VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Mikrokompiuterių panaudojimas robotizuotai rankai valdyti

(Microcomputer-based Robot Arm Control)

Atliko: 4 kurso, 1 grupės studentas
Rokas Vaičikauskas
Darbo vadovas:
Doc. Dr. Aistis Raudys
Recenzentas:
Asist. Linas Litvinas

Vilnius
2017

Turinys

Įvadas 4
1. Raspberry Pi mikrokompiuteris 6
1.1. Raspberry PI 3 B modelis 6
1.2. GPIO 6
1.2.1. Raspberry Pi 3 B GPIO jungčių grupės 7
1.2.2. GPIO standartininės jungtys 8
1.2.3. Kaip veikia standartinės GPIO jungtys? 8
1.3. Raspberry Pi paruošimas darbui 9
2. Servo varikliai 10
2.1. Kas yra servo variklis 10
2.1.1. Kasdienis servo variklių pritaikymas 10
2.2. Kaip veikia servo variklis? 11
2.2.1. Servo variklio veikimas 11
2.2.2. Kuo ypatingas servo variklio veikimas? 12
2.3. Servo variklių tipai 13
2.4. Servo variklio valdymas 14
2.5. PWM- Impulso pločio moduliacija 14
3. Servo variklio valdymo metodikų tyrimas 16
3.1. Tower Pro MG996R servo varikliukas 16
3.2. Fritzing modeliavimo įrankis 17
3.3. Python ir Rpi.GPIO metodika 17
3.3.1. Impulso pločio moduliacijos modelis 18
3.3.2. Darbinis ciklas 18
3.3.3. Saugumo dirbant su GPIO jungtimis rekomendacija 19
3.3.4. Pavyzdys nr.1 Servo sukinėjimas naudojant darbinį ciklą 19
3.4. WiringPi metodika 21
3.4.1. WiringPI ir gpio įrankis 21
3.4.2. Servo variklio valdymas naudojant WiringPi gpio 22
3.4.3. Pavyzdys nr.2 Java ir WiringPi su „pwm“ rėžimu 23
3.4.4. Pavyzdys nr.3 Java ir WiringPi su „out“ rėžimu 24
3.5. Pi4J – alternatyva WiringPi 26
3.6. GPIO programavimo įrankių analizė 27
4. Robotizuotos rankos valdymas, naudojant Raspberry Pi 28
4.1. Servo varikliukų jungimo schema 28
4.2. Rankos valdymas su Python ir Rpi.GPIO 29
4.2.1. GUI su Python Tkinter 29

4.2.2. Programos procesas 30
4.3. Pastebėjimai valdant robotizuotą ranką 31
Rezultatai ir išvados 32
Summary 33
Literatūros sąrašas 34
Priedas nr.1 35
Priedas nr.2 36
Priedas nr.3 37
Priedas nr.4 38

Įvadas

Kasdien vyksta nesustabdomas technologijų tobulėjimas. Jeigu anksčiau pirmieji kompiuteriai buvo tokie dideli, kad jiems reikėdavo didesnių kambarių, tai dabar jų dydį galima lyginti su banko kortelės dydžiu. Tai yra vadinamieji mikrokompiuteriai. Šis terminas „mikrokompiuteris“ nėra naujas. Jau daugiau nei prieš 40 metų buvo pristatytas vienas pirmųjų komercinių mikrokompiuterių. Tai buvo pagamintas kompanijos MITS (*anlg. Micro Instrumentation Telemetry Systems*) 1974 metais mikrokompiuteris „Altair 8800“.

Šiomis dienomis yra labai paplitęs domėjimasis mikrokompiuteriais. Taip yra dėl to, kad technologijos yra ganėtinai pažengusios ir mikrokompiuterių gamintojai sugebėjo sukurti naujausius ir sparčiausius mikrokompiuterius, kurie tuo pačiu būtų pigūs rinkos gaminiai. Šiomis dienomis mikrokompiuteriai yra naudojami įvairiais tikslais- nuo įprastinio skaičiavimo atlikimo iki edukacinių tikslų. Taip pat vienas iš aktualių mikrokompiuterių panaudojimų yra paplitę visokiausi „pasidaryk pats“- DIY (angl. Do It Yourself) projektai. Šiuose projektuose mikrokompiuteriai yra panaudojami labiau robotikos srityse.

Dabar mikrokompiuterį gali įsigyti beveik kiekvienas. Kadangi mikrokompiuterių gamintojų yra daug tai ir pačių mikrokompiuterių modelių yra su įvairiomis specifikacijomis ir atitinkamomis kainomis. Prieš daugiau nei penkis metus 2012 metais į mikrokompiuterių rinką atėjo gamintojas „Raspberry Pi Foundation“ iš Jungtinės Karalystės, kurių produktai iki šių dienų turi didelį susidomėjimą, kuris nenustoja augti.

Darbo tikslas: panaudoti Raspberry Pi mikrokompiuterį robotizuotai rankai valdyti.

Darbo uždaviniai:

1. Išnagrinėti, kaip veikia Raspberry Pi mikrokompiuterio GPIO jungtys
2. Išanalizuoti, kas yra servo variklis ir kaip jis veikia, kur yra panaudojamas.
3. Ištirti ir išanalizuoti, kokiais būdais gali būti valdomi robotizuotos rankos servo varikliukai. Pasirinkti tinkamiausią būdą ir jį implementuoti visos rankos valdymui.
4. Išnagrinėti, kaip robotizuota ranka bus prijungta su mikrokompiuterio.

Siekiami rezultatai:

1. Pagilinti sukauptas programavimo žinias.
2. Suprasti, kaip yra programuojamos Raspberry Pi mikrokompiuterio GPIO jungtys.
3. Ištirti, kokie yra būdai valdyti servo varikliukus, kurie yra panaudojami robotizuotai rankai valdyti, juos palyginti.
4. Pasirinkti tinkamiausią būdą valdyti servo varikliukams ir implementuoti robotizuotos rankos valdymą, nenaudojant papildomos techninės įrangos.

Darbo eiga sudaryta iš keturių etapų:

1. Pirmame etape yra nagrinėjamas mikrokompiuterio modelis Raspberry Pi 3 B. Taip pat yra nagrinėjamos GPIO jungtys.
2. Antrame etape yra bandoma išsiaiškinti kas yra servo variklis, kokia jo paskirtis, kaip jis veikia. Bus panagrinėtas naudojamų servo varikliukų modelis Tower Pro MG996R.
3. Trečią etapą sudaro įrankių ir būdų valdyti servo varikliuką tyrimas bei pavyzdžių pateikimas išbandant analizuotus metodus ir patogaus vieno servo varikliuko valdymo metodo pasirinkimas visai rankai valdyti.
4. Ketvirtas etapas - robotizuotos rankos valdymas panaudojant mikrokompiuterį, valdymo metodo apžvalga.

1. Raspberry Pi mikrokompiuteris

1.1. Raspberry PI 3 B modelis

Naujausias Raspberry Pi 3 B modelio mikrokompiuteris buvo išleistas 2016 metų vasario mėnesį. Tai yra trečios kartos Raspberry Pi modelis, kuris pakeitė prieš tai buvusį Raspberry Pi 2 B modelį. Kaip ir ankstesnis, prieš tai minėtas modelis Raspberry pi 3 B turi tas pačias specifikacijas:

- 1GB RAM operatyviosios atminties
- 4x USB lizdai
- 40 GPIO jungčių
- Full HDMI lizdą
- Ethernet lizdą
- 3.5mm kombinuotą garso lizdą, kuris kartu gali apdoroti ir sudėtinius vaizdo signalus (angl. *composite video signals*)
- Kameros sąsają CSI (angl. *Camera Serial Interface*)
- Displėjaus sąsaja DSI (angl. *Display Serial Inderface*)
- Micro SD atmintės kortelės vieta.
- VideoCore IV 3D grafikos branduolys

Naujausi Raspberry Pi 3 B modelio ypatumai:

- 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)

1.2. GPIO

Vienas iš efektyviausių funkcionalumų, kuriuos turi Raspberry Pi mikrokompiuteriai- eilė jungčių dar žinomų kaip GPIO (angl. *General Purpose Input/Output*). Šias įvesties/išvesties jungtis galima rasti išsidėsčiusias ant mikrokompiuterio krašto, priešingo, nei kur yra pajungiamas maitinimo šaltinis.

Šios jungtys yra fizinė sąsaja tarp mikrokompiuterio ir išorinio pasaulio. Paprasčiausiame lygmenyje, apie juo galima galvoti kaip apie jungiklius, kuriuos mums galima įjungti arba išjungti (įvestis) arba, pats Raspberry Pi mikrokompiuteris gali juos įjungti arba išjungti (išvestis).

1.2.1. Raspberry Pi 3 B GPIO jungčių grupės

Raspberry Pi 3 B modelis turi 40 GPIO jungčių, kurios yra suskirstytos į tokias grupes:

1. **Galios** (angl. *Power*)- šios jungtys skirtos tiesiogiai gauti energijos/galios iš Raspberry Pi 3. Yra dviejų tipų energijos jungtys: dvi 3.3V įtampos ir dvi 5.0V įtampos.
2. **Įžeminimo** (angl. *Ground*) jungtys yra naudojamas naudojamų prietaisų įžeminimui. Kadangi jie yra sujungti į tą pačią liniją, tai galima naudoti bet kurį iš jų. Įžeminimui yra skirtos aštuonios jungtys.
3. **UART** (angl. *Universal Asynchronous Receiver / Transmitter*) – jungtys skirtos naudoti komunikacijai su kitais įrenginiais.
4. **I^2C** (angl. *Inter-Integrated Circuit*) – jungtys, kurios suteikia galimybę prisijungti ir komunikuoti su techninės įrangos moduliais, kurie palaiko šį komunikavimo **I^2C** protokolą.
5. **SPI** (angl. *Serial Peripheral Interface Bus*) – jungtys, kurios suteikia galimybę prisijungti ir komunikuoti su technine įranga, kuri palaiko **SPI** komunikavimo protokolą.
6. **GPIO** (General Purpose Input / Output) – standartinės jungtys, kurios gali būti naudojamos įrenginiams įjungti ir išjungti.

Toliau šiame darbe bus naudojami tiktais šių tipų jungtys: galios, įžeminimo ir GPIO.

1.2.2. GPIO standartininės jungtys

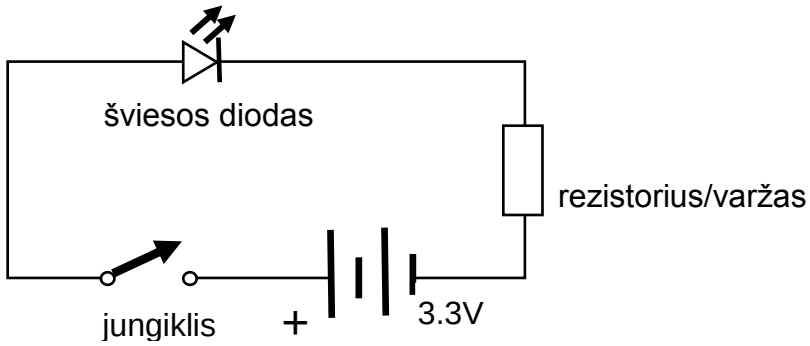
Kam jos naudojamos? Ką su jomis galima padaryti?

Šias jungtis galima užprogramuoti taip, kad jos sąveikautų įvairiais būdais su realiu pasauliu. Įvestis neprivalo atkeliauti iš fizinio jungiklio- tai gali būti įvestis iš įvairių jutiklių arba signalas iš kito kompiuterio ar įrenginio. Išvestis gali taip atlikti įvairius dalykus- nuo šviesos diodo (angl. *LED; Light – emitting diode*) uždegimo iki signalo arba duomenų perdavimo kitiems įrenginiams. Jei Raspberry Pi mikrokompiuteris yra pajungtas į tinklą, tai galima su juo valdyti įrenginius, kurie taip pat yra pajungti į tą patį tinklą, įrenginiai gali siųsti duomenis atgal į mikrokompiuterį.

1.2.3. Kaip veikia standartinės GPIO jungtys?

1.2.3.1. Išvestis

Kalbant apie išvestis, akimirkai reikėtų primiršti apie Pi mikrokompiuterį ir panagrinėti vieną paprasčiausių elektros grandinių. Tokiai grandinei sukurti užtektų: šviesos diodo, baterijos, jungiklio ir rezistoriaus/varžo.



Kai yra naudojamas GPIO jungtis, Raspberry Pi pakeičia abu- jungiklį ir bateriją, pažymėtus viršuje esančioje diagramoje. Kiekviena jungtis gali būti vienoje iš dviejų būsenų: įjungta arba išjungta. Kai jungtis yra įjungta, kai kur naudojamas terminas „HIGH“, ji išveda 3.3 voltų įtampą (3v3), o būsena „LOW“ reiškia- išjungta. Taigi, naudojant Raspberry Pi, būtų naudojamos dvi jungtys: GPIO jungtis, kuri išveda +3.3V įtampą ir įžeminimo jungtis, kuri yra 0V įtampos ir atstoja neigiamą baterijos galą.

1.2.3.2. Įvestis

Kaip prieš tai viršuje nagrinėta- GPIO jungtis naudoti išvesties tikslais yra gana paprasta ir lengva, tačiau to paties nebūtų galima pasakyti apie įvestį. Kai GPIO jungtis yra nustatyta ir naudojama įvesties tikslais, jungtis yra „plaukiojanti“ ir neturi apibrėžtos įtampos. Norint patikimai galėti aptikti ir nustatyti ar įvestis yra (angl. *high*) arba jos nėra (angl. *low*), reikia ją susieti taip, kad ji pastoviai būtų prijungta ir būtų skaitomos *high* arba *low* būsenos.

Tam, kad būtų jungtis susieta, reikia prijungti taip vadinamus *pull up* arba *pull down* rezistorius. *Pull down* rezistorius sujungia jungtį su įžeminimu per didesnį rezistorių, tai reiškia, kad kai jungiklis yra atkeltas – maršrutas eina į įžeminimą ir taip bus nuskaitoma *low* reikšmė. Kai jungiklis yra paspaudžiamas (kita pusė yra prijungta prie tarkim 3.3V įtampos jungties), tada atsiranda mažesnis pasipriešinimo kelias į *high* ir taip jungtis nuskaitys – *high.*

1.3. Raspberry Pi paruošimas darbui

Vien turėti įsigijus Raspberry Pi mikrokompiuterį nepakanka. Jam reikia turėti maitinimo šaltinį ir paruoštą microSD atminties kortelę, kurios vyksta įkrovimas (angl. *boot*) su Linux dažniausiai naudojama „Raspbian“ operacine sistema. Dažniausiai gamintojai siūlo įsigyti mikrokompiuterius su komplektais, į kuriuos įeina paruoštos SD kortelės ir t.t. Tačiau perkant komplektą keičiasi ir kaina. Paprasčiausiai, kai kuriais atvejais mikrokompiuterio maitinimo šaltiniui yra naudojamas telefono pakrovėjas vietoj jam skirto specialaus pakrovėjo.

Kadangi šiam darbui buvo naudotas asmeninis Raspberry PI 3 B mikrokompiuteris, reikėjo dar apgalvoti, kaip bus prie mikrokompiuterio prisijungta darbui. Idealiu atveju, geriausia tam būtų turėti atskirą monitorių, klaviatūrą ir pelę, kas reiškia papildomas išlaidas neturint atliekamų šių komponentų. Tad šiuo atveju buvo pasirinktas kitas būdas- prisijungimas prie mikrokompiuterio naudojant nuotolinį valdymo būdą (angl. *remote desktop*) per maršrutizatorių.

2. Servo varikliai

2.1. Kas yra servo variklis

Servo varikliai ir varikliukai yra savarankiški (angl. *self-contained*) elektriniai prietaisai, kurie pasuka arba pastumia mechanizmų dalis su dideliu ir puikiu tikslumu. Šie servo varikliai yra naudojami daugelyje įvairių vietų- nuo žaislų iki namų elektronikos, iki mašinų ir lėktuvų. Jeigu turėtume radio bangomis valdomus modelius, tokius kaip mašina, lėktuvas ar sraigtasparnis- jau tada būtų naudojami bent jau keli servo varikliukai. Mašinos ar orlaivio modelyje įtaisyti servo varikliukai judina svirtis pirmyn ir atgal, tam, kad būtų kontroliuojamas vairavimas arba būtų reguliuojamas sparnų padėtis. Sukant servo varikliuko veleną, prijungtą prie variklio droselio (angl. *engine throttle*), galima reguliuoti kuru varomos mašinos ar orlaivio greitį.

2.1.1. Kasdienis servo variklių pritaikymas

Kasdieniame gyvenime servo varikliukai yra naudojami prietaisuose, tik mes to nepastebime. Elektroniniai prietaisai, tokie kaip DVD ir Blu-ray DiscTM diskų grotuvai taip pat naudoja servo varikliukus, tam kad galėtų ištiesti ir atgal įtraukti disko dėklą. Dabar XXI amžiaus automobiliuose, servo varikliai valdo mašinos greitį tokiu principu:

1. Panašiai kaip radio garso reguliavimo valdiklis, greičio pedalas siunčia elektroninius signalus, kurie pasako mašinos kompiuteriui kiek giliai pedalas yra paspaustas.
2. Mašinos kompiuteris apdoroja gautą informaciją (taip pat pasitelkęs duomenis iš kitų jutiklių) ir apskaičiuotą rezultatą siunčia signalą servo varikliui, kuris yra pritaisytas prie droselio, kad būtų sureguliuotas mašinos variklio greitis.

Na, o pavyzdžiui, visokie komerciniai lėktuvai naudoja servo variklius kartu su hidraulinėmis technologijomis tam, kad būtų galima traukti ir stumti praktiškai beveik viską lėktuve.

Na ir žinoma, robotai neegzistuotų be servo variklių. Praktiškai galima atkreipti dėmesį į tai, kad beveik kiekviename filme naudojami robotai, sudėtingos lėlės (angl. *animatronic puppets*) gali naudoti daugybę servo varikliukų. Netgi paprasčiausiose parduotuvėse parduodami robotizuoti gyvūnų modelio žaislai naudoja šiuos varikliukus. Mažesni laboratorijos paskirties robotai irgi naudoja servo varikliukus, kad galėtų judinti kardanus, sąnarius ir visokius kitus sujungimus. Pomėgiui skirti servo varikliukai būna įvairiu formų ir dydžių skirtingiems pritaikymams. Pavyzdžiui, didesnio ir galingesnio servo varikliuko reikėtų didesnei roboto rankos judesio valdymui, o kokiam nors roboto galvos antakio valdymui (pakelti aukštyn ir nuleisti žemyn) užtektų mažesnio.

2.2. Kaip veikia servo variklis?

Šiame skyrelyje bus apžvelgta, kaip veikia servo varikliukai ir kodėl jų naudojimas yra gana efektyvus ir patogus.

2.2.1. Servo variklio veikimas

Servo variklių paprastumas yra viena iš ypatybių, todėl jie yra tokie patikimi. Servo variklio „širdį“ sudaro mažas pastovios srovės (angl. *direct current* –DC) motoriukas. Šie motoriukai yra maitinami elektros šaltinio (pavyzdžiui baterijos) ir turi didelį apsisukimų per minutę RPM (angl. *rotations per minute*) skaičių, bet transliuoja ganėtinai mažą sukimo momentą. Sukimo momentą galima įsivaizduoti kaip, pavyzdžiui stiklainio dangtelio atidarymo darbui reikalingą sukimo jėgą. Mechanizmo išsidėstymas priima didelį variklio greitį ir jį sulėtina, tuo pačiu momentu padidindamas sukimo momentą. Pagal vieną iš pagrindinių fizikos taisyklių- darbas[1] (elektriniame lauke), yra lygus jėgos ir atstumo sandaugai- mažas elektrinis motoriukas neturi pakankamai sukimo momento, tačiau jis gali suktis kur kas greičiau (turės mažesnę jėga, didesnį atstumą).

Servo variklio korpuse esantis mechanizmo dizainas konvertuoja išvestį į kur kas lėtesnį posūkio greitį, tačiau su didesniu sukimo momentu (didesnė jėga, mažesnis

[1] $A = F\Delta_d$, kur A – darbas, F- jėga, o Δ_d- atstumas. Nuoroda: http://www.mat.lt/fizikos-formules/elektrostatika.html

atstumas). Antru atveju, atlikto darbo kiekis yra toks pat, tiesiog daugiau suteikia naudos. Pigesniuose servo varikliukuose esantis mechanizmas yra paprastai pagamintas iš plastmasės, kad motoriukas būtų išlaikytas lengvesnis ir pigesnis. Sunkesniam darbui atlikti reikia galingesnio sukimo momento, todėl tokio servo varikliuko mechanizmas bus pagamintas iš metalo ir bus sunkiau pažeidžiamas.

Naudojant mažesnį pastovios srovės motoriuką, reikia jam pritaikyti galios iš maitinimo šaltinio (pavyzdžiui baterijos), tam kad jis galėtų suktis. Skirtingai nei paprasto pastovios srovės motoro, servo variklio sukamas motoro velenas yra prilėtinamas su mechanizmu. Ant galutinio dantiračio esantis pozicijos jutiklis yra prijungtas prie mažos schemos plokštės. Jutiklis nurodo plokštei kiek toli yra servo variklio išvesties velenas pasukamas. Iš kompiuterio ar radijo bangomis- nuotoliniu valdymo prietaiso būdu yra paduodamas įvesties signalas į tą pačią plokštę. Plokštėje esanti elektronika dekoduoja signalą ir nustato kiek toli naudotojas nori pasukti veleną. Tada įvyksta patikrinimas tarp norimos pozicijos ir esamos pozicijos ir taip yra nusprendžiama kuria kryptimi reikia pasukti veleną iki norimos pozicijos.

2.2.2. Kuo ypatingas servo variklio veikimas?

Tarkime turime du objektus A ir B. Objektas A nori siųsdamas signalus objektui B nori pranešti, kokiu atstumu objektui B reikia pasislinkti. Jeigu objektas B pasislinko per daug, objektas A tada siunčia signalą B, kad jis pasislinktų atgal arčiau objekto A, į reikiamą vietą. Šiuo atveju, jei objektas B būtų paprastas motoras robotizuotoje rankoje, o objektas A- mikrokompiuteris, tai objektas A turėtų sugaišti daugiau laiko stebėdamas, ką B padarė ir duodamas toliau komandas grąžinti į reikiamą vietą (tai vadinama grįžtamuoju ryšiu). Tačiau, jei B būtų servo motoras, A užtektų nurodyti tiksliai į kurią vietą B turi nukeliauti ir nebereikėtų stebėti ar B nepajudėjo toliau nei reikia. Šitoks gana tikslus pozicijos nurodymas yra tai, kodėl servo varikliukai yra tokie naudingi ir plačiai naudojami robotikos taikymuose.

2.3. Servo variklių tipai

Servo varikliai būna netik skirtingų dydžių, bet ir tipų, kurių yra trys: pozicinio sukimosi, nuolatinio sukimosi ir linijinis.

- Pozicinio sukimosi servo variklis: tai labiausiai ir dažniausiai sutinkamas servo motorų tipas. Išvesties velenas gali pasisukti apie 180 laipsnių, kas būtų pusė apskritimo. Šių servo variklių mechanizmuose yra įtaisyti fiziniai stabdikliai, kurie neleidžia pasisukimo daugiau nei yra numatyta pagal servo varikliuko specifikaciją. Šiuo atveju servo varikliukas daugiau nei 180 laipsnių pasisukti negalėtų. Stabdikliai skirti tam, kad būtų apsaugotas pasisukimo jutiklis. Šie dažniausiai naudojami servo varikliai yra pritaikomi radijo bangomis valdomoms mašinoms, laivams ir orlaiviams, žaislams, robotams ir t.t.

- Nuolatinio sukimosi servo variklis: šito tipo varikliai yra panašūs į dažnai naudojamus pozicinio pasisukimo servo motoriukus. Vienas pagrindinių skirtumų tas, kad gali suktis bet kuria kryptimi neribotą laiką. Valdymo signalas, skirtinai negu statinės pozicijos nustatymas, yra interpretuojamas kaip sukimosi kryptis ir sukimosi greitis. Galimų komandų pasirinkimo diapazonas priverčia servo variklį suktis prieš arba pagal laikrodžio rodyklę pasirinktinai, kartu ir nustatant įvairų sukimosi greitį, priklausomai nuo komandos. Šiuos servo varikliukus, pavyzdžiui, būtų galima pritaikyti radaro lėkštei valdyti.

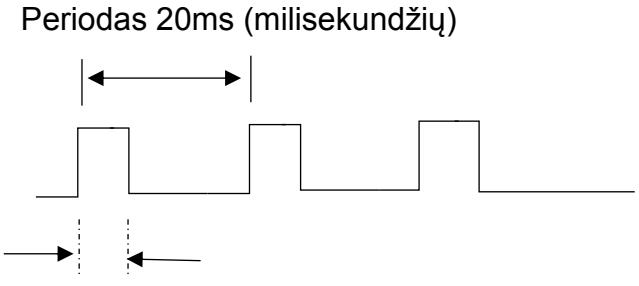
- Linijinis servo variklis: taip pat panašus į prieš tai viršuje aprašytą pozicinio pasisukimo variklį. Skirtumas tas, kad šie turi papildomų mechanizmų, dažniausiai, tai būna lentynos kartu su krumpliaračio mechanizmas. Tai leidžia pakeisti išvestį nuo apskritos iki tiesinės (pirmyn-atgal). Šie servo varikliai gali būti panaudojami kaip pavaros didesnio modelio lėktuvuose

2.4. Servo variklio valdymas

Servo varikliai priima komandas iš impulsų sekų, kurias siunčia kompiuteris arba radijas. Impulsas tai perėjimas, iš žemesnės įtampos į aukštesnę įtampą, kuris yra išlaikomas trumpam laiko momentui, ir tada vėl sugrąžinamas prie žemos įtampos. Tokiems prietaisams kaip servo varikliukams, kuriems pakanka paprastos baterijos kaip maitinimo šaltinio, žema įtampa laikoma įžeminimas arba 0 voltų. Aukšta įtampa tada būtų baterijos įtampa. Paprastai servo varikliukai dirba nuo 4.5 iki 6 voltų įtampos diapazone, todėl jie skaitomi, kad yra ypač mėgėjiški- skirti pramogai.

2.5. PWM- Impulso pločio moduliacija

Kaip minėta, servo varikliai yra valdomi siunčiant kintamo pločio elektrinius impulsus, naudojant impulso pločio moduliaciją- PWM (angl. *Pulse Width Modulation*). Yra minimalus ir maksimalus impulsai, bei pasikartojimo dažnis. Servo motoriukas paprastai gali pasisukti 90 laipsnių į vieną ir kitą kryptį, bendroje sumoje – 180 laipsnių. Neutrali motoriuko pozicija yra apibrėžta kaip pozicija, kur servo varikliukas turi tokio pat dydžio potencialaus pasisukimus pagal laikrodžio rodyklę ir prieš. Todėl svarbu atkreipti dėmesį į tai, kad kiekvieno modelio servo varikliukai gali turėti skirtingus apribojimus- kiek laipsnių leidžiama pasisukti velenui, bet visų jų neutrali (centrinė padėtis) bus visada maždaug apie 1,5 milisekundžių trukmės impulso signale. Žemiau pateikta iliustracija, kurioje pavaizduotas impulso signalas.

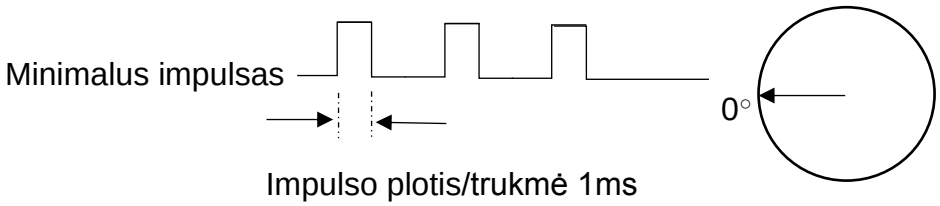


Pagal viršuje paminėtą impulso pločio moduliaciją, pasisukimo kampas yra nustatomas pagal impulso, atėjusio per valdymo laidą, trukmę. Servo variklis tikis gauti impulsą kas 20 milisekundžių. Taigi, impulso trukmė nustato kiek toli motoriukas pasisuks, todėl pavyzdžiui 1,5ms impulso signalas privers motoriuką pasisukti į 90 laipsnių poziciją, kuri kaip ir viršuje buvo minėta yra laikoma neutralia pozicija.

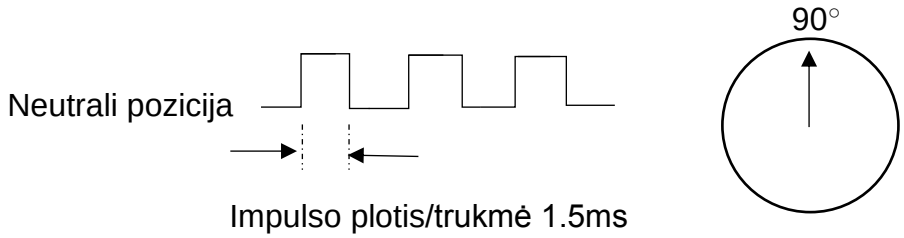
Kai servo varikliukams yra duotos komandos pasisukti, jie pasisuks į reikiamą poziciją ir ją laikys. Jei didesne išorine jėga bus bandoma pasukti servo motoriuką, varikliukas priešinsis ir nesileis būti pajudinamas iš pozicijos. Sukimo momento įvertinimas ir našumas nusako kiek maksimaliai jėgos servo variklis gali atlaikyti. Taip pat servo varikliukas amžinai pozicijos neišlaikys, jam pastoviai reikia siųsti impulso signalą, kad jis gautų nurodymus išlikti esamoje pozicijoje.

Kai nusiųsto impulso trukmė yra mažiau nei 1,5ms, tada servo varikliukas pasisuka išvesties veleną prieš laikrodžio rodyklę nuo neutralios pozicijos per tam tikrą laipsnių skaičių, ir tada lieka išlaikydamas naują poziciją. Kai impulso trukmė yra didesnė nei 1,5ms, tada įvyksta veleno pasisukimas pagal laikrodžio rodyklę. Kiekvienas atskiras servo variklio modelis gali turėti savo atskirą minimumą ir maksimumą. Dažniausiai 1ms trukmės impulso signalas yra minimumas, o 2ms – maksimumas.

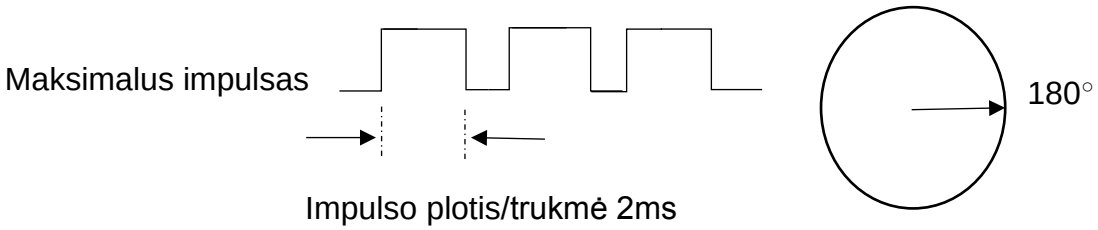
Žemiau pateikta iliustracija, kurioje galima matyti pavaizduotas servo varikliuko pozicijas, pagal impulso trukmę



Impulso plotis/trukmė 1ms



Impulso plotis/trukmė 1.5ms



Kitas parametras, kuris skiriasi tarp servo variklių, tai posūkio sukimosi greitis. Tai laikas, kuris yra sugaištamas servo varikliukui pakeisti iš vienos pozicijos į kitą. Blogiausiu atveju sukimosi laikas yra tada, kai servo variklis laikosi minimumo pozicijoje ir yra priverstas komandų persisukti į maksimumo poziciją. Tai gali užtrukti iki kelių sekundžių jei tai yra didelio sukimo momento servo varikliukas.

3. Servo variklio valdymo metodikų tyrimas

3.1. Tower Pro MG996R servo varikliukas

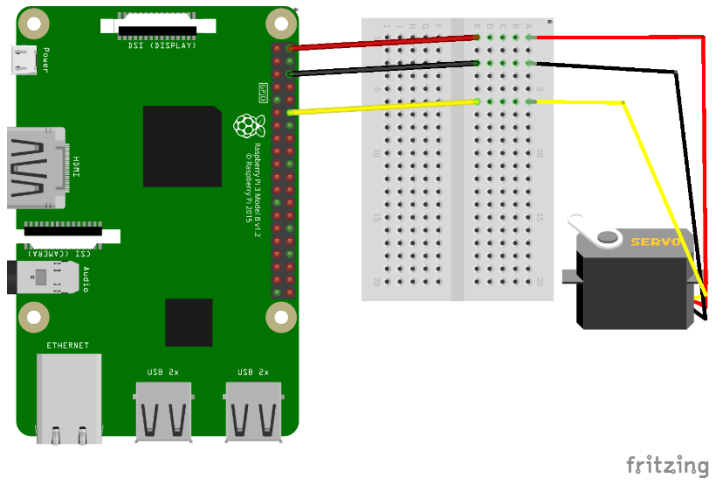
Šiame darbe, robotizuotai rankai valdyti bus naudojami Tower Pro MG998R modelio servo varikliukai. Šie servo varikliukai turi iš metalo pagamintą mechanizmą, todėl tuo pačiu turi ir didelį sukimo momentą, kurio gali pakakti pakelti 10kg objektui. Taip pat šio modeliu varikliukai gali pasisukti per 180 laipsnių (po 90 į abi puses nuo neutralios padėties. Jam valdyti galima naudoti bet kokį servo kodą, techninę įrangą ar biblioteką, todėl šis varikliukas yra labai tinkamas pradedantiesiems, kurie nori kažką sukonstruoti nenaudodami motorų valdymo pultų (angl. *controller*) su grįžtamuoju ryšiu ir pavarų dėžėmis.

Iš servo varikliuko išeina 30cm ilgio trijų rūšių laidai: rudas, raudonas ir oranžinis. Rudas yra jungiamas prie įžeminimo (-) jungties, raudonas skirtas jungti prie maitinimo šaltinio (+), o oranžinis skirtas prijungti prie jungties, iš kurios ateis impulso signalai, šiuo atveju tai būtų Raspberry Pi GPIO jungtis (išvesties fazėje).

Toliau šiame darbe bus bandoma ištirti ir apžvelgti, kokiais būdais ir įrankiais galima pasinaudoti, kad būtų galima valdyti servo varikliuką. Kadangi pradžioje bus dirbama su vienu servo varikliuku, nebus nustatytos tikslios jungtys, prie kurių bus konkrečiai prijungti servo varikliuko minėti trijų paskirčių laidai. Kol dirbama su vienu servu, raudonas (+) laidas skirtas maitinimo šaltiniui, bus prijungtas prie vienos iš dviejų jungčių, kurios generuoja 5V. Rudas (-) laidas, skirtas įžeminimui bus taip pat prijungtas prie bet kurios įžeminimo GND jungties. Na o signalui skirtas oranžinis laidas bus prijungtas prie, bet kurios GPIO jungties ir ta jungtis bus naudojama kaip Raspberry Pi mikrokompiuterio signalo išvesties jungtis.

3.2. Fritzing modeliavimo įrankis

Darbo eigoje buvo rastas patogus įrankis, skirtas modeliuoti projektus – fritzing. Kolkas įrankis yra dar beta versijoje, taip pat yra kartu ir nemokamas. Toliau pateikta diagrama, kuri palengvins vieno servo varikliuko prijungimo prie mikrokompiuterio atvaizdavimą.



Vieno servo varikliuko prijungimo prie Raspberry Pi 3 atvaizdavimas naudojant fritzing įrankį

3.3. Python ir Rpi.GPIO metodika

Anksčiau prie Raspbian operacinės sistemos reikėdavo parsisiųsti ir įsirašyti RPi biblioteką norint dirbti su GPIO jungtimis, jeigu tikslas yra viską daryti su Python programavimo kalba. Dabar naujesnėse Raspbian versijose to daryti nebereikia, ši minėta biblioteka atkeliauja su pačia Raspbian operacine sistema.

3.3.1. Impulso pločio moduliacijos modelis

Tam, kad galėtume pasukinėti servo varikliuką su Raspberry Pi, reikės panaudoti vieną iš RPi.GPIO modulį – impulso pločio moduliaciją. Tai pirmame žingsnyje reikia sukurti minėtos moduliacijos egzempliorių, kuris būtų susietas su kuria nors viena iš GPIO jungtimi. Tam yra komanda:

servo_1 = GPIO.PWM(18, 50) (1)

Kaip matoma, norint panaudoti viršuje aprašytą GPIO.PWM(18, 50) metodą, reikia jam paduoti du parametrus. Pirmas parametras (18) yra GPIO jungties numeris, o antras parametras yra dažnis ir atitinka 50Hz. Dažnis buvo pasirinktas toks todėl, kad servo varikliukas visada tikisi gauti impulsą kas 20 milisekundžių, tai reiškia:

$\frac{1000}{20} = 50Hz$ (arba 50 impulsų per vieną sekundę)

3.3.2. Darbinis ciklas

Toliau, kai esame apsirašę impulso ploto moduliaciją ir ją susieję su konkrečia jungtimi, antras žingsnis būtų pradėti siųsti servo varikliui impulsą su komanda:

servo_1.start(x) (2)

Viršuje esanti parametras x yra vadinamas- *darbiniu ciklu*. Darbinis ciklas apibrėžia, proporcingumą tarp impulso signalo ilgio ir periodo. Tad norint impulso atitinkančio tam tikrą ilgį, jį galima apskaičiuoti:

$$darbinis\ ciklas\ = \frac{norimo\ impulso\ ilgis}{periodo}$$

Kadangi servo varikliai naudoja 20 milisekundžių ciklus, galima pasiskaičiuoti pradžiai servo varikliuko trijų paprasčiausių ir pagrindinių (standartinių) pasukimų darbinius ciklus:

1. 0° laipsnių pozicijos $darbinis\ ciklas = \frac{0.5}{20} \times 100 = 2.5\%$

2. 90° laipsnių pozicijos (neutrali) $darbinis\ ciklas = \frac{1.5}{20} \times 100 = 7.5\%$

3. 180° laipsnių pozicijos $darbinis\ ciklas = \frac{2.5}{20} \times 100 = 12.5\%$

Žinant norimą darbinio ciklo reikšmę galima pereiti prie trečiojo žingsnio. Tai būtų servo varikliuko darbinio ciklo pakeitimas naudojantis šia komanda:

servo_1.start(darbinis ciklas) (3)

Na ir paskutinis žingsnis būtų – impulsų, siunčiamų į servo varikliuką sustabdymas:

servo_1.stop() (4)

3.3.3. Saugumo dirbant su GPIO jungtimis rekomendacija

Svarbu:

Pabaigus darbą su GPIO jungtimis, yra rekomenduojama prieš nutraukiant programos vykdymą, išvalyti naudotų jungčių nustatytas paskirtis. Tai galima padaryti su RPi.GPIO bibliotekos funkcija GPIO.cleanup(). Kodėl tai yra naudinga? Ši funkcija padės išvengti žalos. Tarkime susidaro situacija, kada jungtis yra įjungtoje (angl. *HIGH*) būsenoje kaip išvestis, ir netyčia yra prijungiama prie įžeminimo GND (būsena LOW). Tai gali sukelti trumpą jungimą jungtyje ir ji gali iškepti.

Pastabos:

1. GPIO.cleanup() komanda tik išvalys tų jungčių būsenas (atlaisvins), kurios yra panaudotos toje pačioje programoje, kaip ir pati funkcija.
2. Ji atlaisvins programoje panaudotas jungtis ir suteiks kiekvienai jų įvesties būseną.
3. Taisyklingas funkcijos naudojimas yra programos kodo pabaigoje, o ne pradžioje.

3.3.4. Pavyzdys nr.1 Servo sukinėjimas naudojant darbinį ciklą

Šiame pavyzdyje pavaizduosime paprastą servo varikliuko valdymą naudojantis while ciklu bei prieš tai minėtomis RPi.GPIO komandomis.

```
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setmode(GPIO.BCM) # BCM/BOARD
5 GPIO.setup(18, GPIO.OUT)
6 servo_1 = GPIO.PWM(18, 50)
7 servo_1.start(7.5) #neutrali pozicija
8 time.sleep(3)
9
10 try:
11        while True:
12                  servo_1.ChangeDutyCycle(7.5)
13                  time.sleep(2)
14                  servo_1.ChangeDutyCycle(2.5)
15                  time.sleep(2)
16                  servo_1.ChangeDutyCycle(7.5)
17                  time.sleep(2)
18                  servo_1.ChangeDutyCycle(12.5)
19                  time.sleep(2)
20 except KeyboardInterrupt:
21      servo_1.stop()
22        GPIO.cleanup()
```

Reikia atkreipti dėmesį į 4-tą kodo eilutę. Komanda GPIO.setmode() nurodo kokia bus naudojama GPIO jungčių numeracija.

Yra du būdai, kaip galima naudoti Raspberry Pi mikrokompiuterio įvesties ir išvesties jungčių numeracijų išsidėstymą su RPi.GPIO:

1. Pirmas būdas yra naudoti BOARD numeravimo sistemą. Tai nurodo paprastą jungčių numeravimą einant iš eilės nuo pirmos, viršuje ir kairėje esančios 3.3V įtampos jungties. Privalumas yra tas, kad naudojant tokią numeracijos sistemą techninė įranga visada veiks nepriklausomai nuo RaspberryPi jungčių lentos versijos (Priedas nr1. išorinė diagramos numeracija). Vadinasi, nereikės perjunginėti laidų prie kitų jungčių ar keisti programos kodo.

2. Antras būdas yra BCM numeravimas. Tai yra žemesnio lygio būdas dirbti, kadangi čia naudojamas Broadcom[2] SoC (system-on-a-chip) kanalų numeravimas. Todėl reikia visada dirbti su diagrama, norint žinoti kurie kanalų numeriai atitinka kurią jungtį ant Raspberry Pi lentos (Priedas nr.1

[2] Broadcom- įmonė gamintojas, dėl to turi savo atskirą numeraciją. Nuoroda: https://en.wikipedia.org/wiki/Broadcom

vidinė diagramos numeracija) . Trūkumas tas, kad gali kodo veikimas sugriūti dėl skirtingų Raspberry Pi lentų pakeitimų.

Priede nr.1 pateiktoje diagramoje galima pasižiūrėti, kaip atrodo naujausio mikrokompiuterio Raspberry Pi 3 modelio GPIO jungčių numeracija. BOARD numeracija yra išorėje, o BCM viduje.

Toliau pateiktame pavyzdyje panaudojamos prieš tai aprašytos komandos ir ciklo while pagalba servo varikliuko velenas sukinėjamas iš vienos pozicijos į kitą per 90◦ pirmyn ir atgal, tol kol nebus nutrauktas komandos vykdymas terminale su Ctrl+C.

Pirmoje darbo stadijoje, kurios tikslas buvo- pasukinėti servo varikliuką buvo pastebėtas keletas dalykų: Servo varikliukas sukinėsis tik pagal išskaičiuotą darbini ciklą, todėl norint pasukti veleną pagal tam tikrą laipsnių kampą bus nepatogu, nes reikia žinoti kiekvieno laipsnio darbinį ciklą.

3.4. WiringPi metodika

WiringPi yra biblioteka parašyta C kalba, skirta darbui su GPIO jungtimis. Taip pat yra skirta Raspberry Pi mikrokompiuteriams ir gali būti naudojama įvairių programavimo kalbų: C,C++, Python, Java ir t.t.

Taip pat reikia nepamiršti, kad WiringPi naudoja savo numeracijos sistemą. Ją galima pamatyti „Priede nr.2“. Vadinasi skirtumas galiausiai tarp numeracijų būtų toks: jei turime, pavyzdžiui tą pačią GPIO jungtį, tai pagal BOARD numeraciją tai galėtų būti jungtis nr. 12, kur BCM atitiktų GPIO 18 (Priedas nr.1), na ir galiausiai WiringPi (Priedas nr.2) būtų – 1.

3.4.1. WiringPI ir gpio įrankis

Toliau nagrinėjant WiringPi taip pat galima sakyti yra kaip komandinių eilučių įrankis. Pavyzdžiui norint dirbti su GPIO jungtimis ir jas priskirti išvesties tikslams ar kitiems dalykams reikės naudoji WiringPi komandinių eilučių įrankį **gpio**. Vadinasi, šitą įrankį galima panaudoti jungčių nuskaitymui ir išvedimui per komandinių eilučių interpretatorių.

3.4.2. Servo variklio valdymas naudojant WiringPi gpio

Tarkime servo varikliukui valdyti norime panaudoti 1-ą jungtį (pagal WiringPi numeraciją). Kad ją priskirtume valdymui reikią įvykdyti šią komandinę eilutę:

gpio mode 1 pwm *(5)*

Šitą komanda paruošia jungtį- ji bus naudojama išvesties impulso pločio moduliacijos signalo rėžimu. Taip pat gali būti nurodyti ir kiti rėžimai : input,output,clock, papildomai gali būti nustatyti ir vidiniai pull-up/down rezistoriai.

Jeigu norime tą pačią jungti panaudoti darbui, tik su patogesne numeracija, galime tai padaryti su ta pačia komanda ir prie jos nurodę papildomą parametrą *–g*

gpio -g mode 18 pwm *(6)*

Minėtas parametras leis GPIO jungtims taikyti BCM numeravimo sistemą. Dabar jungtis nr. 18 turi nustatytą impulso pločio moduliacijos išvesties rėžimą, tačiau vis tiek reikia dar nurodyti ir teisingą išvesties dažnį. Kaip prieš tai buvo nagrinėta, servo varikliukai nori 50Hz dažnio išvesties. Tam reikia nustatyti keletą dalykų.

Pagal vieną prieš tai minėtos Raspberry Pi bendruomenės straipsnių „*WiringPython - WiringPi for Python*“[3]: Pirma reikia turėti intervalą su pakankamu detalumu, kad būtų galima turėti kontrolės 1-2ms periode iš viso 20ms periode. Tarkime mes norim 100 pozicijų, todėl padalinus 1ms intervalą į 100 dalių mums duotų 1/100-ąją milisekundės dalį per poziciją. Taigi, mums reikia impulso pločio moduliacijos dažnio su periodu 1/100ms arba 100,000Hz[4]. Todėl moduliacijos su reikšme 100 būtų 100 dalių po 0.01ms arba tiesiog 1ms, o su reikšme 200 būtų 200 dalių po 0.01ms arba 2ms- taip turime būdą kaip reprezentuoti 1-2ms impulsą.

Tada reikia apskaičiuoti intervalo skaitliuką- kiek 0.01ms trukmės impulsų išvesti prieš kartojimą, kadangi bendras periodas yra 20ms, tai 2000 impulsų kiekvienai 0.01ms. Taip gaunasi, kad impulso pločio moduliacijos reikšmė turi būti nuo 100 (minimumas, kairė pozicija) iki 200 (maksimumas, dešinė pozicija) 2000 diapazone.

Galiausiai impulso pločio moduliacijai lieka dažnis, mes žinome kiekvieno impulso periodą, tai dažnis bus 1/periodas – 1/0.01ms ir tai gaunasi 100,000Hz . Reikia

[3] Nuoroda į straipsnį: https://www.raspberrypi.org/forums/viewtopic.php?f=32&t=10010&start=175
[4] T-periodas=1s = 1000ms = 1Hz, 1ms=1000Hz, 0.01ms= 100,000Hz

paskaičiuoti „laikrodžio daliklį“ (angl. clock divider register). Tai padaroma paimant pagrindinį „laikrodį“ (angl. master clock) - 19.2MHz ir padalinama iš 100,000Hz gaunama 192.

Visus šiuos nustatymus galima įvykdyti šiomis straipsnyje nurodytomis komandomis:

gpio pwm-ms - bus mark space rėžimas	*(7)*
gpio pwmc 192 - bus 19.2MHz / 192 = 100KHz	*(8)*
gpio pwmr 2000 - bus 2000 x 0.01ms= 20ms	*(9)*

Raspberry Pi turi du impulso pločio moduliacijos generavimo režimus[5]: „balanced“ ir “mark space“. Rėžimas „balanced“ yra numatytasis Raspberry Pi moduliacijos generavimo rėžimas. Vietoj to, kad būtų modeliuojamas impulso trukmė, rėžimas modeliuoja impulsų skaičių ir išdėsto jį vienodai per kiekvieną bazinį ciklą, o „mark space“ suteikia daugiau laisvės patiems apsibrėžti kaip generuoti impulsą.

3.4.3. Pavyzdys nr.2 Java ir WiringPi su „pwm“ rėžimu

Šiame pavyzdyje buvo pasirinkta išbandyti WiringPi gpio įrankį su Java programavimo kalba. WiringPi komandos bus įvykdomos naudojant java.lang.Runtime.exec(„komanda“) metodą. Šis metodas įvykdo nurodytos komandos eilutė atskirame procese, todėl jį naudoti yra patogu. Metodą galime iškviesti dviem būdais: „....exec(„komanda“)“ ir„....exec(„komanda“, null, null)“. Abejais atvejais iškviesti metodai elgsis lygiai taip pat, antras variantas gali turėti papildomus parametrus, tačiau jie yra visiškai neaktualūs. Taigi, pasinaudoję Java Runtime klase gauname tokius nustatymus:

```
70    Runtime runTime = Runtime.getRuntime();
71    runTime.exec("gpio -g mode 18 pwm");
72    runTime.exec("gpio pwm-ms");
73    runTime.exec("gpio pwmc 192");
74    runTime.exec("gpio pwmr 2000");
75    while(true){
```

[5] Nuoroda į straipsnį: http://www.instructables.com/id/RaspberryPi-Pulse-Width-Modulation-Demonstration/

```
76          runTime.exec("gpio -g pwm 18 "+slider.getValue());
77      }
```

Kaip matome pateiktame kodo fragmente yra naudojamas amžino while ciklas. Tame cikle yra vykdoma komanda, kurioje nurodomas impulso ilgio reikšmė. Reiškmė nurodoma slinkties juostos pagalba, tai suteikia geresnį ir patogesnį servo varikliuko valdymą, kadangi daugiau nenaudojama jokia kitą techninė įranga. Reikšmės slinkties juostoje yra apibrėžtos yra apibrėžtos nuo 100-200, pagal prieš tai aprašytus 3.4.2 skyrelyje esančius skaičiavimus, kur 100-200 reprezentuoja 1-2ms trukmės impulsą, tai jai pavyzdžiui norėsime neutralios pozicijos, jai reikės 150 (slinkties juostos reikšmės).

Dirbant su šiuo pavyzdžiu buvo pastebėta keletas dalykų:

1. Kadangi šiame darbe nėra naudojama jokių kitų įrenginių, valdiklių ir kitokių įrankių- valdymas naudojant slinkties juosta atrodo pats geriausias metodas valdyti servo varikliuką.
2. Taip pat dirbant su WiringPi gpio įrankiu buvo pastebėtas dar vienas dalykas. Prieš tai minėta komanda *(6)*, kuri yra naudojama jungties rėžimo priskyrimui, neveikia su kitomis GPIO jungtimis.

Detaliau panagrinėjus WiringPi dokumentaciją buvo rasta, kad impulso pločio moduliacijos rėžimas tinka tik vienai jungčiai – 1 (pagal WiringPi numeraciją) ar kitaip 18 (pagal BCM).

3.4.4. Pavyzdys nr.3 Java ir WiringPi su „out“ rėžimu

Kadangi, kaip jau prieš tai minėjome „pwm“ rėžimas veikia tik vienai jungčiai, žemiau pateikiamame pavyzdyje bus naudojama kita GPIO jungtis. Tarkime pasirinkime jungtį 0 (pagal WiringPi numeraciją), kitaip tariant 17, jeigu naudotume BCM numeraciją.

Kadani naudosime „out“ rėžimą su komanda „gpio –g mode 17 out“, tai mums nebereikės prieš tai aprašytų 5-9 komandų, nes jos bus nenaudojamos, tai yra, bus nereikalingos. Naudojant „out“ rėžimą, reikia žinoti, kad bus galima daryti du veiksmus: įjungti/išjungti išvestį. Kitaip sakant 0 nėra išvesties ir 1 – yra. Todėl norėdami įjungti išvesties būseną naudosime tokią komandą:

gpio -g write 17 1 *(10)*

Komandoje 17 vaizduoja GPIO jungties numerį, o 1- įjungtą būseną. Aišku, kadangi ši komanda kontroliuoja dvi būsenas, reikės nepamiršti ir išjungti būseną, vietoj 1 nurodant 0. Žemiau pateiktas kodo fragmentas pademonstruoja, kaip vietoj „pwm“ rėžimo galėtų būti naudojamas „out“ rėžimas.

```
70      Runtime runTime = Runtime.getRuntime();
71      runTime.exec("gpio -g mode 17 out");
72      while(true){
73                  runTime.exec("gpio -g write 17 1");
74                  java.lang.Thread.sleep(slider.getValue()/100);
75                  runTime.exec("gpio -g write 17 0");
76                  java.lang.Thread.sleep(20- slider.getValue()/100);
77      }
```

Pagal kodo fragmentą matome, kad jungtims atlikti nustatymus vos tereikia vienos komandos. Tačiau čia atsiranda kitas aspektas. Kadangi čia nebėra naudojamas „pwm“ rėžimas, mes nebepanaudosime prieš tai minėto impulso pločio moduliacijos generatoriaus. Čia jau reikia patiems apibrėžti norimo impulso trukmę ir tai galim padaryti panaudodami Java klasės Thread *sleep* metodą. Kaip galima matyt sleep metodui yra paduodamas parametras, kuris yra nurodytas skirtinais būdais:

1. „`java.lang.Thread.sleep(slider.getValue()/100);`“ – kadangi sleep metodui paduodamas parametras yra milisekundės, tai jei slinkties juosta grąžintų reikšmę 150, būtų kviečiamas sleep(1.5), kas reikštų jog suteikiamas impulsas, kurio ilgis būtų 1.5ms, kas reikštų servo varikliuko neutralią padėtį. Taigi įjungus išvesties būseną iškart yra užmigdomas gijos darbas nurodytam laikui, paliekant įjungtą išvesties būseną.

2. „`java.lang.Thread.sleep(20- slider.getValue()/100);`“ –nurodo likusį laisvą laiką iš 20ms, kuriuo metu turi būti išjungta išvesties būsena, prieš vėliau paduodant kitą impulsą. Taigi, jeigu nagrinėtume neutralią poziciją, tai prieš tai esančiame punkte būtų slinkties juosta su reikšme 150, tai

įjungta išvesties būsena būtų 1.5ms ir šiame punkte naudojamos komandos dėka likusį 18.5ms (20ms-1.5ms) laiką išvesties būsena būtų išjungta.

Deja, naudojant šį būdą pastebėtas trūkumas. Nors ir servo varikliukas yra puikiai valdomas slinkties juostos pagalba, tačiau, kai slinkties juosta yra nejudinama, palikta vienoje pozicijoje, servo varikliukas pastoviai spurda ir atlieka labai nežymius posūkius pirmyn ir atgal aplink tą poziciją, ties kurią buvo palikta slinkties juostos reikšmė. Vadinasi, servo varikliukas ramybės būsenoje tampa nestabilus ir tikrai toks būdas nebus tinkamas robotizuotos rankos valdymui, kai vienu metu panašus kodo fragmentas turės būti pritaikytas visiems 6 servo varikliukams.

3.5. Pi4J – alternatyva WiringPi

Pi4J yra atvirojo kodo projektas, suteikiantis bibliotekas, skirtas Java programuotojams, kad jie galėtų lengvai sąveikauti su žemo lygio įvesties ir išvesties galimybėmis, naudojant Raspberry Pi. Daugiau panagrinėjus šią biblioteką buvo pastebėta, kad ji turi daug panašumų su WiringPi:

1. Pirmiausia, Pi4J naudoja GPIO jungčių numeraciją kaip ir WiringPi (Priedas nr.2).
2. Toliau dar buvo pastebėta, kad Pi4J yra „pwm“ rėžimas, impulso pločio moduliacijos generavimo būdai, kaip ir WiringPi.
3. Taip pat buvo pastebėta, kad impulso pločio moduliacija irgi galioja, tik tai pačiai vienai jungčiai kaip ir WiringPi ir ant kitų jungčių neveikia.
4. Pi4J programas reikia paleisti su ***sudo***, kitaip neveiks.

Susidaro tokia nuomonė, kad Pi4J yra kaip WiringPi sąsaja, skirtumas tas, kad čia Pi4J labiau objektiškai orientuotas, bet išlaiko lygiai tokį patį funkcionalumą kaip ir WiringPi, nes komandos panašios ir abiejuose metoduose reikia naudoti Java Thread klasės sleep metodą impulso trukmei valdyti.

3.6. GPIO programavimo įrankių analizė

Įrankio pavadinimas	RPi.GPIO	WiringPi	Pi4J
Kalba, kuria parašyta	Python	C	Java
Palaikomos kalbos	Python	C,C++, Python, Java,...	Java
Impulso pločio moduliacija (PWM)	„pwm“ rėžimas galimas daugiau nei vienai jungčiai, naudojamas darbinis ciklas.	„pwm“ rėžimas veikia tiktais vienai GPIO jungčiai nr.18 pagal BCM numeraciją ir nr.1 pagal WiringPi, kaip ir RPi.GPIO naudojamas darbinis ciklas. „out“ rėžimas panaudojamas sudarant impulso pločio moduliaciją, naudojant atitinkamos funkcijos, kurios vykdo jungties išvesties išjungimo/įjungimo būsenas .	
Jungčių numeracijos	BOARD/BCM	WiringPi/BCM	
Privalumai	„pwm“ rėžimas veikia su daugiau nei viena jungtimi, todėl „out“ rėžimas visiškai neaktualus paprasta naudoti	suderinamumas su skirtingomis programavimo kalbomis funkcionalumas gali būti pasiektas per komandines eilutes arba per jas atitinkančius metodus.	Objektiškai orientuota.
Trūkumai	naudoti pritaikyta tiktais su Python	„pwm“ rėžimas veikia tik vienai jungčiai „out“ rėžimas nesuteikia stabilaus servo varikliuko valdymo reikia papildomų žinių apie impulso pločio generavimo rėžimus	kadangi yra specialiai skirta Java kaip WiringPi implementacija, paveldi visus minusus norint vykdyti programą, reikia paleisti ***sudo*** rėžimu, todėl nepatogu dirbti naudojant IDE

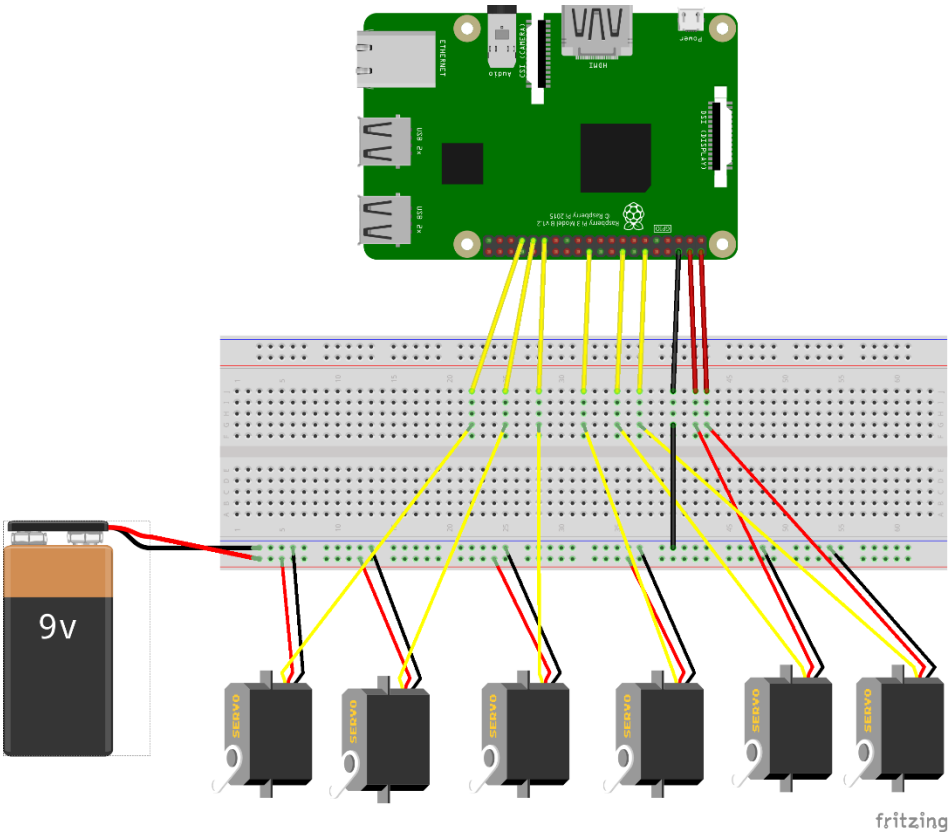
Kadangi WiringPi ir Pi4J metodikos valdyti servo varikliukams nepasiteisino tiek dėl patogumo valdyti servo varikliukui (tinka vienam, bet iš viso bus juk 6), tiek dėl stabilumo, tolimesniam darbui jau minėtų būdų ir nebenaudosime. Taigi, rankai valdyti belieka Rpi.GPIO metodika.

4. Robotizuotos rankos valdymas, naudojant Raspberry Pi

Išsirinkus metodą valdyti vieną servo varikliuką, galima jau pradėti galvoti apie robotizuotos rankos valdymą. Robotizuotai rankai (Priedas nr.3) valdyti reikės 6 servo varikliukų. Kadangi robotizuota ranka bus valdoma kelių servo varikliukų, ją galima suskirstyti į atskiras dalis:

1. Pirštai (žnyplės)- atstotų žmogaus rankos pirštus. Jos galėtų tik susispausti, kad būtų paimamas daiktas, ir po to – atsilaisvinti.
2. Plaštaka- atstotų žmogaus rankos plaštaką. Ši dalis turėtų judesius į šonus, bet ne aukštyn-žemyn.
3. Riešas – atstotų žmogaus riešą ir galėtų judėti tik aukštyn ir žemyn.
4. Alkūnės sąnarys – atstotų žmogaus alkūnės sąnarį, turėtų judesius aukštyn ir žemyn.
5. Peties sąnarys – atstotų žmogaus peties sąnarį. Šiai daliai reikės dviejų servo varikliukų, kadangi žmogaus peties sąnarys leidžia ranka judinti ne tik aukštyn-žemyn, bet ir į šonus.

4.1. Servo varikliukų jungimo schema



Diagramoje servo varikliukai (pagal prieš tai aprašytas dalis) išsidėstę tokia tvarka, iš kairės į dešinę: peties sąnarys – pirštai. Atitinkamai bus suteikiamos GPIO jungčių numeracijos pagal BCM sistemą (irgi iš kairės į dešinę): 13, 6, 5, 25, 23, 18.

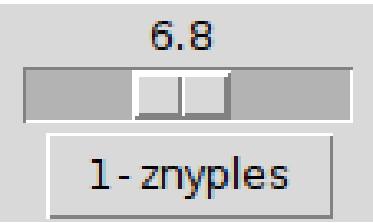
Pastaba: modeliavimo programa neturi galimybės atvaizduoti AC-DC adapterio (5V), todėl vietoj jo vaizduojama kronos baterija, kuri irgi gali būti maitinimo šaltinis servo varikliukams.

4.2. Rankos valdymas su Python ir Rpi.GPIO

Kadangi slinkties juosta (Java būdas su WiringPi) valdant servo varikliuką pasiteisino kaip geriausias metodas valdymui pagal turimas galimybes, tai ja pasinaudosime ir visos rankos valdymui. Reikėtų pasidaryti visoms 5 dalims po slinkties juostas, iš viso būtų – 6, nes peties sąnariui reikės 2.

4.2.1. GUI su Python Tkinter

Tkinter yra Python įrankis skirtas darbui su grafine vartotojo sąsaja ir yra pristatomas kartu su Python. Jo dėka rankai valdyti bus sukurtos slinkties juostos, kurios bus išdėstytos taip: robotizuotos rankos dalims, kurios gali judėti horizantaliai, valdymo slinkties juostos bus atvaizduotos horizontaliai, vertikaliai judančioms dalims- vertikalios. Sukurtą pilną grafinę vartotojo sąsają galima pamatyti „Priedas nr.4“. Žemiau pateikiama tik dalis GUI.



Kaip matome iš pateikto paveiksliuko, 1 dalis atitinka žnyples, kas ir yra nurodyta prie valdymo slinkties juostos. Reikšmę, kurią mes matome (6.8) yra darbinio ciklo reikšmė, apie darbini ciklą buvo kalbama 3.3 skyriuje.

4.2.2. Programos procesas

Programos veikimą suskirstytas į šias dalis:

1. GPIO jungčių ir pradinių servo varikliukų būsenų nustatymas.
2. Robotizuotos rankos (arba visų 6 servo varikliukų) valdymas.
3. Programos darbo pabaigos dalis.

Pirmoje dalyje kaip ir prieš tai paminėta, yra nustatomos kaip ir kokios bus naudojamas Raspberry Pi GPIO jungtys. Taip pat iškarto yra paleidžiami servo varikliukai (vienas po kito) su nustatytomis pradinėmis pozicijomis. Taip padaryta dėl to, kad būtų patogiau iškarto valdyti jau nuo pagrindo pakilusią robotizuotą ranką. Žemiau pateikiamas kodo fragmentas kaip visa tai atrodo:

```
17 #Servo varikliuku apibrezimas ir surisimas su GPIO jungtimis
18
19 #1- znyples
20 print "Sukonfiguruojamas 1-as servo varikliukas - znyples"
21 GPIO.setup(18, GPIO.OUT)
22 servo_1 = GPIO.PWM(18, 50)
23 servo_1.start(Const.S1)
24 time.sleep(1)
```

Reikėtų atkreipti dėmesį į 23-24 eilutes. Eilutėje nr.23 vietoj pradinio darbinio ciklo yra nurodoma konstanta, kuri yra laikoma pradine ir galutine servo varikliuko būsena. Kiekvienas servo varikliukas turi savo atskirą konstantos reikšmę. Taip yra padaryta dėl to, kad surenkant ranką buvo išsukinėti servo varikliukai ir detalių dalių sujungimas nebuvo turbūt pasirinktas pats geriausias. Antra to priežastis bus aptarta prie programos baigimo dalies. Toliau 24-oje eilutėje naudojama time.sleep(1) su tikslu, kad kiekvienas servo varikliukas įsijungtų į darbą po vieną, ramiai, ir kad nepradėtų iškart judėti visa ranka, kadangi ranka yra iš metalinių detalių, ji yra sunki ir šiuo atveju gali nusverti jos pagrindą, nes nėra rankos kur pritvirtinti.

Antroje dalyje vyksta robotizuotos rankos valdymas, prieš jau yra pateiktas slinkties juostos 4.2.1 skyrelyje pavyzdys. Slinkties juosta gali turėti 2.5-12.5 darbinio ciklo reikšmes, kas atitinka 1-2ms trukmės impulsus.

Trečioje dalyje, kai yra išjungiama programa, prieš galutinai užsidarant programai yra įvykdomas keletas veiksmų. Pirmiausiai, nuspaudus valdymo lango uždarymo mygtuką, yra visi servo varikliukai (po vieną), grąžinami į pradines padėtis. Tai leidžia robotizuotai rankai palengva nusileisti, nesitrankant į žemiau esantį pagrindą. Kai visi servo varikliukai yra išjungti, yra įvykdoma GPIO jungčių nustatymų išvalymas, ir galiausiai, programa išsijungia. Tai galima pamatyti žemiau pateiktame kodo fragmente.

```
158     print "servo varikliukas nr.6 baigia darba..."
159     servo_6.ChangeDutyCycle(Const.S6)
160     time.sleep(1)
161     servo_6.stop()
162
163     print "GPIO jungciu isvalymas..."
164     GPIO.cleanup()
165
166     print "programa baigia darba..."
```

4.3. Pastebėjimai valdant robotizuotą ranką

Keletą kartų paleidus programą ir pabandžius valdyti robotizuotą ranką buvo pastebėti keli dalykai:

1. Dėl skirtingai išsukinėtų servo varikliukų, kai kurios robotizuotos rankos dalių valdymas yra apribotas – negalimas, nes didesniam judesiui atlikti trukdo kitos rankos dalys, pagrindas.
2. Slinkties juostos panaudojimas visos rankos valdymui pasiteisino, tačiau reikia būti atsargiems, nes greitai slankiojant slinkties juostą, ranka taip pat staigiai atlieka judesius, todėl reikia ją valdyti atidžiai, kad būtų išvengta žalos, nes robotizuota ranka yra metalinė.
3. Žnyples valdantis servo varikliukas, palyginus su kitų dalių servo varikliukais, gali labai greitai užkaisti. Tai gali būti dėl to, kad žnyplės yra surinktos iš kelių dalių, kurios yra tvirtai prisuktos, kas turbūt reikalauja daugiau darbo iš servo varikliuko.
4. Stabiliam rankos valdymui reikia įtvirtinti jos pagrindą, priešingu atveju ranką nusvers jos dalys.

Rezultatai ir išvados

Darbo metu buvo ištirtos galimybės slinkties juostų pagalba valdyti šešis servo varikliukus, kurie skirti visos robotizuotos rankos valdymui. Taip pat buvo ištirti ir palyginti skirtingi įrankiai: RPi.GPIO, WiringPi ir Pi4J, skirti darbui su GPIO jungtimis ir servo varikliukais, išsiaiškintos jungčių ir varikliukų jungimo schemos.

Darbo išvados:

1. Raspberry Pi 3 mikrokompiuteris tinkamas robotizuotai rankai valdyti.
2. Naudoti įrankiai nesuteikia vienodų galimybių servo varikliukams valdyti – skirtingi impulso pločio moduliacijos generavimo būdai.
3. Patogiausias įrankis servo varikliukams valdyti – RPi.GPIO.
4. Norint sukurti patogesnį robotizuotos rankos valdymą reikia papildomos įrangos ir laiko.

Rekomendacijos:

1. Pritaikyti papildomą įrangą robotizuotos rankos valdymui. Tai gali būti paprasčiausias žaidimų pultelis (angl. joystic/gamepad) arba kokie nors kiti valdikliai. Naudojant papildomą įrangą reikės naudoti atitinkamas GPIO jungtis su įvesties rėžimais.
2. Išanalizuoti ir išbandyti kitas GPIO sąsajas: I2C, SPI ir UART, skirtas mikrovaldikliams.

Summary

Nowadays usage of microcomputers is high. Their size is compared to credit card‘s so it is very convenient to have and use them. There are lots of home projects called DIY- Do it yourself. People use microcomputers in many different ways. Some of them use them in robotics to control robot arms and build up car models or many other things. On the other hand microcomputers are also used for building small game arcades, or they are used with other hardware that may have different kinds of sensors. One of the biggest advantages of microcomputers is that they are used by people with different ages, so it is a good tool for beginners (especially for youth). Raspberry Pi is one of microcomputer‘s brands. During past few years it‘s popularity increased and doesn‘t stop growing now.

This thesis investigates how Raspberry Pi‘s microcomputer can be used to control robot arm. It also investigates how to control arm‘s servo motors in the most convienent way as much as possible without any other equipment or any additional hardware. One of the purpose was to investigate different tools that could be used for servo motor and robot arm control.

The result of this papper is an overview of different approaches how servo motors could be controlled with different tools. Some of the tools appeard to be not as convenient as the other. But this may be due to that tools are not created for only working with servo motors on microcomputers.

Literatūros sąrašas

[Cam] University of Cambridge. Physical computing with Raspberry Pi, URL: https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/

[Don14] Wolfram Donat. Learn Raspberry Pi Programming with Python. 256psl. 2014-05-08

[Gor] Gordon Henderson. WiringPi URL: http://wiringpi.com/

[HS] Howard Eglowstein, Science Buddies. Introduction to Servo Motors, URL: http://www.sciencebuddies.org/science-fair-projects/project_ideas/Robotics_ServoMotors.shtml

[Mon16] Simon Monk. Software, URL: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor/software , 2016-10-29

[Nor] Donald Norris. Raspberry Pi Projects for the Evil Genius. 224 psl. 2013-09-05

[Ras13] Raspberry Pi Stack Exchange. Control Hardware PWM frequency, URL: https://raspberrypi.stackexchange.com/questions/4906/control-hardware-pwm-frequency , 2013-02-08

[Ras13] RasPi.TV. RPi.GPIO basics 3 – How to Exit GPIO programs cleanly, avoid warnings and protect your Pi, URL: http://raspi.tv/2013/rpi-gpio-basics-3-how-to-exit-gpio-programs-cleanly-avoid-warnings-and-protect-your-pi , 2013-07-13

[RPi] RPi Labs. Controlling a Servo from the Raspberry Pi ,URL: http://rpi.science.uoit.ca/lab/servo/

[Sca15] Matthew Scarpino. Motors for Makers: A Guide to Steppers, Servos, and Other Electrical Machines. 320psl. 2015-11-30

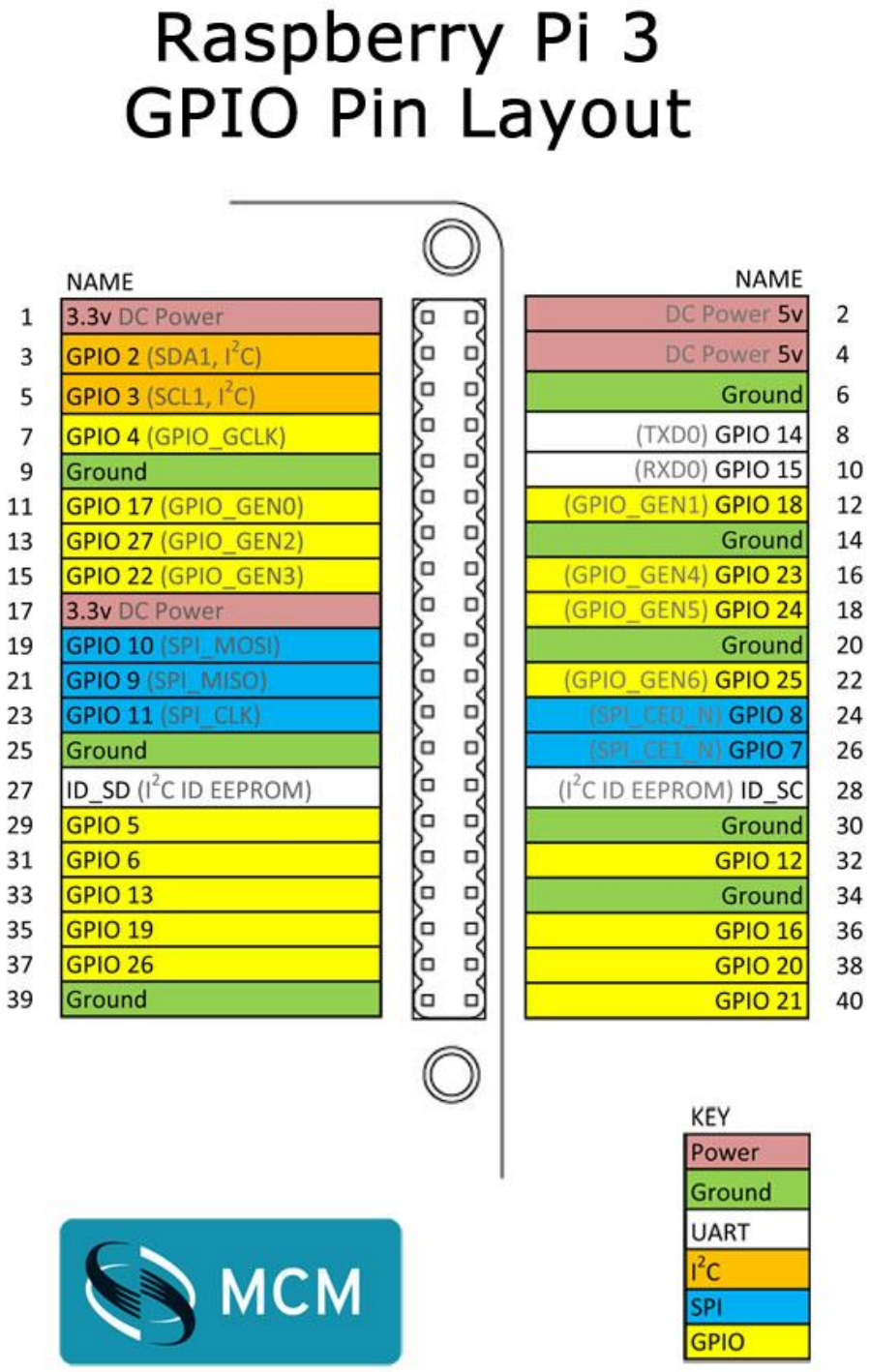
[Tow] TowerPro. MG996R Robot servo 180° Rotation, URL: http://www.towerpro.com.tw/product/mg995-robot-servo-180-rotation/

[Wen15] Richard Wentk Raspberry Pi for Kids For Dummies, 352psl. 2015-07-28

[War14] Warren Gay.Mastering the Raspberry Pi. 504psl. 2014-08-17

[Who15] Paul McWhorter. Raspberry Pi LESSON 28: Controlling a Servo on Raspberry Pi with Python, URL: http://www.toptechboy.com/raspberry-pi/raspberry-pi-lesson-28-controlling-a-servo-on-raspberry-pi-with-python/ , 2015-06-02

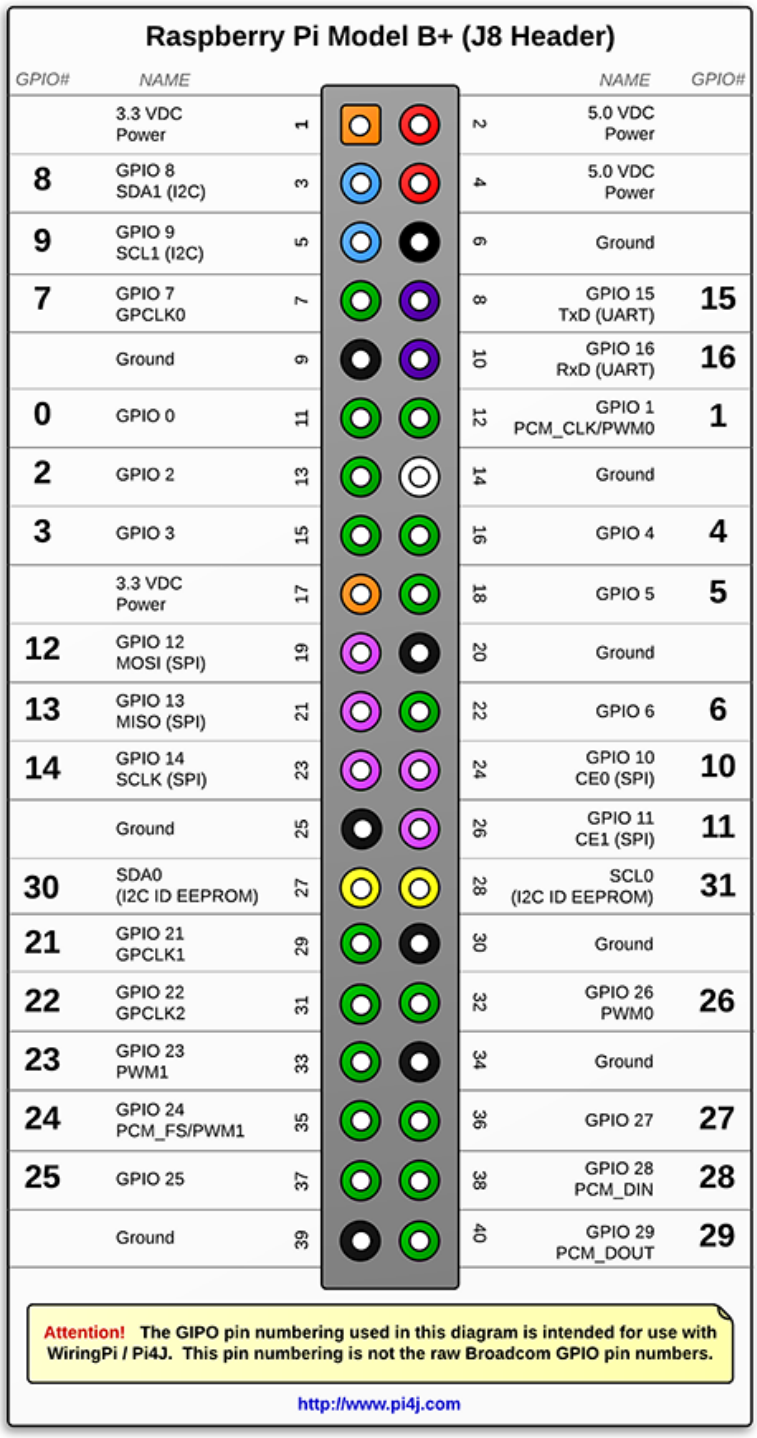
Priedas nr.1



Raspberry Pi 3 B GPIO jungčių išsidėstymas.

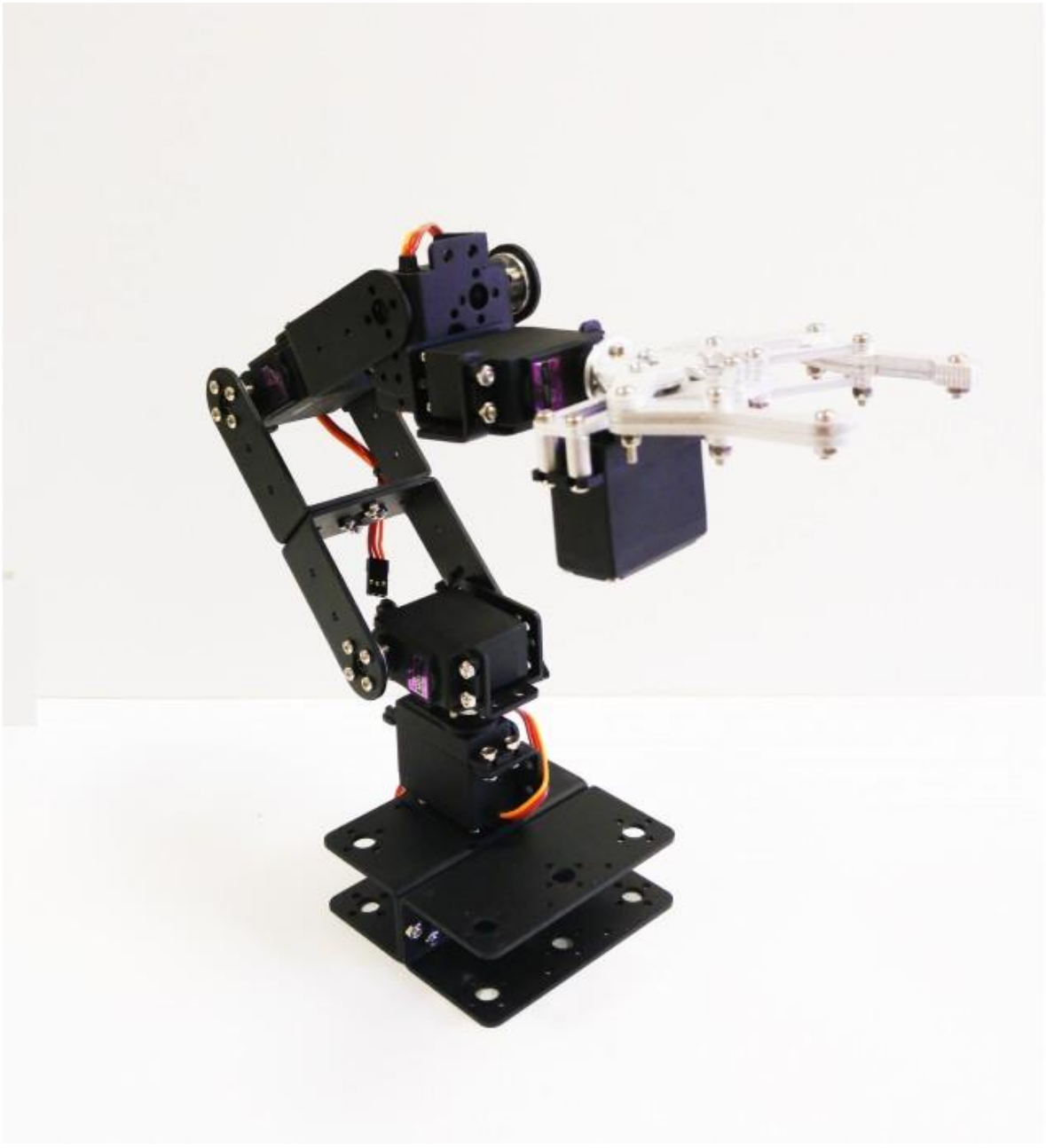
Nuoroda:http://blog.mcmelectronics.com/image.axd?picture=%2f2016%2f03%2fGPIO-Chart2.jpg

Priedas nr.2



Raspberry Pi B+ modelių GPIO jungčių išsidėstymų numeracija. Stulpelis GPIO# nurodo numeraciją naudojamą WiringPi ir Pi4J
Nuoroda: http://pi4j.com/pins/model-b-plus.html

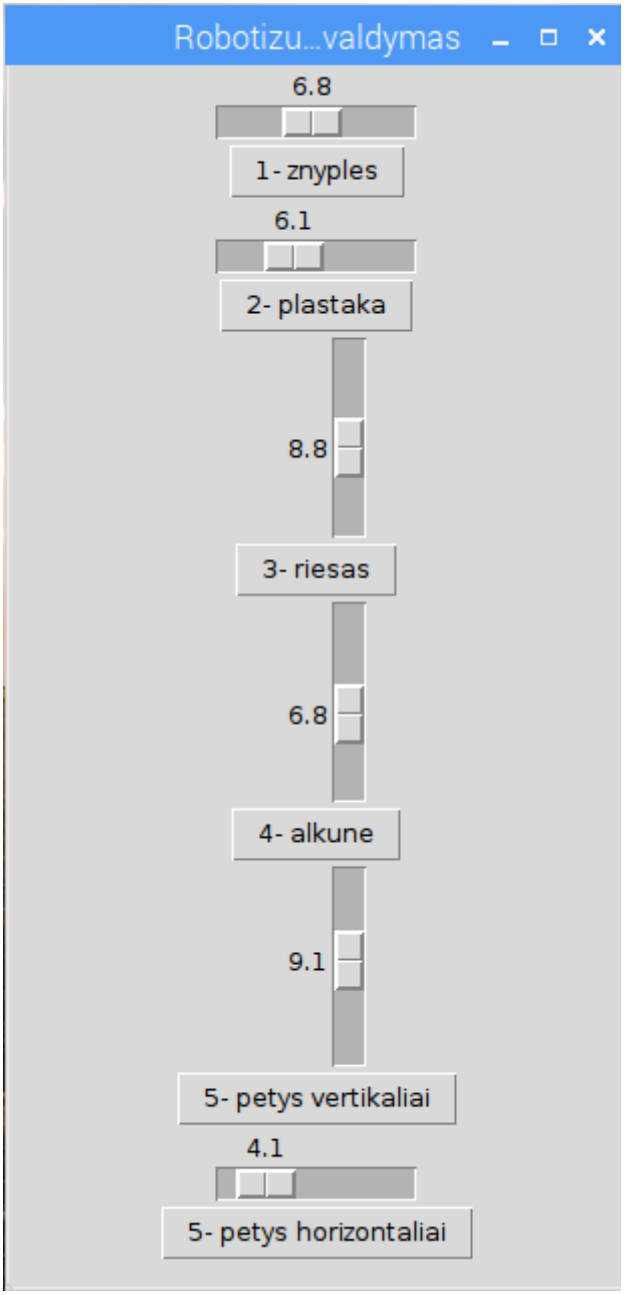
Priedas nr.3



Aliuminė šešių servo varikliukų valdoma robotizuota ranka

Nuoroda:http://www.hobbytronics.co.za/content/images/thumbs/0003761_robotpark-6-dof-aluminum-robot-arm-kit-info.jpeg

Priedas nr.4



Tkinter pagalba sukurtas GUI robotizuotai rankai valdyti. Pilnas valdymo lango pavadinimas „Robotizuotos rankos valdymas“