEEE International Conference, p. 475-480.
12. Kang D. B., Truong L. B., Ji S. H. et al. 2011. A design of learning assistant using smartphone for R-Learning. ICT Convergence, 2011 International Conference, p. 770-771.
13. Kim D., Kim J., Lee K. et al. 2009. Excavator tele-operation system using a human arm. *Automation in Construction*, vol. 18, No. 2, p. 173-182.

14. Lambrecht J., Chemnitz M., Kruger J. 2011. Control layer for multi-vendor industrial robot interaction providing integration of supervisory process control and multifunctional control units. Technologies for Practical Robot Applications, 2011 IEEE Conference, p. 115-120.
15. Lee H. T., Tsai H. L., Chen Z. Q. et al. 2012. Mobile detecting robot with IPCam feedback. SICE Annual Conference, 2012 Proceedings, p. 1969-1973.
16. Leow T. J., Sim K. S. 2010. Teleoperated mobile manipulator robot. Cybernetic Intelligent Systems, 2010 IEEE 9th International Conference, p. 1-6.
17. MEMS & Nanotechnology Exchange. 2013. What is MEMS Technology [interaktyvus]. [žiūrėta 2013 m. kovo 12 d.]. Prieiga per Internetą: <https://www.mems-exchange.org/MEMS/what-is.html>.
18. Moon S. W., Kim Y. J., Myeong H. J. et al. 2011. Implementation of smartphone environment remote control and monitoring system for Android operating system-based robot platform. Ubiquitous Robots and Ambient Intelligence, 2011 8th International Conference, p. 211-214.
19. Murata Y., Sato N., Takayama T. et al. 2012. A gesture-based remote control for finger disabled people. Consumer Electronics, 2012 IEEE 1st Global Conference, p. 406-410.
20. Neto P., Pires J. N., Moreira A. P. 2009. Accelerometer-based control of an industrial robotic arm. Robot and Human Interactive Communication, 2009 IEEE 18th International Symposium, p. 1192-1197.
21. Oh H. K., Kim I. C. 2011. Hybrid control architecture of the robotic surveillance system using smartphones. Ubiquitous Robots and Ambient Intelligence, 8th International Conference, p. 782-785.
22. Open handset alliance. 2013. Android [interaktyvus]. [žiūrėta 2013 m. kovo 10 d.]. Prieiga per Internetą: <http://www.openhandsetalliance.com/android_overview.html>.
23. Pandit A., Dand D., Mehta S. et al. 2009. A Simple Wearable Hand Gesture Recognition Device Using iMEMS. Soft Computing and Pattern Recognition, 2009 International Conference, p. 592-597.
24. Sasaki T., Kawashima K. 2008. Remote control of backhoe at construction site with a pneumatic robot system. *Automation in Construction*, vol. 17, No. 8, p. 907-914.
25. Serafimov K., Angelkov D., Koceska N. et al. 2012. Using mobile-phone accelerometer for gestural control of soccer robots. Embedded Computing, 2012 Mediterranean Conference, p. 140-143.
26. Song M., Kim B., Ryu K. et al. 2010. A design of real time robot control system based on android smartphone. The 7th International conference on ubiquitous robots and ambient intelligence, p. 1-3.

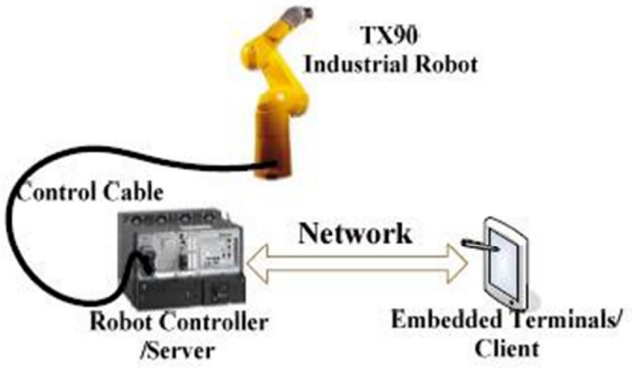
27. Tsuchiya R., Shimazaki S., Sakai T. et al. 2012. Simulation environment based on smartphones for Cloud computing robots. Telecommunications and Signal Processing, 2012 35th International Conference, p. 96-100.
28. University of Surrey. 2013. World's first "phonesat" Smartphone successfully launched into orbit [interaktyvus]. [žiūrėta 2013 m. kovo 2 d.]. Prieiga per Internetą: <http://www.surrey.ac.uk/mediacentre/press/2013/98519_worlds_first_phonesat_smartphone_strand1_successfully_launched_into_orbit.htm>.
29. Walker A. M., Miller D. P. 2012. Tele-operated robot control using attitude aware smartphones. Human-Robot Interaction, 2012 7th ACM/IEEE International Conference, p. 269-270.
30. Wang Y., Song Z., Zhou Y. et al. 2011. Design of a tele-operated field robot. Defense Science Research Conference and Expo, p. 1-4.
31. Xuhong M., Zhizeng L. 2012. The Design and Implement of Embedded Remote Control System in Industrial Robot. Computer Science and Electronics Engineering, 2012 International Conference, p. 332-335.

1 PRIEDAS

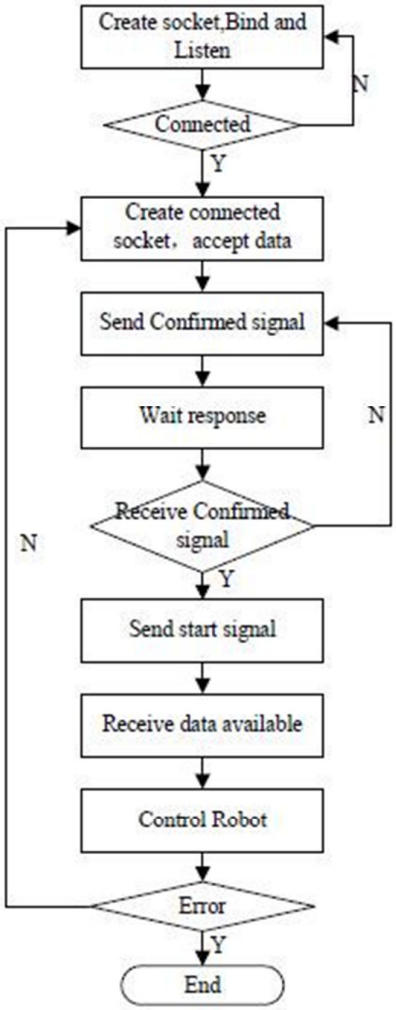
PAPILDOMOS ILIUSTRACIJOS



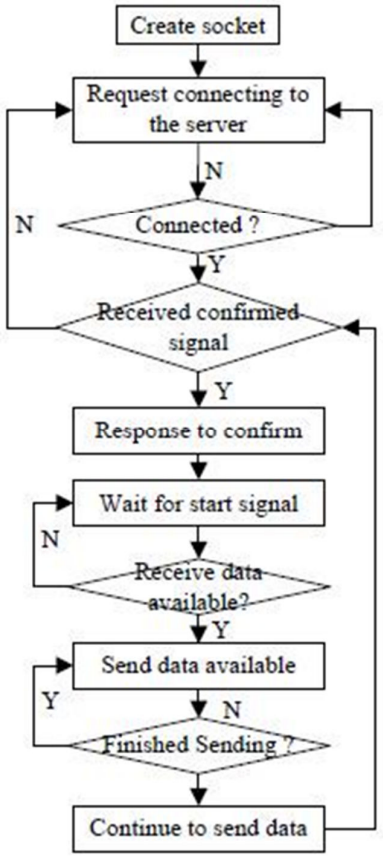
1 pav. Pirštinės su įmontuotais akselerometriniais jutikliais, jungikliais, XBEE moduliu ir atmega8 mikrovaldikliu [23]



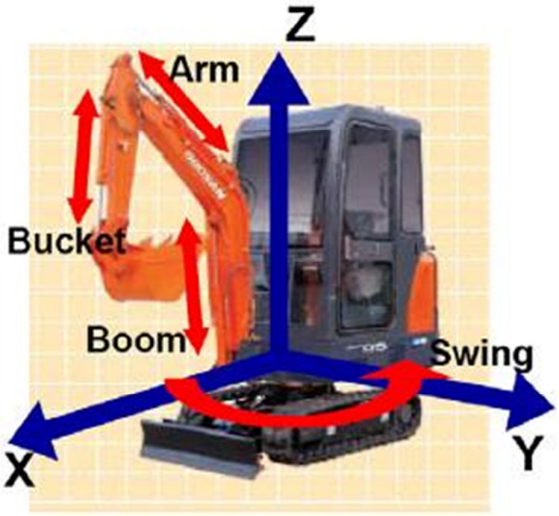
2 pav. Industrinio roboto nuotolinio valdymo sistema [31]



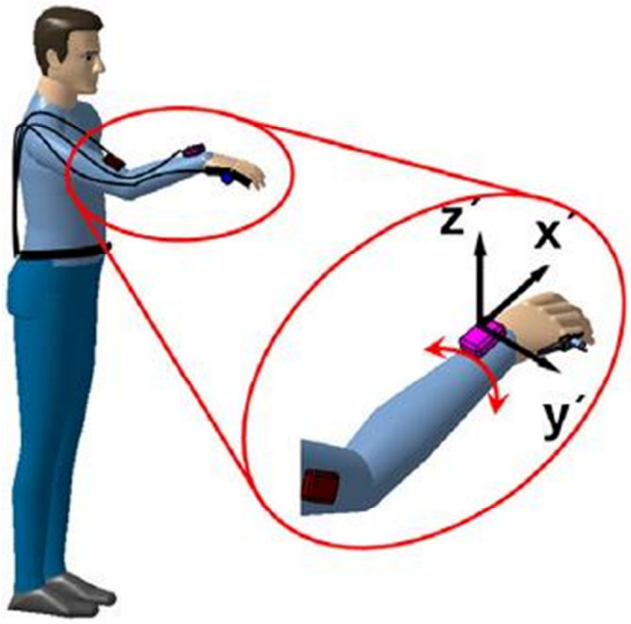
3 pav. Serverio programos duomenų srautų diagrama [31]



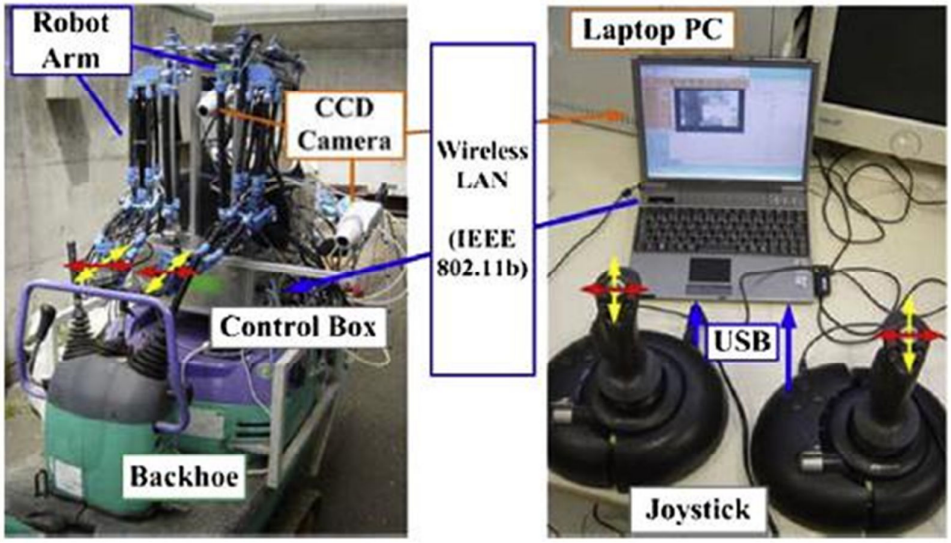
4 pav. Kliento terminalo duomenų srautų diagrama [31]



5 pav. Operatoriaus rankos judesiais valdoma ekskavatoriaus strėlė [13]

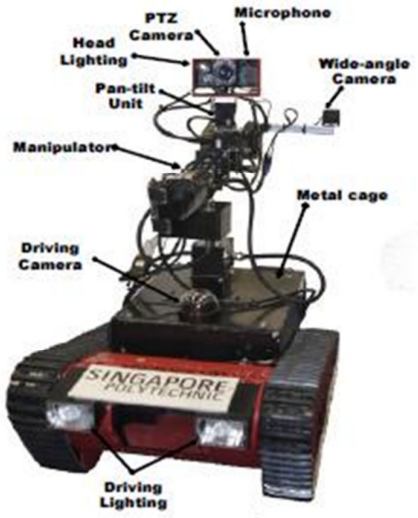


6 pav. Ant operatoriaus rankos uždėtas posūkio kampo jutiklis [13]



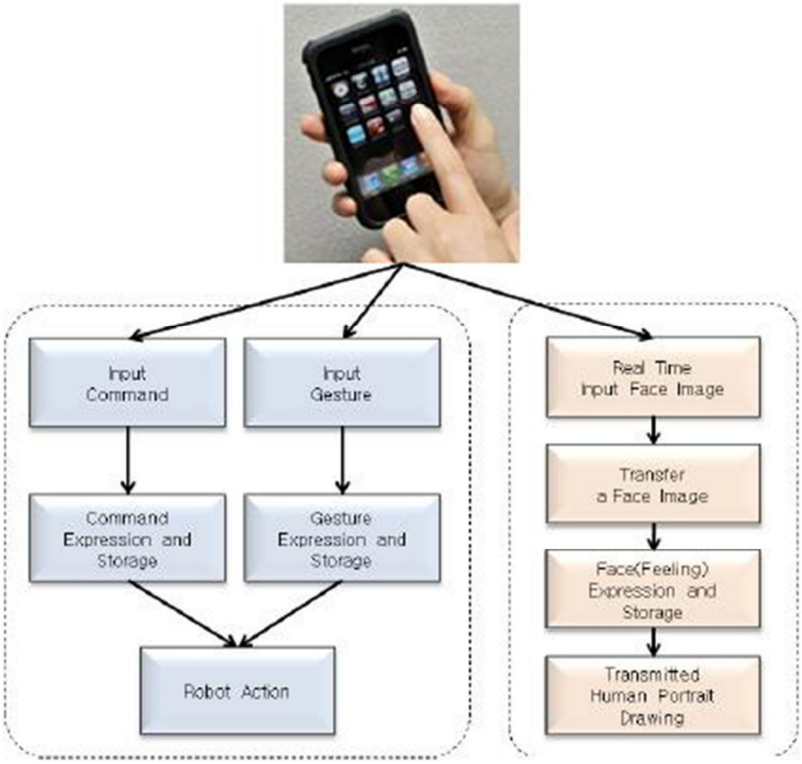
7 pav. Ekskavatoriaus nuotolinio valdymo sistema [24]

8 pav. Nuotolinis roboto (ekskavatoriaus) valdymas [24]

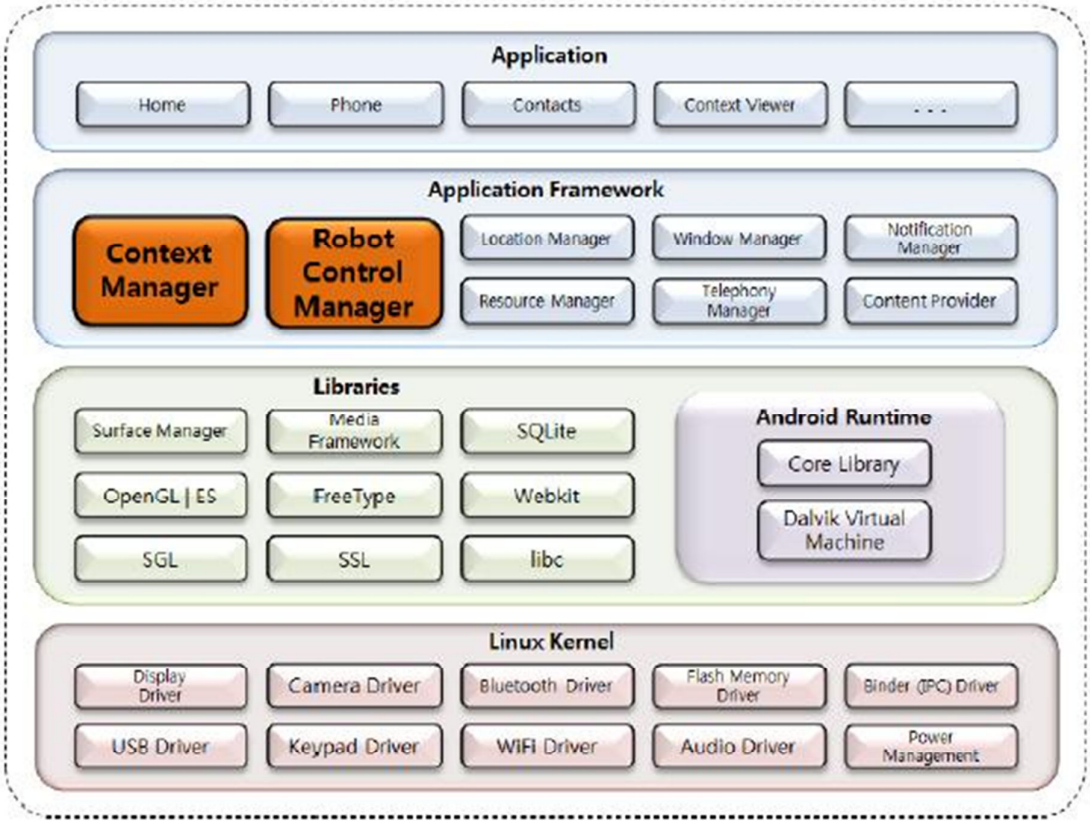


9 pav. Robotas manipuliatorius naudojamas paieškai ir gelbėjimo darbams [30]

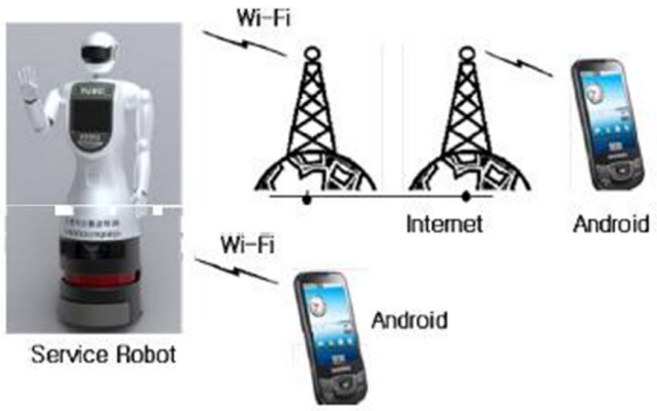
10 pav. Operatoriaus valdymo kompiuterio ekrano vaizdas [30]



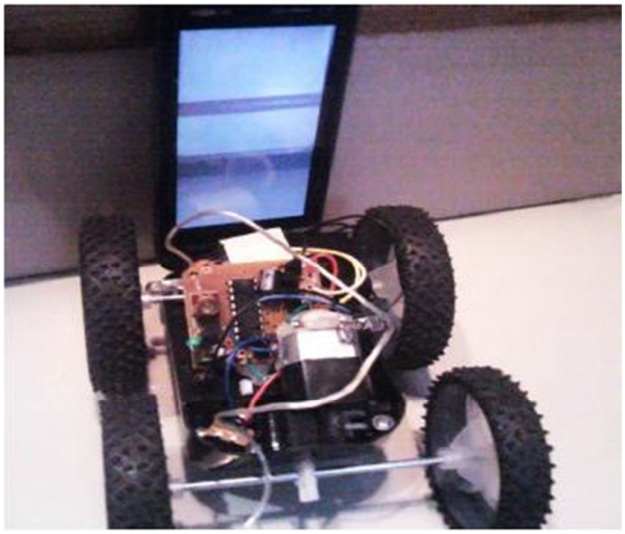
11 pav. Roboto nuotolinio valdymo sistema paremta android išmaniuoju telefonu [26]



12 pav. Android sistemos architektūra [26]

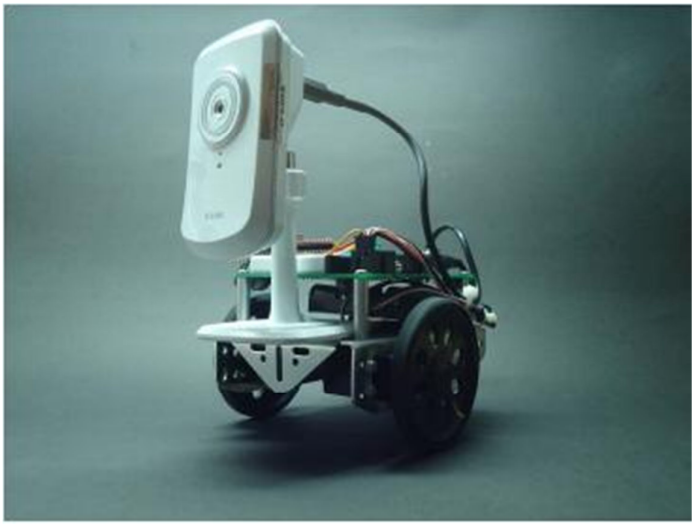


13 pav. Roboto nuotolinio valdymo sistema [10]

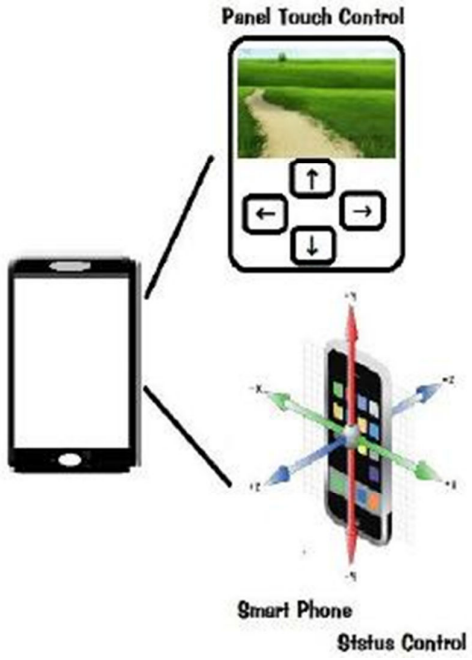


14 pav. Gelbėjimo roboto prototipas [5]

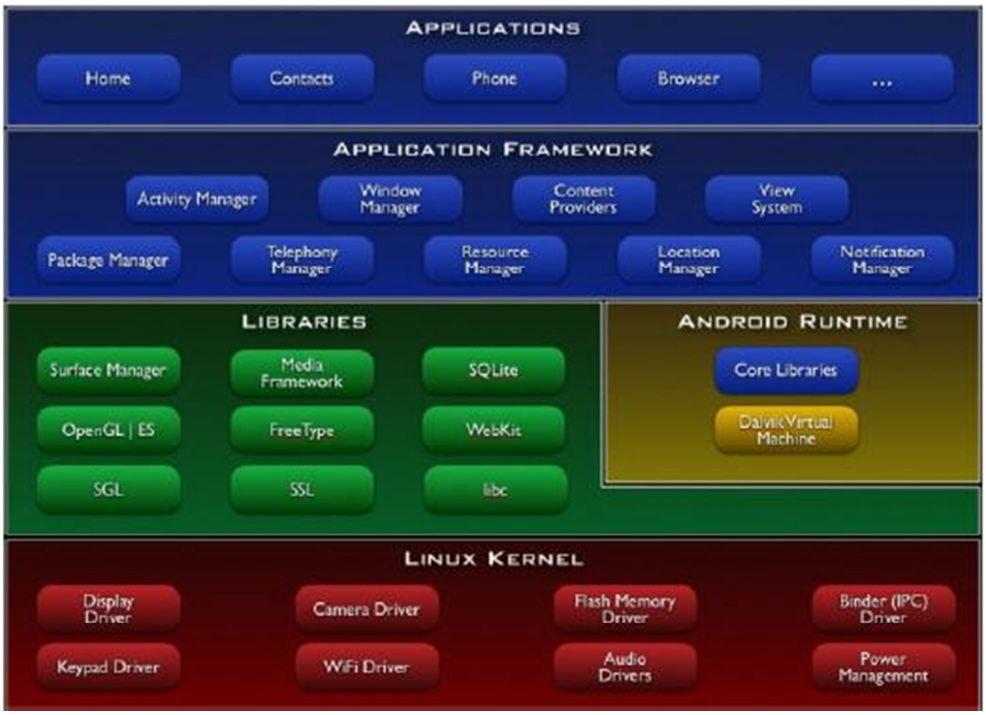
15 pav. Mobiliojo telefono programa skirta roboto valdymui [5]



16 pav. Paieškos robotas [15]



17 pav. Išmaniojo telefono programos valdymo režimai [15]



18 pav. Android operacinės sistemos architektūra [4]

19 pav. Išmaniojo telefono programos vaizdas [4]

20 pav. Roboto nuotolinio valdymo sistema [21]

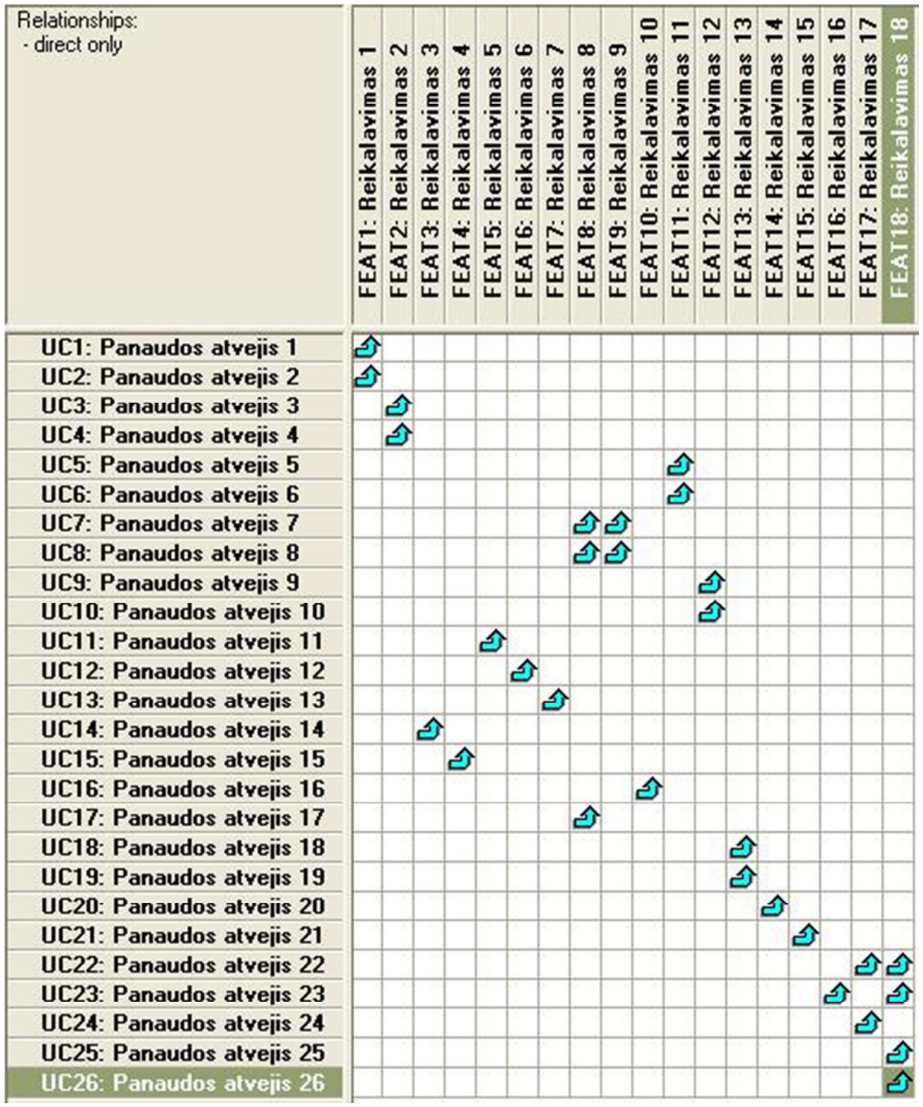
MainActivity.java | activity_main.xml | strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Train Robot</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">Train Robot</string>
    <string name="prisijungti_on">Prisijungti prie roboto</string>
    <string name="prisijungti_off">Atsijungti nuo roboto</string>
    <string name="start_robot">Įjungti robotą</string>
    <string name="stop_robot">Išjungti robotą</string>
    <string name="status"></string>
    <string name="start_magnet">Įjungti magnetą</string>
    <string name="stop_magnet">Išjungti magnetą</string>
    <string name="platforma">Platforma</string>
    <string name="kranas">Kranas</string>
    <string name="kaire">Prieš laikr.</string>
    <string name="desine">Pagal laikr.</string>
    <string name="stop">Stop</string>
    <string name="akselerometras_x">Akselerometras x:</string>
    <string name="akselerometras_y">Akselerometras y:</string>
    <string name="linijinis_pagreitis_x">Linijinis pagreitis x:</string>
    <string name="linijinis_pagreitis_y">Linijinis pagreitis y:</string>
    <string name="kampinis_pagreitis_x">Kampinis pagreitis x:</string>
    <string name="kampinis_pagreitis_y">Kampinis pagreitis y:</string>
    <string name="vykdoma_komanda">Vykdoma komanda:</string>
    <string name="gestu_valdymas">Valdymas tik gestais</string>

</resources>
```

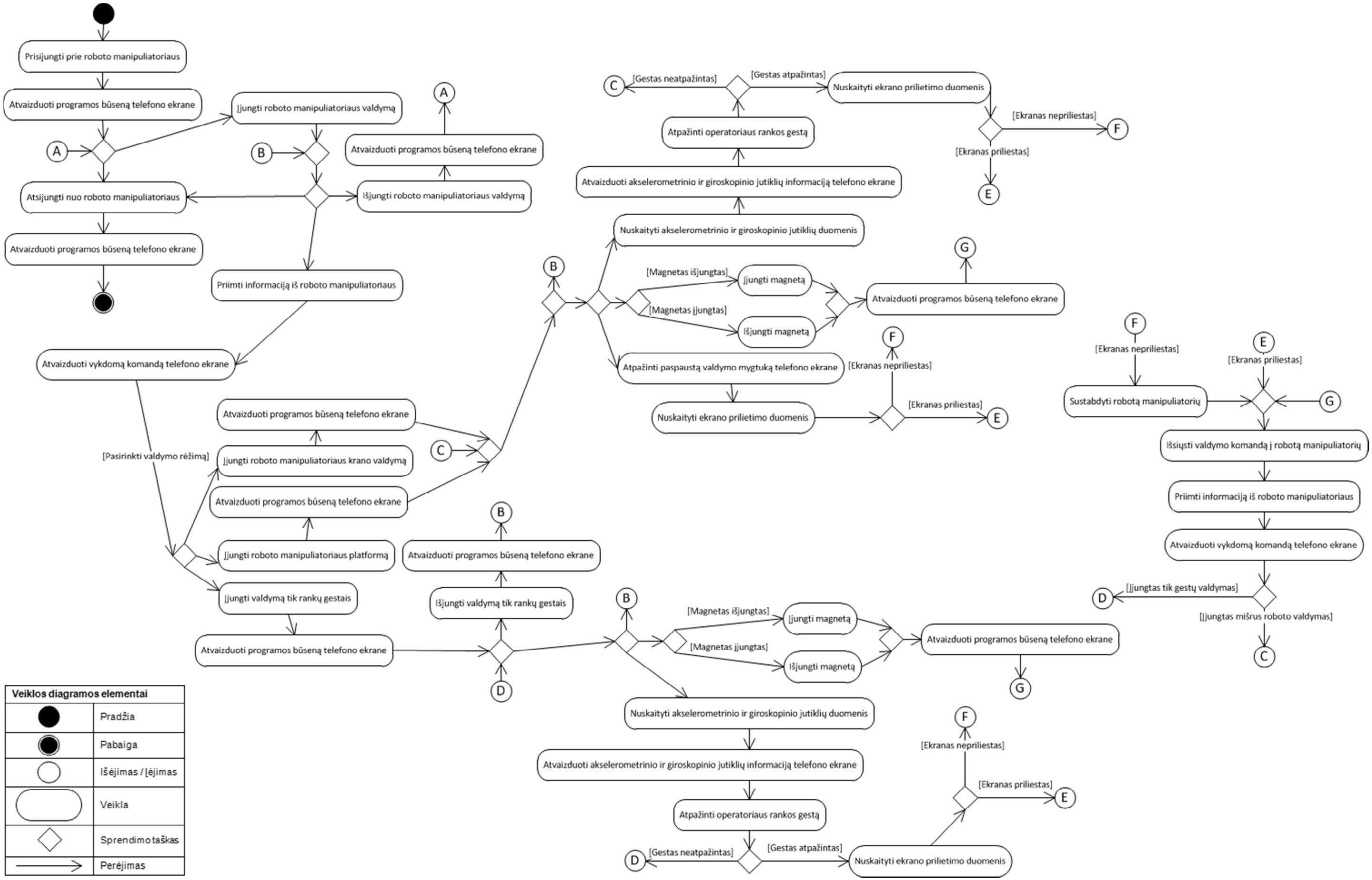
21 pav. Strings.xml failo langas Eclipse programoje

2 PRIEDAS

KURIAMOS ROBOTO MANIPULIATORIAUS NUOTOLINIO VALDYMO SISTEMOS SUSIETUMO MATRICA IR MODELIAVIMO DIAGRAMA



1 pav. Panaudojimo atvejų ir funkcinių reikalavimų susietumo matrica



2 pav. Kuriamos specializuotos išmaniojo telefono programinės įrangos veiklos diagrama

3 PRIEDAS

SUKURTOS ROBOTO MANIPULIATORIAUS PROGRAMINĖS ĮRANGOS ARDUINO KODAS

Pagrindinio mikrovaldiklio Atmega 2560 programinis kodas:

```
#include <Servo.h>
char IncomingByte;
char vykdoma_komanda = 's';
Servo kranas_variklis1;
Servo kranas_variklis2;
Servo kranas_variklis3;
Servo kranas_variklis4;
Servo platforma_variklis3;
Servo platforma_variklis4;
Servo platforma_variklis5;
int kranas_value1 = 24;
int kranas_value2 = 153;
int kranas_value3 = 90;
int kranas_value4 = 65;
int platforma_value3 = 93;
int platforma_value4 = 90;
int platforma_value5 = 116;
int magnet = 30;
boolean komanda_gauta = false;
boolean komanda_gauta_3variklis = false;
boolean komanda_gauta_4variklis = false;
boolean magnetas = false;
boolean nuleidziam_svirti = false;
boolean pakeliam_svirti = false;
boolean serial_send_to_phone = false;
boolean serial_gauta_komanda = false;

void setup(){
  Serial1.begin(19200);
  Serial3.begin(19200);
  kranas_variklis1.attach(8);
  kranas_variklis2.attach(9);
  kranas_variklis3.attach(12);
  kranas_variklis4.attach(13);
  platforma_variklis3.attach(2);
  platforma_variklis4.attach(3);
  platforma_variklis5.attach(4);
  pinMode(magnet, OUTPUT);
}

void loop()
{
  serial_gauta_komanda = false;
  komanda_gauta = false;
  komanda_gauta_3variklis = false;
  komanda_gauta_4variklis = false;

     if (Serial3.available() > 0) {
     IncomingByte = Serial3.read();
     serial_gauta_komanda = true;
         if ((IncomingByte == 'a' || IncomingByte == 'b') || IncomingByte == 's')
         {
           Serial1.println(IncomingByte);
         }
```

```
   }

  //platforma vaziuoja į kairę
  if (IncomingByte == 'a')
  {
    platforma_value3 = 91;
    platforma_value4 = 92;
    if (serial_gauta_komanda)
    {
    vykdoma_komanda = 'a';
    serial_send_to_phone = true;
    }
  }
  //platforma vaziuoja į dešinę
  if (IncomingByte == 'b')
  {
    platforma_value3 = 95;
    platforma_value4 = 88;
    if (serial_gauta_komanda)
    {
    vykdoma_komanda = 'b';
    serial_send_to_phone = true;
    }
  }
  //sustabdom platformą
  if (IncomingByte == 's')
  {
    platforma_value3 = 93;
    platforma_value4 = 90;
    if (serial_gauta_komanda)
    {
    vykdoma_komanda = 's';
    serial_send_to_phone = true;
    }
  }
  //nuleidžiam platformos svirtį
  if (IncomingByte == 'c')
  {
    Serial1.println('s');
    platforma_value3 = 93;
    platforma_value4 = 90;
    if (platforma_value5 == 116)
    {
      nuleidziam_svirti = true;
    }
  }
  if (nuleidziam_svirti == true)
{
  if (platforma_value5 > 24)
  {
  platforma_value5 = platforma_value5 - 1;
  delay(50);
  }
    vykdoma_komanda = 'c';
    serial_send_to_phone = true;
  if (platforma_value5 == 24)
  {
    nuleidziam_svirti = false;
    vykdoma_komanda = 's';
    serial_send_to_phone = true;
  }
```

```
}

//pakeliam platformos svirtį
 if (IncomingByte == 'd')
  {
    Serial1.println('s');
    platforma_value3 = 93;
    platforma_value4 = 90;
    if (platforma_value5 == 24)
    {
      pakeliam_svirti = true;
    }
  }
  if (pakeliam_svirti == true)
{
  if (platforma_value5 < 116)
  {
  platforma_value5 = platforma_value5 + 1;
  delay(50);
  }
  vykdoma_komanda = 'd';
  serial_send_to_phone = true;
  if (platforma_value5 == 116)
  {
    pakeliam_svirti = false;
    vykdoma_komanda = 's';
    serial_send_to_phone = true;
  }
}
 //magnetas
 if (IncomingByte == 'm')
 {
   if (magnetas == false)
   {
     digitalWrite(magnet, HIGH);
     magnetas = true;
     vykdoma_komanda = 'm';
     serial_send_to_phone = true;
   }
 }
 if (IncomingByte == 'n')
 {
   if (magnetas == true)
   {
     digitalWrite(magnet, LOW);
     magnetas = false;
     vykdoma_komanda = 'n';
     serial_send_to_phone = true;
   }
 }
 //kranas sukasi prieš laikrodžio rodyklę
 if (IncomingByte == 'e' && (kranas_value1 < 150 && kranas_value2 == 153))
 {
  kranas_value1 = kranas_value1 + 1;
  komanda_gauta = true;
  vykdoma_komanda = 'e';
  serial_send_to_phone = true;
 }
 if (IncomingByte == 'e' && (kranas_value2 < 153 && kranas_value1 == 24))
 {
   kranas_value2 = kranas_value2 + 1;
```

```
   komanda_gauta = true;
   vykdoma_komanda = 'e';
   serial_send_to_phone = true;
 }
//kranas sukasi pagal laikrodžio rodyklę
 if (IncomingByte == 'f' && (kranas_value1 > 24 && kranas_value2 == 153))
 {
   kranas_value1 = kranas_value1 - 1;
   komanda_gauta = true;
   vykdoma_komanda = 'f';
   serial_send_to_phone = true;
 }
 if (IncomingByte == 'f' && (kranas_value2 > 25 && kranas_value1 == 24))
 {
   kranas_value2 = kranas_value2 - 1;
   komanda_gauta = true;
   vykdoma_komanda = 'f';
   serial_send_to_phone = true;
 }
//krano variklis leidžiamas į apačią
if (IncomingByte == 'g' && kranas_value3 < 126)
{
  kranas_value3 = kranas_value3 + 1;
  komanda_gauta_3variklis = true;
  vykdoma_komanda = 'g';
  serial_send_to_phone = true;
}
//krano variklis keliamas į viršų
if (IncomingByte == 'h' && kranas_value3 > 56)
{
  kranas_value3 = kranas_value3 - 1;
  komanda_gauta_3variklis = true;
  vykdoma_komanda = 'h';
  serial_send_to_phone = true;
}
//krano variklis keliamas į viršų
if (IncomingByte == 'i' && kranas_value4 < 160)
{
  kranas_value4 = kranas_value4 + 1;
  komanda_gauta_4variklis = true;
  vykdoma_komanda = 'i';
  serial_send_to_phone = true;
}
//krano variklis leidžiamas į apačią
if (IncomingByte == 'j' && kranas_value4 > 28)
{
  kranas_value4 = kranas_value4 - 1;
  komanda_gauta_4variklis = true;
  vykdoma_komanda = 'j';
  serial_send_to_phone = true;
}

 if (komanda_gauta == true)
 {
  delay(100);
 }
  if (komanda_gauta_3variklis == true)
 {
  delay(70);
 }
 if (komanda_gauta_4variklis == true)
```

```
 {
   delay(70);
 }
 //valdymo signalai į variklius
 platforma_variklis3.write(platforma_value3);
 platforma_variklis4.write(platforma_value4);
 platforma_variklis5.write(platforma_value5);
 kranas_variklis1.write(kranas_value1);
 kranas_variklis2.write(kranas_value2);
 kranas_variklis3.write(kranas_value3);
 kranas_variklis4.write(kranas_value4);
 //informacijos siuntimas į išmanųjį telefoną
 if (serial_send_to_phone == true)
 {
  Serial3.println(vykdoma_komanda);
  serial_send_to_phone = false;
 }
}
```

Papildomo mikrovaldiklio Atmega 328 programinis kodas:

```
#include <Servo.h>
Servo platforma_variklis1;
Servo platforma_variklis2;
char IncomingByte;
int variklis_value_1 = 89;
int variklis_value_2 = 93;
void setup(){
  Serial.begin(19200);
  platforma_variklis1.attach(5);
  platforma_variklis2.attach(6);
}
void loop()
{
  if (Serial.available() > 0) {
  IncomingByte = Serial.read();
  }
  //platforma važiuoja į kairę
  if (IncomingByte == 'a')
  {
    variklis_value_1 = 92;
    variklis_value_2 = 91;
  }
  //platforma važiuoja į dešinę
  if (IncomingByte == 'b')
  {
    variklis_value_1 = 87;
    variklis_value_2 = 95;
  }
  //sustabdom platformą
  if (IncomingByte == 's')
  {
    variklis_value_1 = 89;
    variklis_value_2 = 93;
  }
  //valdymo signalai į variklius
 platforma_variklis1.write(variklis_value_1);
 platforma_variklis2.write(variklis_value_2);
}
```

4 PRIEDAS

SUKURTOS IŠMANIOJO TELEFONO PROGRAMINĖS ĮRANGOS VARTOTOJO GRAFINĖS SĄSAJOS XML KODAS (ACTIVITY_MAIN.XML FAILAS)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ToggleButton
        android:id="@+id/toggleButton3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:onClick="MagnetoValdymas"
        android:textOff="@string/start_magnet"
        android:textOn="@string/stop_magnet" />

    <LinearLayout
        android:id="@+id/LinearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal" >

        <ToggleButton
            android:id="@+id/toggleButton1"
            android:layout_width="170dp"
            android:layout_height="wrap_content"
            android:gravity="center_vertical|center_horizontal"
            android:onClick="PrisijungtiPrieRoboto"
            android:textOff="@string/prisijungti_on"
            android:textOn="@string/prisijungti_off" />

        <ToggleButton
            android:id="@+id/toggleButton2"
            android:layout_width="170dp"
            android:layout_height="wrap_content"
            android:onClick="IjungtiRobota"
            android:textOff="@string/start_robot"
            android:textOn="@string/stop_robot" />
    </LinearLayout>

    <TextView
        android:id="@+id/tV_status"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/LinearLayout1"
        android:gravity="center_horizontal"
        android:keepScreenOn="true"
        android:text="@string/status"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textSize="15sp" />

    <LinearLayout
        android:id="@+id/LinearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tV_status"
        android:gravity="left"
        android:orientation="vertical" >

        <LinearLayout
            android:id="@+id/LinearLayout8"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/tV_accel_x"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="5dp"
                android:paddingRight="0dp"
                android:text="@string/akselerometras_x"
                android:textAppearance="?android:attr/textAppearanceSmall" />

            <TextView
                android:id="@+id/textView_vnt1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="@dimen/padding_small"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:textIsSelectable="true" />
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout9"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/tV_accel_y"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="5dp"
                android:paddingRight="0dp"
                android:text="@string/akselerometras_y"
                android:textAppearance="?android:attr/textAppearanceSmall" />

            <TextView
                android:id="@+id/textView_vnt2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="@dimen/padding_small"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:textIsSelectable="true" />
        </LinearLayout>
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/toggleButton3"
        android:layout_alignParentLeft="true"
        android:gravity="center_horizontal" >

        <Button
```

```
            android:id="@+id/button1"
            style="?android:attr/borderlessButtonStyle"
            android:layout_width="140dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.41"
            android:onClick="Platforma"
            android:text="@string/platforma" />

        <Button
            android:id="@+id/button2"
            style="?android:attr/borderlessButtonStyle"
            android:layout_width="140dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.45"
            android:onClick="Kranas"
            android:text="@string/kranas" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/linearLayout4"
        android:layout_alignParentLeft="true"
        android:gravity="center_horizontal" >

        <Button
            android:id="@+id/button3"
            style="?android:attr/borderlessButtonStyle"
            android:layout_width="179dp"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:onClick="PriesLaikrodi"
            android:text="@string/kaire" />

        <Button
            android:id="@+id/button5"
            style="?android:attr/borderlessButtonStyle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:onClick="PagalLaikrodi"
            android:text="@string/desine" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/LinearLayout3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/LinearLayout2"
        android:gravity="left"
        android:orientation="vertical" >

        <LinearLayout
            android:id="@+id/LinearLayout7"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="left"
            android:orientation="vertical" >
```

```
    <LinearLayout
        android:id="@+id/LinearLayout10"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/tV_lin_accel_x"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:paddingRight="0dp"
            android:text="@string/linijinis_pagreitis_x"
            android:textAppearance="?android:attr/textAppearanceSmall" />

        <TextView
            android:id="@+id/textView_vnt3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="@dimen/padding_small"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textIsSelectable="true" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/LinearLayout11"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/tV_lin_accel_y"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:paddingRight="0dp"
            android:text="@string/linijinis_pagreitis_y"
            android:textAppearance="?android:attr/textAppearanceSmall" />

        <TextView
            android:id="@+id/textView_vnt4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="@dimen/padding_small"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textIsSelectable="true" />
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:id="@+id/LinearLayout12"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/tV_giro_x"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="0dp"
        android:text="@string/kampinis_pagreitis_x"
        android:textAppearance="?android:attr/textAppearanceSmall" />
```

```
        <TextView
            android:id="@+id/textView_vnt5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="@dimen/padding_small"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textIsSelectable="true" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/LinearLayout14"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/tV_giro_y"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:paddingRight="0dp"
            android:text="@string/kampinis_pagreitis_y"
            android:textAppearance="?android:attr/textAppearanceSmall" />

        <TextView
            android:id="@+id/textView_vnt6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="@dimen/padding_small"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textIsSelectable="true" />
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:id="@+id/LinearLayout6"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/LinearLayout3" >

    <TextView
        android:id="@+id/textView_vykdoma_komanda"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:text="@string/vykdoma_komanda"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textSize="15sp" />

    <TextView
        android:id="@+id/textView_BT_input"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textIsSelectable="false"
        android:textSize="15sp" />
</LinearLayout>

<ToggleButton
    android:id="@+id/toggleButton4"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/linearLayout5"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="11dp"
        android:onClick="ValdymasTikGestais"
        android:text="@string/gestu_valdymas"
        android:textOff="@string/gestu_valdymas"
        android:textOn="@string/gestu_valdymas" />

</RelativeLayout>
```

5 PRIEDAS

SUKURTOS IŠMANIOJO TELEFONO PROGRAMINĖS ĮRANGOS PAGRINDINĖS KLASĖS JAVA KODAS (MAINACTIVITY.JAVA FAILAS)

```
package aurimas.vaitekunas.train.robot;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.os.Handler;
import android.text.Html;
import android.view.MotionEvent;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener {
     private TextView tV_status;
     private TextView tV_accel_x;
     private TextView tV_accel_y;
     private TextView tV_lin_accel_x;
     private TextView tV_lin_accel_y;
     private TextView tV_giro_x;
     private TextView tV_giro_y;
     private TextView textView_BT_input;
     private TextView textView_vnt1;
     private TextView textView_vnt2;
     private TextView textView_vnt3;
     private TextView textView_vnt4;
     private TextView textView_vnt5;
     private TextView textView_vnt6;
     private boolean bluetooth_open_on_off = false;
     private boolean robot_control_on_off = false;
     private BluetoothAdapter bluetoothadapter;
     private BluetoothDevice bluetoothdevice;
     private BluetoothSocket bluetoothsocket;
     private OutputStream outputstream;
     private InputStream inputstream;
     private String message;
     private String message_command;
     private static final int ENABLE_BLUETOOTH = 1;
     private SensorManager sensor_manager;
     private Sensor gyro;
     private Sensor accelerometer;
     private Sensor linear_accelerometer;
     private Sensor sensor;
     private float akselerometras_x = 0;
```

```
private float akselerometras_y = 0;
private float linijinis_akselerometras_x = 0;
private float linijinis_akselerometras_y = 0;
private float giroskopas_x = 0;
private float giroskopas_y = 0;
private boolean platforma = false;
private boolean kranas = false;
private boolean magnetas = false;
private Handler handler;
volatile boolean stop_bluetooth_input = false;
Runnable runnable;
private boolean valdymas_gestais = false;
private boolean galima_vykdyti_komanda = true;
private boolean ekranas_priliestas = false;
private boolean pagal_laikr = false;
private boolean pries_laikr = false;

@Override
public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     handler = new Handler();
     tV_status = (TextView) findViewById(R.id.tV_status);
     tV_accel_x = (TextView) findViewById(R.id.tV_accel_x);
     tV_accel_y = (TextView) findViewById(R.id.tV_accel_y);
     tV_lin_accel_x = (TextView) findViewById(R.id.tV_lin_accel_x);
     tV_lin_accel_y = (TextView) findViewById(R.id.tV_lin_accel_y);
     tV_giro_x = (TextView) findViewById(R.id.tV_giro_x);
     tV_giro_y = (TextView) findViewById(R.id.tV_giro_y);
     textView_BT_input = (TextView) findViewById(R.id.textView_BT_input);
     textView_vnt1 = (TextView) findViewById(R.id.textView_vnt1);
     textView_vnt2 = (TextView) findViewById(R.id.textView_vnt2);
     textView_vnt3 = (TextView) findViewById(R.id.textView_vnt3);
     textView_vnt4 = (TextView) findViewById(R.id.textView_vnt4);
     textView_vnt5 = (TextView) findViewById(R.id.textView_vnt5);
     textView_vnt6 = (TextView) findViewById(R.id.textView_vnt6);
     bluetoothadapter = BluetoothAdapter.getDefaultAdapter();
     sensor_manager = (SensorManager) getSystemService(SENSOR_SERVICE);
     gyro = sensor_manager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
     accelerometer = sensor_manager
               .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
     linear_accelerometer = sensor_manager
               .getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
}

public boolean onTouchEvent(MotionEvent event) {
     if (robot_control_on_off == true && bluetooth_open_on_off == true) {
          int eventaction = event.getAction();
          switch (eventaction) {
          case MotionEvent.ACTION_DOWN:
               ekranas_priliestas = true;
               linijinis_akselerometras_x = 0;
               linijinis_akselerometras_y = 0;
               giroskopas_x = 0;
               giroskopas_y = 0;
               akselerometras_x = 0;
               akselerometras_y = 0;
               if (message_command != "f" && pagal_laikr == true) {
                    message_command = "f";
                    try {
                         send_command_bluetooth();
```

```
                    } catch (IOException ex) {
                    }
                }
                if (message_command != "e" && pries_laikr == true) {
                    message_command = "e";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                break;
            case MotionEvent.ACTION_UP:
                ekranas_priliestas = false;
                galima_vykdyti_komanda = true;
                pagal_laikr = false;
                pries_laikr = false;
                if (message_command != "s") {
                    message_command = "s";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                } else {
                    textView_BT_input.setText("robotas sustabdytas");
                }
                break;
            }
        }
        return true;
    }

    public void PrisijungtiPrieRoboto(View view) {
        if (bluetooth_open_on_off == false) {
            enable_bluetooth();
            return;
        }

        if (bluetooth_open_on_off == true) {
            try {
                close_bluetooth();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

    public void IjungtiRobota(View view) {
        if (bluetooth_open_on_off == true) {
            if (robot_control_on_off == false) {
                robot_control_on_off = true;
                tV_status.setText("Roboto valdymas įjungtas");
                if (platforma == true) {
                    tV_status.setText("Įjungtas platformos valdymas");
                }
                if (kranas == true) {
                    tV_status.setText("Įjungtas krano valdymas");
                }
                textView_BT_input.setText("robotas sustabdytas");
                return;
            }
```

```
            if (robot_control_on_off == true) {
                robot_control_on_off = false;
                tV_status.setText("Roboto valdymas išjungtas");
                if (message_command != "s") {
                    message_command = "s";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                } else {
                    textView_BT_input.setText("robotas sustabdytas");
                }
            }
        }
    }

    public void PagalLaikrodi(View view) {
        if ((robot_control_on_off == true && bluetooth_open_on_off) == true
                && (valdymas_gestais == false && kranas == true)) {
            pagal_laikr = true;
        }
    }

    public void PriesLaikrodi(View view) {
        if ((robot_control_on_off == true && bluetooth_open_on_off) == true
                && (valdymas_gestais == false && kranas == true)) {
            pries_laikr = true;
        }
    }

    public void Platforma(View view) {
        if ((robot_control_on_off == true && bluetooth_open_on_off == true)
                && valdymas_gestais == false) {
            platforma = true;
            kranas = false;
            tV_status.setText("Įjungtas platformos valdymas");
            if (message_command != "s") {
                message_command = "s";
                try {
                    send_command_bluetooth();
                } catch (IOException ex) {
                }
            } else {
                textView_BT_input.setText("robotas sustabdytas");
            }
        }
    }

    public void Kranas(View view) {
        if ((robot_control_on_off == true && bluetooth_open_on_off == true)
                && valdymas_gestais == false) {
            platforma = false;
            kranas = true;
            tV_status.setText("Įjungtas krano valdymas");
            if (message_command != "s") {
                message_command = "s";
                try {
                    send_command_bluetooth();
                } catch (IOException ex) {
                }
            } else {
```

```
                textView_BT_input.setText("robotas sustabdytas");
            }
        }
    }

    public void MagnetoValdymas(View view) {
        if (robot_control_on_off == true && bluetooth_open_on_off == true) {
            if (magnetas == false) {
                if (message_command != "m") {
                    message_command = "m";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                magnetas = true;
                return;
            }
            if (magnetas == true) {
                if (message_command != "n") {
                    message_command = "n";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                magnetas = false;
                return;
            }
        }
    }

    private void enable_bluetooth() {
        if (!bluetoothadapter.isEnabled()) {
            Intent bluetoothIntent = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(bluetoothIntent, ENABLE_BLUETOOTH);
        } else {
            tV_status.setText("Bluetooth įjungtas");
            find_bluetooth();
        }
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == ENABLE_BLUETOOTH)
            if (resultCode == RESULT_OK) {
                tV_status.setText("Bluetooth įjungtas");
                find_bluetooth();
            } else {
                tV_status.setText("Vykdymas negalimas");
            }
    }

    private void find_bluetooth() {
        Set<BluetoothDevice> pairedDevice = bluetoothadapter.getBondedDevices();
        if (pairedDevice.size() > 0) {
            for (BluetoothDevice device : pairedDevice) {
                if (device.getName().equals("Serial Adaptor")) {
                    bluetoothdevice = device;
                    tV_status.setText("Bluetooth rastas");
                    break;
```

```
                } else
                    tV_status.setText("Bluetooth nerastas");
            }
            if (bluetoothdevice.getName().equals("Serial Adaptor")) {
                try {
                    connect_bluetooth();
                } catch (IOException ex) {
                }
            }
        }
    }

    private void connect_bluetooth() throws IOException {
        tV_status.setText("Jungiamasi...");
        UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
        bluetoothsocket = bluetoothdevice
                .createRfcommSocketToServiceRecord(uuid);
        bluetoothsocket.connect();
        outputstream = bluetoothsocket.getOutputStream();
        inputstream = bluetoothsocket.getInputStream();
        tV_status.setText("Prisijungta sėkmingai");
        bluetooth_open_on_off = true;
        bluetooth_input_listening();
        startThread();
    }

    private void close_bluetooth() throws IOException {
        stopThread();
        outputstream.close();
        inputstream.close();
        bluetoothsocket.close();
        tV_status.setText("Atsijungta sėkmingai");
        bluetooth_open_on_off = false;
    }

    private void send_command_bluetooth() throws IOException {
        message = message_command;
        outputstream.write(message.getBytes());
    }

    protected void onResume() {
        super.onResume();
        sensor_manager.registerListener(this, gyro,
                SensorManager.SENSOR_DELAY_GAME);
        sensor_manager.registerListener(this, accelerometer,
                SensorManager.SENSOR_DELAY_GAME);
        sensor_manager.registerListener(this, linear_accelerometer,
                SensorManager.SENSOR_DELAY_GAME);
    }

    protected void onPause() {
        super.onPause();
        if (robot_control_on_off == true && bluetooth_open_on_off == true) {
            if (message_command != "s") {
                message_command = "s";
                try {
                    send_command_bluetooth();
                } catch (IOException ex) {
                }
            }
        }
```

```
        sensor_manager.unregisterListener(this, gyro);
        sensor_manager.unregisterListener(this, accelerometer);
        sensor_manager.unregisterListener(this, linear_accelerometer);
    }

    protected void onStop() {
        super.onStop();
        stopThread();
    }

    private synchronized void stopThread() {
        stop_bluetooth_input = true;
    }

    private synchronized void startThread() {
        new Thread(runnable).start();
    }

    public void ValdymasTikGestais(View view) {
        if (robot_control_on_off == true && bluetooth_open_on_off == true) {
            if (valdymas_gestais == false) {
                valdymas_gestais = true;
                tV_status.setText("Įjungtas valdymas tik gestais");
                if (message_command != "s") {
                    message_command = "s";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                } else {
                    textView_BT_input.setText("robotas sustabdytas");
                }
                return;
            }
            if (valdymas_gestais == true) {
                valdymas_gestais = false;
                tV_status.setText("Išjungtas valdymas tik gestais");
                if (message_command != "s") {
                    message_command = "s";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                } else {
                    textView_BT_input.setText("robotas sustabdytas");
                }
            }
        }
    }

    void VykdomosKomandosNustatymas(String vykdoma_komanda) {
        if (vykdoma_komanda.contentEquals("a")) {
            textView_BT_input.setText("platforma važiuoja į kairę");
            return;
        }

        if (vykdoma_komanda.contentEquals("b")) {
            textView_BT_input.setText("platforma važiuoja į dešinę");
            return;
        }
```

```
        if (vykdoma_komanda.contentEquals("c")) {
            textView_BT_input.setText("svirtis nuleista");
            return;
        }

        if (vykdoma_komanda.contentEquals("d")) {
            textView_BT_input.setText("svirtis pakelta");
            return;
        }

        if (vykdoma_komanda.contentEquals("e")) {
            textView_BT_input.setText("kranas sukasi prieš laikrodžio rodyklę");
            return;
        }

        if (vykdoma_komanda.contentEquals("f")) {
            textView_BT_input.setText("kranas sukasi pagal laikrodžio rodyklę");
            return;
        }

        if (vykdoma_komanda.contentEquals("g")) {
            textView_BT_input
                    .setText("trečias krano variklis leidžiasi į apačią");
            return;
        }

        if (vykdoma_komanda.contentEquals("h")) {
            textView_BT_input.setText("trečias krano variklis kyla į viršų");
            return;
        }

        if (vykdoma_komanda.contentEquals("j")) {
            textView_BT_input.setText("ketvirtas krano variklis kyla į viršų");
            return;
        }

        if (vykdoma_komanda.contentEquals("i")) {
            textView_BT_input
                    .setText("ketvirtas krano variklis leidžiasi į apačią");
            return;
        }

        if (vykdoma_komanda.contentEquals("m")) {
            textView_BT_input.setText("magnetas įjungtas");
            return;
        }

        if (vykdoma_komanda.contentEquals("n")) {
            textView_BT_input.setText("magnetas išjungtas");
            return;
        }

        if (vykdoma_komanda.contentEquals("s")) {
            textView_BT_input.setText("robotas sustabdytas");
        }
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent event) {
```

```
if (robot_control_on_off == true && bluetooth_open_on_off == true) {
     sensor = event.sensor;
     if (sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
          linijinis_akselerometras_x = event.values[0];
          linijinis_akselerometras_y = event.values[1];
          tV_lin_accel_x.setText("Linijinis pagreitis x: "
                    + linijinis_akselerometras_x);
          textView_vnt3.setText(Html.fromHtml("m/s<sup>2</sup>"));
          tV_lin_accel_y.setText("Linijinis pagreitis y: "
                    + linijinis_akselerometras_y);
          textView_vnt4.setText(Html.fromHtml("m/s<sup>2</sup>"));
     }
     if (sensor.getType() == Sensor.TYPE_GYROSCOPE) {
          giroskopas_x = event.values[0];
          giroskopas_y = event.values[1];
          tV_giro_x.setText("Kampinis pagreitis x: " + giroskopas_x);
          textView_vnt5.setText("rad/s");
          tV_giro_y.setText("Kampinis pagreitis y: " + giroskopas_y);
          textView_vnt6.setText("rad/s");
     }
     if (sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
          akselerometras_x = event.values[0];
          akselerometras_y = event.values[1];
          tV_accel_x.setText("Akselerometras x: " + akselerometras_x);
          textView_vnt1.setText(Html.fromHtml("m/s<sup>2</sup>"));
          tV_accel_y.setText("Akselerometras y: " + akselerometras_y);
          textView_vnt2.setText(Html.fromHtml("m/s<sup>2</sup>"));
     }
     if (valdymas_gestais == true) {
          // platformos valdymas
          // platforma važiuoja į kairę
          if (linijinis_akselerometras_y < -2.5
                    || linijinis_akselerometras_y > 2.5) {
               if (galima_vykdyti_komanda == true
                         && akselerometras_x > 3.5 && giroskopas_y < -0.7) {
                    galima_vykdyti_komanda = false;
                    if (message_command != "a"
                              && ekranas_priliestas == true) {
                         message_command = "a";
                         try {
                              send_command_bluetooth();
                         } catch (IOException ex) {
                         }
                    }
                    return;
               }
               // platforma važiuoja į dešinę
               if (galima_vykdyti_komanda == true
                         && akselerometras_x < -3.5 && giroskopas_y > 0.7) {
                    galima_vykdyti_komanda = false;
                    if (message_command != "b"
                              && ekranas_priliestas == true) {
                         message_command = "b";
                         try {
                              send_command_bluetooth();
                         } catch (IOException ex) {
                         }
                    }
                    return;
               }
               // nuleidžiam svirtį
```

```
                    if (galima_vykdyti_komanda == true
                              && akselerometras_y < -4.5 && giroskopas_x < -1.5)
{
                         galima_vykdyti_komanda = false;
                         if (message_command != "c"
                                   && ekranas_priliestas == true) {
                              message_command = "c";
                              try {
                                   send_command_bluetooth();
                              } catch (IOException ex) {
                              }
                         }
                         return;
                    }
                    // pakeliam svirtį
                    if (galima_vykdyti_komanda == true
                              && akselerometras_y > 3.2 && giroskopas_x > 1.2) {
                         galima_vykdyti_komanda = false;
                         if (message_command != "d"
                                   && ekranas_priliestas == true) {
                              message_command = "d";
                              try {
                                   send_command_bluetooth();
                              } catch (IOException ex) {
                              }
                         }
                         return;
                    }
               } else {
                    // krano valdymas
                    // kranas sukasi prieš laikrodžio rodyklę
                    if (linijinis_akselerometras_x > 4.0
                              || linijinis_akselerometras_x < -4.0) {
                         if (galima_vykdyti_komanda == true
                                   && akselerometras_x > 5.0
                                   && giroskopas_y < -0.7) {
                              galima_vykdyti_komanda = false;
                              if (message_command != "e"
                                        && ekranas_priliestas == true) {
                                   message_command = "e";
                                   try {
                                        send_command_bluetooth();
                                   } catch (IOException ex) {
                                   }
                              }
                              return;
                         }
                         // kranas sukasi pagal laikrodžio rodyklę
                         if (galima_vykdyti_komanda == true
                                   && akselerometras_x < -5.0
                                   && giroskopas_y > 0.7) {
                              galima_vykdyti_komanda = false;
                              if (message_command != "f"
                                        && ekranas_priliestas == true) {
                                   message_command = "f";
                                   try {
                                        send_command_bluetooth();
                                   } catch (IOException ex) {
                                   }
                              }
                              return;
```

```
                    }
                } else {
                    // krano varikliu valdymas
                    // leidžiam variklį į apačią
                    if (galima_vykdyti_komanda == true
                            && akselerometras_x > 8.0
                            && giroskopas_y < -1.5) {
                        galima_vykdyti_komanda = false;
                        if (message_command != "g"
                                && ekranas_priliestas == true) {
                            message_command = "g";
                            try {
                                send_command_bluetooth();
                            } catch (IOException ex) {
                            }
                        }
                        return;
                    }
                    // keliam variklį į viršų
                    if (galima_vykdyti_komanda == true
                            && akselerometras_x < -8.0
                            && giroskopas_y > 1.5) {
                        galima_vykdyti_komanda = false;
                        if (message_command != "h"
                                && ekranas_priliestas == true) {
                            message_command = "h";
                            try {
                                send_command_bluetooth();
                            } catch (IOException ex) {
                            }
                        }
                        return;
                    }
                    // leidžiam variklį į apačią
                    if (galima_vykdyti_komanda == true
                            && akselerometras_y < -4.0
                            && giroskopas_x < -1.5) {
                        galima_vykdyti_komanda = false;
                        if (message_command != "i"
                                && ekranas_priliestas == true) {
                            message_command = "i";
                            try {
                                send_command_bluetooth();
                            } catch (IOException ex) {
                            }
                        }
                        return;
                    }
                    // keliam variklį į viršų
                    if (galima_vykdyti_komanda == true
                            && akselerometras_y > 4.0 && giroskopas_x >
1.5) {
                        galima_vykdyti_komanda = false;
                        if (message_command != "j"
                                && ekranas_priliestas == true) {
                            message_command = "j";
                            try {
                                send_command_bluetooth();
                            } catch (IOException ex) {
                            }
                        }
```

```
                    return;
                }
                // sustabdom robotą
                if (akselerometras_x > -0.5 && akselerometras_x < 0.5
                        && akselerometras_y > -0.5
                        && akselerometras_y < 0.5) {
                    galima_vykdyti_komanda = true;
                    linijinis_akselerometras_x = 0;
                    linijinis_akselerometras_y = 0;
                    giroskopas_x = 0;
                    giroskopas_y = 0;
                    akselerometras_x = 0;
                    akselerometras_y = 0;
                    if (message_command != "s") {
                        message_command = "s";
                        try {
                            send_command_bluetooth();
                        } catch (IOException ex) {
                        }
                    } else {
                        textView_BT_input
                                .setText("robotas sustabdytas");
                    }
                    return;
                }
            }
        }
    } else {
        // platformos valdymas
        if (platforma == true) {
            // platforma važiuoja į kairę
            if (akselerometras_x > 3.5 && giroskopas_y < -1.0) {
                if (message_command != "a"
                        && ekranas_priliestas == true) {
                    message_command = "a";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                return;
            }
            // platforma važiuoja į dešinę
            if (akselerometras_x < -3.5 && giroskopas_y > 1.0) {
                if (message_command != "b"
                        && ekranas_priliestas == true) {
                    message_command = "b";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                return;
            }
            // nuleidžiam svirtį
            if (akselerometras_y < -3.5 && giroskopas_x < -1.0) {
                if (message_command != "c"
                        && ekranas_priliestas == true) {
                    message_command = "c";
                    try {
                        send_command_bluetooth();
```

```
                    } catch (IOException ex) {
                    }
                }
                return;
            }
            // pakeliam svirtį
            if (akselerometras_y > 3.5 && giroskopas_x > 1.0) {
                if (message_command != "d"
                        && ekranas_priliestas == true) {
                    message_command = "d";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                return;
            }
            // sustabdom platformą
            if ((akselerometras_x > -2.0 && akselerometras_x < 2.0)
                    && (akselerometras_y > -2.0 && akselerometras_y <
2.0)) {
                if (message_command != "s") {
                    message_command = "s";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                return;
            }
        }

        // krano valdymas
        if (kranas == true) {
            // leidžiam variklį į apačią
            if (akselerometras_x > 3.5 && giroskopas_y < -1.0) {
                if (message_command != "g"
                        && ekranas_priliestas == true) {
                    message_command = "g";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                return;
            }
            // keliam variklį į viršų
            if (akselerometras_x < -3.5 && giroskopas_y > 1.0) {
                if (message_command != "h"
                        && ekranas_priliestas == true) {
                    message_command = "h";
                    try {
                        send_command_bluetooth();
                    } catch (IOException ex) {
                    }
                }
                return;
            }
            // leidžiam variklį į apačią
            if (akselerometras_y < -3.5 && giroskopas_x < -1.0) {
                if (message_command != "i"
```

```
                                        && ekranas_priliestas == true) {
                                    message_command = "i";
                                    try {
                                        send_command_bluetooth();
                                    } catch (IOException ex) {
                                    }
                                }
                                return;
                            }
                            // keliam variklį į viršų
                            if (akselerometras_y > 3.5 && giroskopas_x > 1.0) {
                                if (message_command != "j"
                                        && ekranas_priliestas == true) {
                                    message_command = "j";
                                    try {
                                        send_command_bluetooth();
                                    } catch (IOException ex) {
                                    }
                                }
                                return;
                            }
                            // sustabdom robotą
                            if (message_command != "e" && message_command != "f") {
                                if ((akselerometras_x > -2.0 && akselerometras_x < 2.0)
                                        && (akselerometras_y > -2.0 &&
akselerometras_y < 2.0)) {
                                    if (message_command != "s") {
                                        message_command = "s";
                                        try {
                                            send_command_bluetooth();
                                        } catch (IOException ex) {
                                        }
                                    } else {
                                        textView_BT_input
                                                .setText("robotas sustabdytas");
                                    }
                                    return;
                                }
                            }
                        }
                    }
                }
            }

            public void bluetooth_input_listening() {
                runnable = new Runnable() {

                    public void run() {
                        byte[] buffer = new byte[5];
                        int bytes_available;
                        do {
                            try {
                                bytes_available = inputstream.read(buffer);
                                if (bytes_available > 0) {
                                    boolean good_vykdoma_komanda = false;
                                    byte[] command_string_aray = new byte[1];
                                    command_string_aray[0] = buffer[0];
                                    String command_string = new String(
                                            command_string_aray, "UTF8");
                                    final String vykdoma_komanda = command_string;
                                    if (vykdoma_komanda.endsWith("a")) {
```

```
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("b")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("c")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("d")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("e")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("f")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("g")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("h")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("j")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("i")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("m")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("n")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (vykdoma_komanda.endsWith("s")) {
                                        good_vykdoma_komanda = true;
                                    }
                                    if (good_vykdoma_komanda == true) {
                                        handler.post(new Runnable() {

                                            public void run() {

    VykdomosKomandosNustatymas(vykdoma_komanda);
                                            }

                                        });
                                    }
                                }
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        } while (!stop_bluetooth_input);
                    }
                };
            }
        }
```

6 PRIEDAS

SUKURTOS IŠMANIOJO TELEFONO PROGRAMINĖS ĮRANGOS PAGRINDINIŲ NUSTATYMŲ XML KODAS (ANDROIDMANIFEST.XML FAILAS)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="aurimas.vaitekunas.train.robot"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="aurimas.vaitekunas.train.robot.MainActivity"
            android:label="@string/title_activity_main"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

7 PRIEDAS
STRAIPSNIS
ROBOTO MANIPULIATORIAUS NUOTOLINIO VALDYMO SISTEMOS PROGRAMINIS PROTOTIPAS

A. Vaitekūnas, A. Andziulis, K. Gerasimov, J. Vaupšas
KU, Informatikos inžinerijos katedra, Bijūnų 17, LT-91225, Klaipėda, Lietuva

Anotacija
Aktualių informatikos inžinerijos taikomųjų problemų sričiai gali būti priskirtinas nuotolinio valdymo sistemos programinio prototipo sukūrimas ir jo pritaikymas automatizuotai mechatronikos ar robotikos įrenginio pavarai. Šią inovatyvią nuotolinio valdymo sistemą gali sudaryti specializuoto modelio išmanusis telefonas su įrengtais akselerometriniu ir giroskopiniu jutikliais, bevielio ryšio maršrutizatorius, paties įrenginio manipuliatorius ir atitinkamai sudaryta ir adaptuota programinė įranga. Šios nuotolinio valdymo sistemos pritaikymas suteikia naujų galimybių, spendžiant įvairias tikslaus ir greito pozicionavimo užduotis, pavyzdžiui, siekiant efektyviai valdyti transporto priemonių krovos įrenginio manipuliatoriaus judėjimą ir tinkamai užtikrinti krovos darbų saugą.
PAGRINDINIAI ŽODŽIAI: išmanusis telefonas, specializuota programinė įranga, roboto manipuliatoriaus prototipas, Roboto manipuliatoriaus nuotolinio valdymo sistemos prototipas.

Abstract
The weighty domain of modern computer engineering includes the various application-oriented problems of software prototypes of remote control system designed for mechatronical and robot-like automated devices. This innovative system can be composed of specialized smart phone equipped with accelerometer and gyroscopic sensors, wireless router, customized software and the properly manipulator of the device. The application of the aforesaid system provides new opportunities across the wide range of various exact positioning tasks, for example, for precision and fast remote control of transport vehicle loading manipulator and handling operations to ensure safety.
KEY WORDS: smart phone, customized software, robotic manipulator prototype, robotic manipulator remote control system prototype.

Įvadas

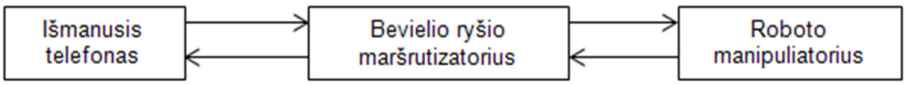
Šiuolaikinių informacinių technologijų vystymasis suteikia daug naujų galimybių įrenginių valdymui. Kai kurių robotizuotų įrenginių manipuliatorių (pavyzdžiui, kranų, ekskavatorių, gamybinių robotų ir kt.) operatorius gali tiesiogiai valdyti iš savo darbo vietos arba šių įrenginių

valdymui gali būti panaudotos įvairios uždarosios grįžtamųjų ryšių automatinio valdymo sistemos su specialiais valdikliais ir jiems pritaikytomis programomis numatytoms užduotims atlikti. Tačiau gali pasitaikyti atvejų, kai sparčiam ir tiksliam manipuliatoriaus valdymui yra tikslinga panaudoti visiškai kitokius valdymo būdus [1-7], ypač žmogaus sveikatai potencialiai pavojingomis sąlygomis, pavyzdžiui, dirbant ant nestabilaus gruntinio pagrindo, atliekant įvairius gelbėjimo ar valymo darbus pavojingose vietose ir kt. [1, 5, 7]. Vienas iš novatoriškų būdų šioms problemoms spręsti yra nuotolinio valdymo galimybių panaudojimas. Patys naujausieji išmaniųjų telefonų modeliai su įmontuotais naujos kartos mikro elektrinės mechaninės sistemos (MEMS) tipo akselerometriniu bei giroskopiniu jutikliais suteikia galimybę robotizuoto įrenginio judėjimą valdyti, keičiant išmaniojo telefono padėtį erdvėje. Šis pažangus technologinis sprendimas galėtų būti panaudotas įvairios paskirties įrenginių robotinių manipuliatorių ir su jais sujungtų darbo įrankių sparčiam ir tiksliam pozicionavimui, manipuliatoriaus judėjimo optimalios trajektorijos užtikrinimui, suteiktų galimybę atsisakyti papildomų valdymo svirčių [3, 5, 7], leistų supaprastinti valdymo sistemos funkcinę struktūrą, padidintų valdymo efektyvumą ir viso įrenginio konstrukciją bei jos patikimumą.

Šiame straipsnyje yra aprašomas roboto manipuliatoriaus modelio nuotolinio valdymo sistemos sukurtasis programinis prototipas, pateikiama jo išplėstoji schema ir šios sistemos veikimo principiniai ypatumai.

Roboto manipuliatoriaus nuotolinio valdymo sistemos duomenų srautai

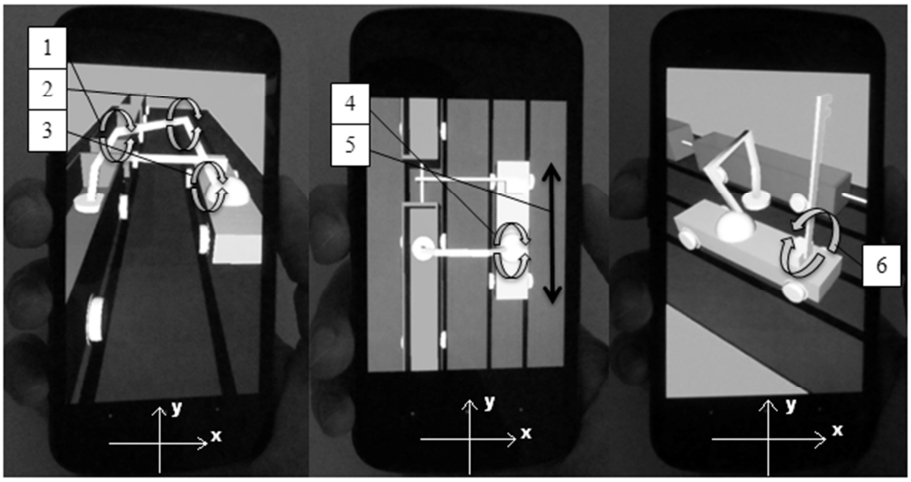
1 pav. yra pateikta roboto manipuliatoriaus nuotolinio valdymo sistemos prototipo schema, susidedanti iš trijų pagrindinių dalių - išmaniojo telefono, bevielio ryšio maršrutizatoriaus ir paties įrenginio - roboto manipuliatoriaus.



1 pav. Roboto manipuliatoriaus nuotolinio valdymo sistemos prototipo duomenų srautų diagrama

Roboto manipuliatoriaus prototipo (RMP) nuotolinio valdymo sistemos kūrimui buvo pasirinktas išmanusis telefonas su „Android 4 Ice Cream Sandwich“ operacine sistema bei įrengtais mikro elektrinės ir mechaninės sistemos akselerometriniu ir giroskopiniu jutikliu. Sudarant nuotolinio valdymo sistemą, pasirinktajam išmaniajam telefonui buvo sukurta specializuotoji programinė įranga, kuri leido šį telefoną pritaikyti manipuliatoriaus prototipo nuotoliniam valdymui (2, 5 pav.), kartu išvengiant papildomos šio telefono optimizavimo techninės įrangos būtinybės. Ši specializuotoji nuotolinio valdymo programinė įranga, panaudodama išmaniajame telefone įrengtų MEMS tipo akselerometrinio bei giroskopinio jutiklių duomenimis, automatiškai valdo virtualųjį

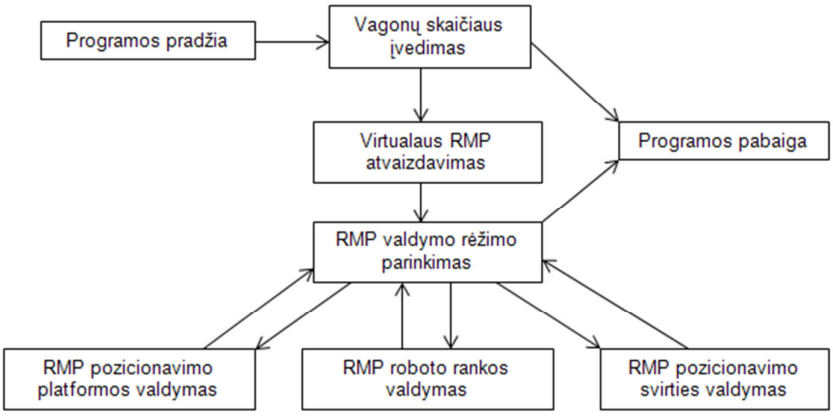
manipuliatoriaus prototipą išmaniojo telefono ekrane (2 pav.). Tuo pačiu metu specializuotoji programinė įranga perduoda valdymo komandas per bevielio ryšio maršrutizatorių į patį roboto manipuliatoriaus prototipą, kuris jau realiu laiku įvykdo gautąsias komandas ir duomenis apie įvykdytą arba neįvykdytą komandą perduoda per bevielio ryšio maršrutizatorių atgal į išmanųjį telefoną.



2 pav. Virtualus roboto manipuliatoriaus prototipo vaizdas išmaniojo telefono ekrane

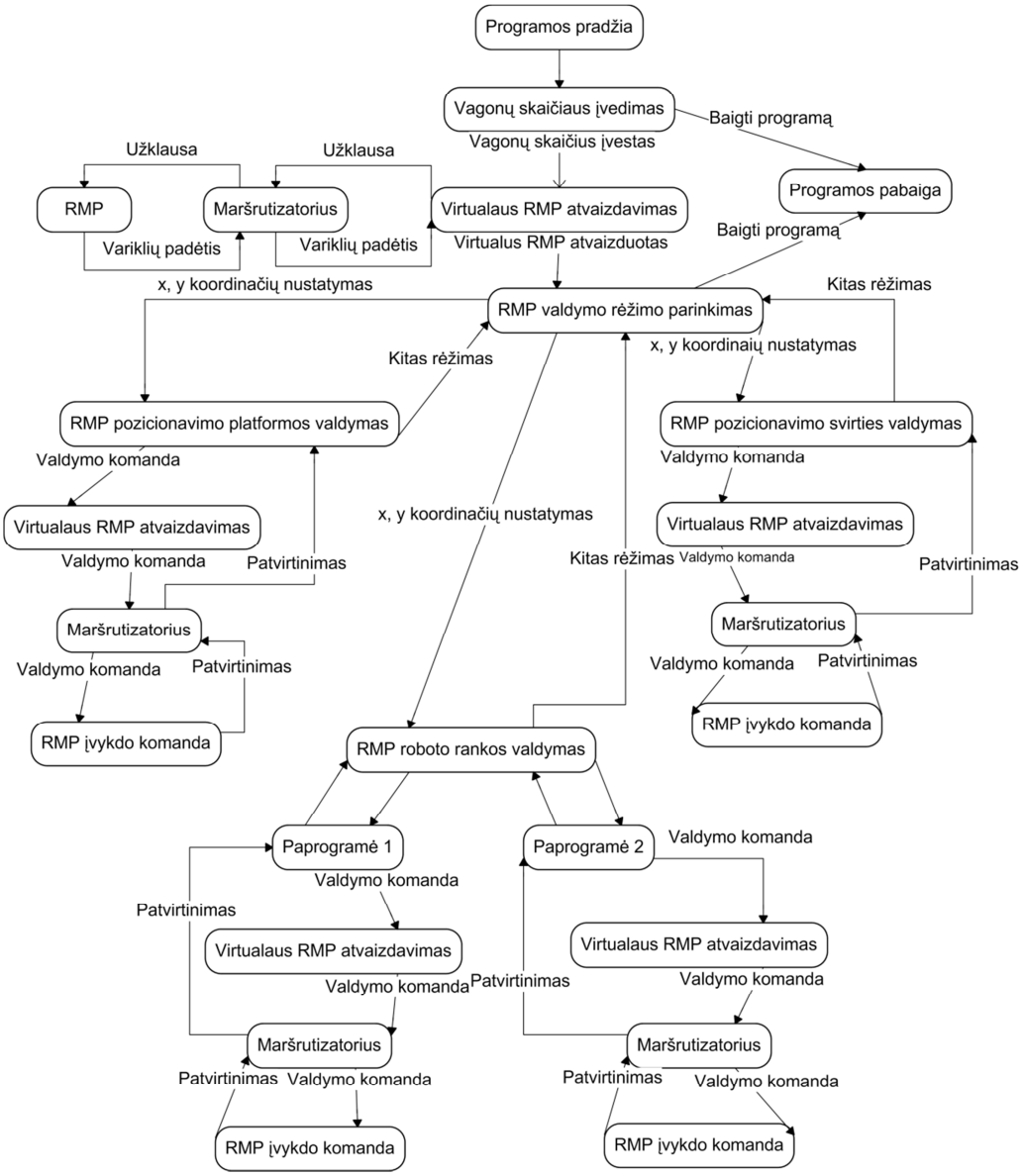
Roboto manipuliatoriaus nuotolinio valdymo sistemos programinis prototipas

Specializuotoji programinė įranga yra sukurta Java programavimo kalba, o roboto manipuliatoriaus prototipą valdo papildoma programa, įdiegta manipuliatoriaus prototipo mikrovaldiklyje, kuris vykdo iš išmaniojo telefono gautąsias valdymo komandas. Ši specializuotoji programinė įranga yra sukurta taip, kad operatorius vienu metu galėtų valdyti tiek virtualųjį roboto manipuliatoriaus prototipą išmaniojo telefono ekrane, tiek ir realųjį paties įrenginio manipuliatorių.



3 pav. Roboto manipuliatoriaus nuotolinio valdymo sistemos programinio prototipo pagrindiniai elementai

Nuotoliniu būdu valdomas roboto manipuliatorius gali būti skirtas įvairiems krovos darbams atlikti, pavyzdžiui, metalo laužo ar medienos rąstų iškrovimo ar pakrovimo į traukinių vagonus ar į kitas transporto priemones. Roboto manipuliatorius susideda iš traukinių pozicionavimo platformos su įmontuota santykinių keturių laisvės laipsnių roboto ranka bei svirtimi (2, 5 pav.). Pozicionavimo platforma gali judėti bėgiais pirmyn ir atgal slenkamuoju judėjimu (2, 5 pav., 5 nr.), o įrenginyje įmontuotuose šarnyriniuose sujungimuose roboto ranka gali pasisukti apie tris ašis (2, 5 pav. 1, 2, 3 nr.) ir pasisukti paties korpuso atžvilgiu (2, 5 pav. 4 nr.), o pozicionavimo svirtis gali judėti sukamuoju judėjimu (2, 5 pav. 6 nr.). Realaus roboto manipuliatoriaus judesiai yra valdomi septynių servo variklių pagalba. Keturi servo varikliai valdo roboto rankos judesius, vienas variklis - pozicionavimo svirtį, o du varikliai - pozicionavimo platformą.



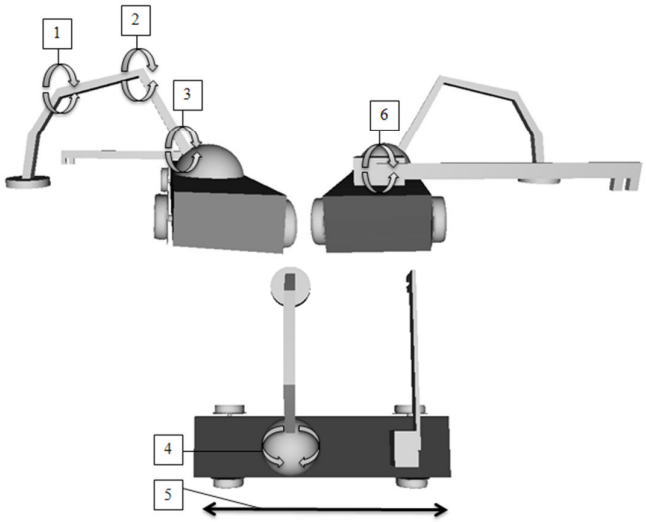
4 pav. Roboto manipuliatoriaus nuotolinio valdymo sistemos programinio prototipo išplėsta schema

Pradėjus veikti nuotolinio valdymo sistemai, išmaniajame telefone įdiegta specializuotoji programinė įranga paprašo įvesti reikiamą vagonų skaičių (3, 4 pav.). Įvedus šią informaciją, programinė įranga iš roboto manipuliatoriaus gauna duomenis apie kiekvieno servo variklio rotoriaus padėtį ir išmaniojo telefono ekrane suformuoja virtualiojo roboto manipuliatoriaus prototipo trimatį vaizdą. Kai šis trimatis vaizdas tampa matomas išmaniojo telefono ekrane, tai specializuotoji programa operatoriui suteikia galimybę pasirinkti vieną iš trijų roboto manipuliatoriaus prototipo valdymo režimų (3, 4 pav.). Veikiant specializuotajai programinėi įrangai, galima laisvai pereiti iš vieno valdymo režimo į kitą. Pasirinkus norimą valdymo režimą programinė įranga automatiškai nustato pradinį koordinačių x ir y atskaitos tašką, pagal kurį yra apskaičiuojamas išmaniojo telefono padėties pasikeitimas x ir y koordinačių plokštumoje.

Įjungus pirmąjį valdymo režimą, yra įjungiamas roboto manipuliatoriaus prototipo pozicionavimo platformos valdymas (3, 4 pav.). Tiek virtuali, tiek ir reali pozicionavimo platforma yra valdoma, sukant išmanųjį telefoną į vieną arba kitą pusę ašies y atžvilgiu (2 pav.), tuo būdu valdant pozicionavimo platformos slenkamąjį judėjimą (2, 5 pav.).

Roboto manipuliatoriaus prototipo pozicionavimo svirties valdymas (3, 4 pav.) yra atliekamas, įjungus antrąjį valdymo režimą. Šiuo atveju, sukant išmanųjį telefoną apie ašį x į vieną arba į kitą pusę (2 pav.), yra valdomas pozicionavimo platformos svirties sukamasis judėjimas (2, 5 pav.).

Įjungus trečiąjį valdymo režimą, yra įjungiamas roboto manipuliatoriaus rankos valdymas (3, 4 pav.). Šiam režimui yra numatytos dvi paprogrames. Įjungus pirmąją paprogramę trečiajame režime yra valdomas manipuliatoriaus roboto rankos sukamasis judėjimas apie patį įrenginio korpusą (2, 5 pav. 4 nr.). Šis judėjimas yra valdomas, sukant išmanųjį telefoną į vieną arba kitą pusę y ašies atžvilgiu. Įjungus antrąją paprogramę trečiajame režime yra valdomi manipuliatoriaus rankos pirmojo, antrojo, trečiojo sujungimų sukamieji judėjimai (2, 5 pav. 1, 2, 3 nr.). Pirmasis, antrasis ir trečiasis sujungimai yra valdomi vienu metu. Trečiojo sujungimo sukamasis judėjimas yra valdomas išmanųjį telefoną sukant į vieną arba kitą pusę apie ašį y. Antrojo sujungimo judėjimas yra valdomas, išmanųjį telefoną sukant į vieną arba kitą pusę ašies x atžvilgiu. Pirmasis sujungimas yra valdomas, liečiant ekraną nykščiu ir braukiant per ekraną į telefono viršų arba apačią. Keičiant programos valdymo režimus realiu laiku yra valdomas tiek virtualusis roboto manipuliatoriaus prototipas išmaniojo telefono ekrane, tiek realusis įrenginio manipuliatorius.



5 pav. Roboto manipuliatoriaus prototipas

Išvados

1. Sistema, kurią sudaro specialiai sukurtoji programinė įranga, maršrutizatorius bei išmanusis telefonas su įrengtais MEMS tipo akselerometriniu ir giroskopiniu jutikliais, gali būti panaudota robotinio manipuliatoriaus prototipo ir realaus įrenginio manipuliatoriaus nuotoliniam valdymui.
2. Nuotolinio valdymo sistemos programinis prototipas gali būti pritaikytas įvairių transporto priemonių pakrovimo ar iškrovimo robotinio įrenginio manipuliatoriaus ir jo darbo įrankio sparčiam ir tiksliam pozicionavimui bei reikalingos judėjimo trajektorijos užtikrinimui.
3. Sukurtasis techninių informacinių sistemų inžinerinės problemos sprendimas sudaro realią galimybę atsisakyti robotinių manipuliatorių tradicinių valdymo svirčių ir tuo būdu supaprastinti viso įrenginio konstrukciją, padidinti jo patikimumą ir valdymo efektyvumą.

Padėka

Autoriai dėkoja projektui LLIV-215 "JRTC Extension in Area of Development of Distributed Real-Time Signal Processing and Control Systems" už paramą atliekant tyrimą.

Literatūra

1. Bernold, L.E., 2007. Control schemes for tele – robotic pipe installation. *Automation in Construction*, 16(4), p. 518-524.
2. Dongmok, K., 2009. Excavator tele-operation system using a human arm. *Automation in Construction*, 18(2), p.173-182.
3. Jungwon, Y., 2010. Development of an intuitive user interface for a hydraulic backhoe. *Automation in Construction*, 19(6), p.779-790.
4. Seo, J.W., 2007. Graphical modeling and simulation for design and control of a tele-operated clinker clearing robot. *Automation in Construction*, 16(1), p. 96-106.
5. Takahiro, S., 2008. Remote control of backhoe at construction site with a pneumatic robot system. *Automation in Construction, 17(8), p.907-914.*
6. Trivino, G., 2009. Towards an architecture for semiautonomous robot telecontrol systems. *Information Sciences*, 179(23), p. 3973-3984.
7. Xinxing, T., 2011. Tele-operation construction robot control system with virtual reality technology. *Procedia Engineering,* 15, p. 1071-1076.