|  |  |
| --- | --- |
| tallinna tehnikaülikool | |
| Infotehnoloogia teaduskond | |
| Automaatikainstituut | |
|  | |
| ISL40LT | |
| Kevin Ree 134862IASB | |
| helile järgnev robotauto | |
| Bakalaureusetöö | |
| Juhendaja: | Erki Suurjaak |
|  | Bakalaureus |
|  | Insener |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kevin Ree

10.06.2016

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks oli ehitada ja programmeerida robotauto, mis on võimeline leidma ruumis oleva liikuva või seisva heliallika ja sellele järgnema, kuniks robotauto tee pole takistatud. Heliallikas, millele robotauto reageerib, peab olema tugevam kui servomootorite tekitatud müra. Heliallika leidmiseks kasutatakse Tallinna Tehnikaülikooli Proaktiivtehnoloogiate Teaduslabori poolt koostatud C programmi.

Töös oli kolm põhiülesannet, millest esimene oli robotauto elektrooniliste komponentide ühendamine ja tööle saamine. Teiseks oli vaja leida viis, kuidas juhtida robotauto servomootoreid efektiivselt erineva suuna ja kiirusega. Kolmandaks oli sensorite lugemise üheaegne integreerimine ja liikumisotsuste vastuvõtmine.

Töö tulemuseks on töötav robotauto, mida juhitakse Pythoni-keelse programmiga ruumis kõige valjema heliallika suunas, vältides mootorite mürast tekitatud valesid suundi. Juhtimiseks kasutatakse C programmist saadud heliallika asukoha nurga ja helitugevuse väärtusi. Robotauto liigub heliallika poole, kuni tema tee ei ole takistatud.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 9 peatükki, 21 joonist, 0 tabelit.

Abstract  
Sound-Following Robot Car

The aim of this thesis was to build and program a robot car that is able to detect a stationary or a moving sound source in the room, and move towards it until the chosen route is obstructed. The car must only follow sound that is stronger than the noise generated by its servo motors. The sound source is located using C code provided by the Research Laboratory for Proactive Technologies.

There were three main tasks in this work, the first of which was wiring all the electronic components and getting them to work. Secondly, we needed to find a way to control the robot car servo motors efficiently at different speeds and directions. Lastly, we needed to integrate sensor readings and control car driving.

As a result we have a working robot car, controlled by Python program, that follows the loudest sound source in the room while ignoring the noise generated by servo motors. To control the robot we use data from C program: sound source angle and volume level. Robot car will follow the sound source until the chosen route is obstructed.

The thesis is in Estonian and contains 30 pages of text, 9 chapters, 21 figures, 0 tables.

Lühendite ja mõistete sõnastik

|  |  |
| --- | --- |
| BBB | Beaglebone Black |
| PWM | *Pulse-width modulation*, pulsilaiusmodulatsioon |
| DC | *Duty cycle*, töötsükkel |
| I2C | Inter-Integrated Circuit, kahejuhtmeliides |
| RAM | Random access memory, suvapöördusmälu |
| PRU | *Programmable Real-Time Unit,* programmeeritav reaalaja üksus |
| IDE | *Integrated development environment*, integreeritud Programmeerimiskeskkond |
| USB | *Universal Serial Bus*, universaalne järjestiksiin |
| SCL | *Serial Clock Line*, taktisignaali liin |
| SDA | *Serial Data Line*, andmeliin |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Sisukord

[1 Sissejuhatus 8](#_Toc453336788)

[2 Robotauto komponendid 9](#_Toc453336789)

[3 Miniarvuti: BeagleBone Black 10](#_Toc453336790)

[4 Servomootorite juhtimine 11](#_Toc453336791)

[4.1 *PWM* signaal 11](#_Toc453336792)

[4.2 Servomootorite liikumise esimene katse 12](#_Toc453336793)

[4.3 Servomootorite liikumise teine katse 13](#_Toc453336794)

[5 Ultraheli kaugusmõõtjate kasutamine 17](#_Toc453336795)

[6 Heliallika leidmine 19](#_Toc453336796)

[7 Robotauto komponentide ühendused 24](#_Toc453336797)

[8 Robotauto juhtimise algoritm 26](#_Toc453336798)

[9 Kokkuvõte 29](#_Toc453336799)

[Kasutatud kirjandus 30](#_Toc453336800)

[Lisa 1 – Robotauto ümberehitus, et mõõta servomootorite kiirust 31](#_Toc453336801)

[Lisa 2 – Robotauto ühendused enne vaheplaati 32](#_Toc453336802)

[Lisa 3 –Servomootorite liikumise Python funktsioon 33](#_Toc453336803)

[Lisa 4 – Ultraheli kaugusmõõtjatest kauguse saamise Python kood 34](#_Toc453336804)

[Lisa 5 –Python põhitsükkel 35](#_Toc453336805)

[Lisa 6 – Robotauto mikrofonide kinnitused 36](#_Toc453336806)

[Lisa 7 – Valmis robotauto 37](#_Toc453336807)

Jooniste loetelu

[Joonis 1. Beaglebone Black sisend-väljund viikude diagramm [2] 10](#_Toc453336808)

[Joonis 2. Esimene katse: vasaku külje liikumise test 13](#_Toc453336809)

[Joonis 3. Esimene katse: parema külje liikumise test 13](#_Toc453336810)

[Joonis 4. Teine katse: vasaku külje liikumise test 14](#_Toc453336811)

[Joonis 5. Teine katse: parema ja vasaku külje liikumise test 15](#_Toc453336812)

[Joonis 6. Teine katse: parema ja vasaku külje liikumise test vastupidises suunas 15](#_Toc453336813)

[Joonis 7. SRF08 ultraheli kaugusmõõtja [8] 17](#_Toc453336814)

[Joonis 8. Teise I2C seadme aktiveerimine 17](#_Toc453336815)

[Joonis 9. I2C siinide aadressid 18](#_Toc453336816)

[Joonis 10. Kahesuunalise loogilise taseme konverter [10] 18](#_Toc453336817)

[Joonis 11. INMP401 (ADMP401) mikrofon [11] 19](#_Toc453336818)

[Joonis 12. Heliallika positsioneerimine: heliallikas- (A) taga paremal (B) taga vasakul (C) otse paremal (D) otse vasakul 20](#_Toc453336819)

[Joonis 13. Heliallika positsioneerimine: heliallikas- (A) ees keskel (B) taga keskel 20](#_Toc453336820)

[Joonis 14. Heliallika tugevus: keskväärtus 21](#_Toc453336821)

[Joonis 15. Heliallika tugevus: normeeritud väärtus 21](#_Toc453336822)

[Joonis 16. Heliallika tugevus: väärtuste summa 22](#_Toc453336823)

[Joonis 17. Heli tugevuse mõõtmised 22](#_Toc453336824)

[Joonis 18. Trükkplaadi disain 24](#_Toc453336825)

[Joonis 19. Robotauto komponentide ühendused 25](#_Toc453336826)

[Joonis 20. Heli tugevuse piiramine 27](#_Toc453336827)

[Joonis 21. Takistuse vältimine 28](#_Toc453336828)

# Sissejuhatus

Käesoleva bakalaureusetöö teema sai valitud Tallinna Tehnikaülikooli Proaktiivtehnoloogiate Teaduslabori pakutavatest teemadest. Töö põhieesmärgiks on ehitada ja programmeerida autonoomne robotauto, mis on võimeline leidma ruumis oleva liikuva või seisva heliallika ja sellele järgnema. Heliallika leidmiseks kasutatakse TTÜ Proaktiivtehnoloogiate Teaduslabori poolt koostatud C programmi. Samuti on töö eesmärgiks saada praktiline kogemus ja oskus robootika, riistvara ja sensoorika kasutamisel ja programmeerimisel.

Töös on kolm põhiülesannet, millest esimene on robotauto elektrooniliste komponentide ühendamine ja tööle saamine. Teiseks on vaja leida viis, kuidas juhtida robotauto servomootoreid efektiivselt erineva suuna ja kiirusega. Kolmandaks on vaja saada üheaegselt sensorite väärtusi ja vastu võtta liikumisotsuseid.

Robotautole esitatavad nõuded: robotauto peab sõitma heliallika suunas nii kaua, kuni tema liikumistee pole takistatud. Kui leidub takistus või robotauto on jõudnud heliallikani, siis ta ootab uue või tugevama heliallika teket. Robotauto reageerib heliallikale, mis on küllalt tugev, et ületab robotauto mootorite tekitatud müra.

# Robotauto komponendid

Robotauto komponendid:

* metallkere,
* neli HS-311 Hitec servomootorit,
* kaks Intenso akupanka,
* kaks SRF08 ultraheli kaugusmõõtjat,
* kaks kahesuunalise loogilise taseme konverterit,
* neli ADMP401 mikrofoni,
* trükkplaat,
* Beaglebone Black miniarvuti.

Robotauto metallkere koos nelja servomootoriga sai võetud Automaatikainstituudi Proaktiivtehnoloogiate teaduslabori varudest. Robotauto juhtimiseks kasutatakse Beaglebone Black miniarvutit, mis on sobilik oma kõrge võimsuse ja suure sisend-väljund viikude arvu poolest, ja omab analoog-digitaalmuundureid, mis on vajalikud mikrofonidega suhtlemiseks.

Robotauto akupangad toidavad miniarvutit ja servomootoreid. Esimest akupanka Intenso 5200 mahtuvusega 5200mAh ja väljundiga 1.0A kasutatakse *BBB* miniarvuti toitena. Teist akupanka Intenso s6000, mille mahtuvus on 6000mAh ja väljundiks on 2.1A kasutatakse nelja servomootori toiteks

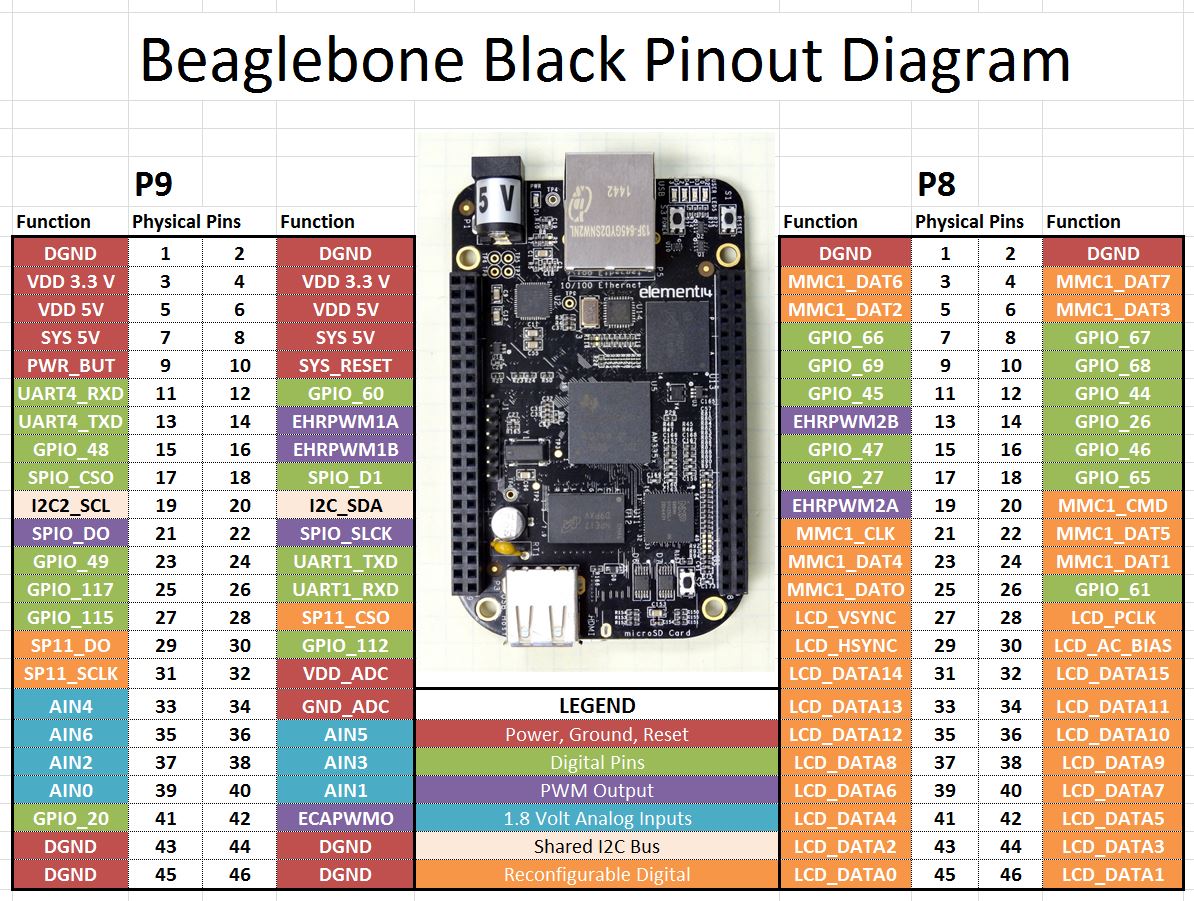
# Miniarvuti: BeagleBone Black

Beaglebone Black (Joonis 1) on hinnalt soodne miniarvuti, mille peal jookseb Linuxi tuumal põhinev operatsioonisüsteem. Antud töös on kasutuses operatsioonisüsteem Debian GNU/Linux 7.4.

*BBB* näitajad;

* AM335x 1GHz ARM Cortex-A8 protsessor,
* 512MB *RAM*,
* 4GB välkmälu (inglise keeles *flash storage*),
* 2x *PRU* 32-bit mikrokontrollerit [1].

*BBB* omab kaheksat *PWM* viiku, seitset analoog sisendit ja kahte *I2C* siini, mida meil on vaja robotauto juhtimisel [2].



Joonis . Beaglebone Black sisend-väljund viikude diagramm [2]

# Servomootorite juhtimine

Robotauto kasutab liikumiseks nelja HS-311 Hitec servomootorit. Mootoreid kontrollitakse *PWM* signaaliga. Kuna mõlemal robotauto küljel on kaks mootorit, mis peavad liikuma samas suunas ja sama kiirusega, saame kontrollida nelja mootorit kahe *PWM* signaaliga. Üks *PWM* signaal juhib robotauto vasakut külge ja teine signaal juhib paremat. Sellel roboti ehitamise staadiumil kasutatakse *BBB* toitena mini–A *USB* arvuti ühendust ja mootorite toiteks kasutatakse 5V 3000mA toitekaablit, kuna *BBB* ei suuda korraga nelja servomootorit ära toita. Iga servomootor vajab töötamisel 160mA –180mA aga *BBB* miniarvuti SYS 5V (Joonis 1) väljastab ainult 250mA [3], [4].

## *PWM* signaal

*BBB* omab kaheksat *PWM* viiku, millest kasutame kahte viiku: P9.14 ja P8.19 (Joonis 1), mis on juba vaikimisi aktiveeritud. Viikude valik ja kogus on selline, kuna BBB ei lase samal ajal teatud viikude kombinatsioone nagu P9.14 ja P9.16 (Joonis 1) deklareerida [2], [5].

Servomootoreid juhitakse sagedusega 50Hz (20mS) ja 4.8V–6V toitega. Esimesed servomootorite juhtimise katsed 50Hz–ga ei andnud korrektseid tulemusi. *PWM* signaalid ei olnud teineteisest sõltumatud ja selle tagajärjeks oli servomootorite mittekorrektne liikumine. Kui parema külje *DC* on suurem kui vasaku külje oma, siis mootorid liiguvad päripäeva ja kui vasaku külje *DC* on suurem paremast, siis mootorid liiguvad vastupäeva. Samuti tootja poolt antud sagedusega tekkisid mootorite liikumisel pausid. Algne hüpotees oli, et probleemiks on ühe *PWM* signaaliga kahe mootori juhtimine, aga see hüpotees ei leidnud kinnitust, kuna kahe servomootori juhtimise testimine kahe erineva *PWM* signaaliga ei parandanud juhtimise probleeme. Järgnevates testides testiti *PWM* signaali mõjutamist nii erinevate *DC* väärtustega kui ka erinevate sagedusega, ja *PWM* viikude vahetust teiste võimalike kombinatsioonidega. Ükski läbi viidud testidest ei lahendanud probleemi [6].

## Servomootorite liikumise esimene katse

Kuna *PWM* signaali probleemile lahendust ei olnud, oli vaja leida viis, kuidas robotautot juhtida. Selleks oli vaja robot ümber ehitada (Lisa 1), et mõõta servomootorite suunda ja kiirust teatud *DC* väärtuste kombinatsioonidega.

Et leida *DC* väärtuste vahemikud, millega saaks robotautot juhtida, tuli mõõta erinevate *DC* kombinatsioonidega servomootorite kiirusi. Testi käigus mõõdeti servomootorite pöörete arvu 20 sekundi jooksul ja pandi kirja mootorite suund iga lugemise juures. Pöörete arvu loeti ultraheli kaugusanduritega (Joonis 7), kasutades märkmepaberit täisringi märkimiseks. Kui märkmepaber oli kaugusandurile lähemal kui 6cm, pandi kirja 1 pööre, ja seda lugemist tehti 20 sekundit, peale mida küsiti mootorite suunda. Testi viidi läbi kaks korda. Esimene test uuris vasaku külje *DC* sõltuvust ja teine test parema külje *DC* sõltuvust. Esimese testi käigus suurendati vasaku külje *DC* väärtust viie võrra pärast iga 20-sekundilist lugemist ja kui vasaku *DC* väärtus ületas sadat, siis suurendati parema poole *DC* väärtust kümne võrra sajani. Teises testis tehti vastupidiselt, paremat suurendati viie võrra sajani ja vasakut kümne võrra sajani. Mõlemas testis tehti 210 lugemist. Pöörete lugemiste viga on ±1 pööre.

Vaadates vasaku (Joonis 2) ja parema (Joonis 3) külje testi tulemusi näeme, et tulemused on peaaegu identsed peegelpildid teineteisest. Samuti on näha, et servomootorite kiirused ja suund on mõjutatud mõlema mootori *PWM* signaali kombinatsioonist terve testi ulatuses.

Testi tulemustest (Joonis 2), (Joonis 3) oli võimalik kätte saada *DC* kombinatsioonid, millega saaks robotautot juhtida neljas erinevas suunas efektiivse liikumiskiirusega:

* Otse: vasaku poole *DC* = 95 ja parema poole *DC* = 30
* Tagurdamine: vasaku poole *DC* = 0 ja parema poole *DC* = 60
* Kohapeal paremale: vasaku poole *DC* = 100 ja parema poole *DC* = 77
* Kohapeal vasakule: vasaku poole *DC* = 94.9 ja parema poole *DC* = 50

Joonis . Esimene katse: vasaku külje liikumise test

Joonis . Esimene katse: parema külje liikumise test

## Servomootorite liikumise teine katse

Esimeses katses leidsime ühe viisi, kuidas robotauto servomootoreid juhtida, aga see viis pole kõige efektiivsem, kuna mootorid on eriti müratekitavad ja vahepeal tekib liikumises anomaaliaid (suunamuutusi). Nendel põhjustel oli vaja teha veel katseid, et leida põhjus mootorite juhtimise probleemile. Selleks ühendasime robotauto mootorite toite toiteplokiga, et näha voolu tarvet ja pinge muutusi.

Testides esimeses katses leitud *DC* väärtustega mootorite suunda ja kiirusi tuli välja, et suunad ja kiirused olid täielikult muutunud. Samuti testimise käigus avastasime, et kui puutuda *PWM* signaali toiteallikat (metallkorpusega sülearvutit), siis robotauto servomootorid hakkasid kiiremini liikuma. Sellest tuli järeldus, et servomootorite juhtimisel on pinge probleemid. Pinge probleemi lahenduseks sai akupankade kasutamine, kus servomootoreid toitva akupanga maandus ühendati ka *BBB* miniarvutile.

Nüüd kui servomootoreid sai teineteisest sõltumatult mõjutada, oli vaja teha uued testid, kiiruste ja suundade mõõtmiseks *DC* väärtustel. Selleks oli vaja robot uuesti ümber ehitada (Lisa 1) ja eelnevalt testida, kas juhtimise probleem on täielikult eemaldatud. Selleks vaatame vasaku külje *DC* väärtusi, suurendades ühe võrra neid nullist sajani ja parema *DC* on konstant 0.

Joonis . Teine katse: vasaku külje liikumise test

Jooniselt 4 on näha, et *DC* väärtused vahemikul 0–92 on vasaku servomootori kiirus negatiivne ja 93–100 vahemikul positiivne. Samuti näeme, et parema külje kiirus testi jooksul ei muutunud rohkem kui ±1, mis on lubatud mõõtmise viga.

Nüüd kui on teada, et servomootorid töötavad korrektselt, saame mõõta mõlema robotauto poole kiiruseid iga *DC* väärtuse korral. Selles testis anname mõlemale poolele sama *DC* väärtuse suurendades nullist sajani.

Joonis . Teine katse: parema ja vasaku külje liikumise test

Jooniselt 5 saame kätte parema külje edaspidi liikumise ja vasaku külje tagurpidi liikumise *DC* väärtused. Samuti saame piirkonna 92.56–93.35, kus on vaja veel üks test teha, mis annaks täpsema *DC* vahemiku, millega juhtida paremat külge vastupäeva ja vasakut päripäeva.

Joonis . Teine katse: parema ja vasaku külje liikumise test vastupidises suunas

Jooniselt 6 on näha, et vasaku ja parema külje algsed kiirused on erinevad. Vasaku külje kiirused on alguses väiksemad kui paremal küljel. Testi käigus oli samuti näha, et vasaku külje esimestel lugemistel ei liikunud servomootorid konstantselt ühes suunas. Nendel põhjustel ei kasutata vasaku külje mootorite juhtimisel *DC* väärtusi vahemikus 92.56 kuni 92.66.

Uute testi tulemustest (Joonis 5), (Joonis 6) saame paremad *DC* väärtused kui esimeses katses, kuna nüüd on võimalik teha Pythoni funktsioon (Lisa 3), mis suudab peale stabiilse suuna ka kiirust ühtlaselt hallata. Testi tulemustest (Joonis 5), (Joonis 6) saame järgnevad *DC* vahemikud:

* Vasaku külje mootorid otse: 92.67 – 93.17
* Vasaku külje mootorid tagurpidi: 5 – 39
* Parema külje mootorid otse: 5 – 39
* Parema külje mootorid tagurpidi: 92.56 – 93.06

Teise katse tulemuseks on täielikult juhitavad servomootorid. Kasutades Pythoni funktsiooni (Lisa 3), saab mootoreid juhtida erineva kiirusega nii otse kui ka tagurpidi ilma, et robotauto külgede erinevad *DC* väärtuste teineteisest mõjutaksid.

# Ultraheli kaugusmõõtjate kasutamine

Selles töös on kasutusel kaks SRF08 ultraheli kaugusmõõtjat (Joonis 7), mille mõõtmisvahemik on 3cm–6m. Kaugusandureid kasutab robotauto selleks, et heliallika poole liikudes jääks robot pidama, kui tema tee on takistatud [7].



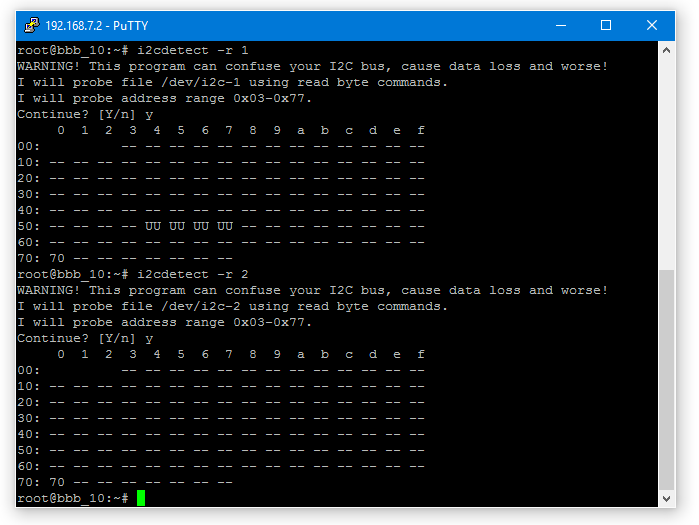
Joonis . SRF08 ultraheli kaugusmõõtja [8]

Ultraheli kaugusmõõtja kasutab *BBB* miniarvutiga suhtlemiseks *I2C* suhtlusprotokolli. *I2C* suhtlus vajab kahte viiku *SDA* ja *SCL*. Kuna kasutame kahte kaugusmõõtjat, siis vajame *BBB* kahte viigupaari (siini), milleks on P9.19 – P9.20 ja P9.21 – P9.22 (Joonis 1). Vaikimisi on aktiveeritud ainult P9.19 – P9.20 viigud (Joonis 1). P9.21 – P9.22 (Joonis 1) viigud tuleb programmiliselt aktiveerida (Joonis 8).

echo BB-I2C1 > /sys/devices/bone\_capemgr.9/slots

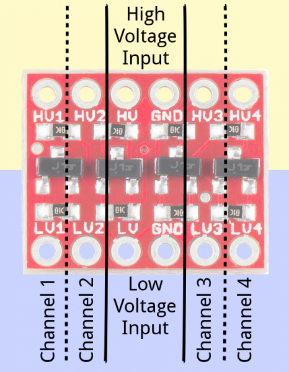
Joonis . Teise I2C seadme aktiveerimine

Selleks, et suhelda anduritega kasutades *I2C* protokolli, peame teadma andurite aadresse. Aadressid saame BBB miniarvuti käest küsida peale andurite ühendamist, kasutades käsku „i2cdetect -r“. Kasutusel on 2 andurit, seega peame küsima iga siini jaoks ühe aadressi. Jooniselt 9 näeme, et esimese ja teise siini aadress on 0x70. Selleks, et küsida anduritelt kauguse väärtust sentimeetrites, peame kirjutama saadud aadressitele väärtuse 0x51. Väärtuste lugemine toimub Pythoni funktsiooniga „GetSonarSensorData()“ (Lisa 4) [7], [9].



Joonis . I2C siinide aadressid

Andureid ei saa ühendada otse miniarvutile, kuna *BBB* miniarvuti *I2C* ei tööta sama pingega. Kahesuunaline loogilise taseme konverter (Joonis 10) laseb BBB poolt saadud 3.3 volti konverteerida viieks voldiks ja ultraheli kaugusmõõtja viis volti konverteerida 3.3 voldiks. Ultrahelianduri ühendused lähevad kõrgema pinge sisendisse (inglise keeles *High Voltage Input*) (Joonis 10) ja BBB ühendused lähevad madalama pinge sisendisse (inglise keeles *Low Voltage Input*) (Joonis 10) [10].



Joonis . Kahesuunalise loogilise taseme konverter [10]

# Heliallika leidmine

Esialgselt oli plaanis heliallika leidmiseks kasutada ainult kahte mikrofoni, aga testimise käigus tuli ilmseks, et robotauto ei tee vahet heliallikal, mis asub täpselt ees või täpselt taga. Seega kasutatakse selles töös nelja ADMP401 mikrofoni (Joonis 11).

Mikrofonid on ühendatud *BBB* miniarvuti analoogsisenditesse P9.40, P9.39, P9.38 ja P9.37, mis omavad nii oma maandust P9.34, kui ka 1.8V toidet P9.32 (Joonis 1).

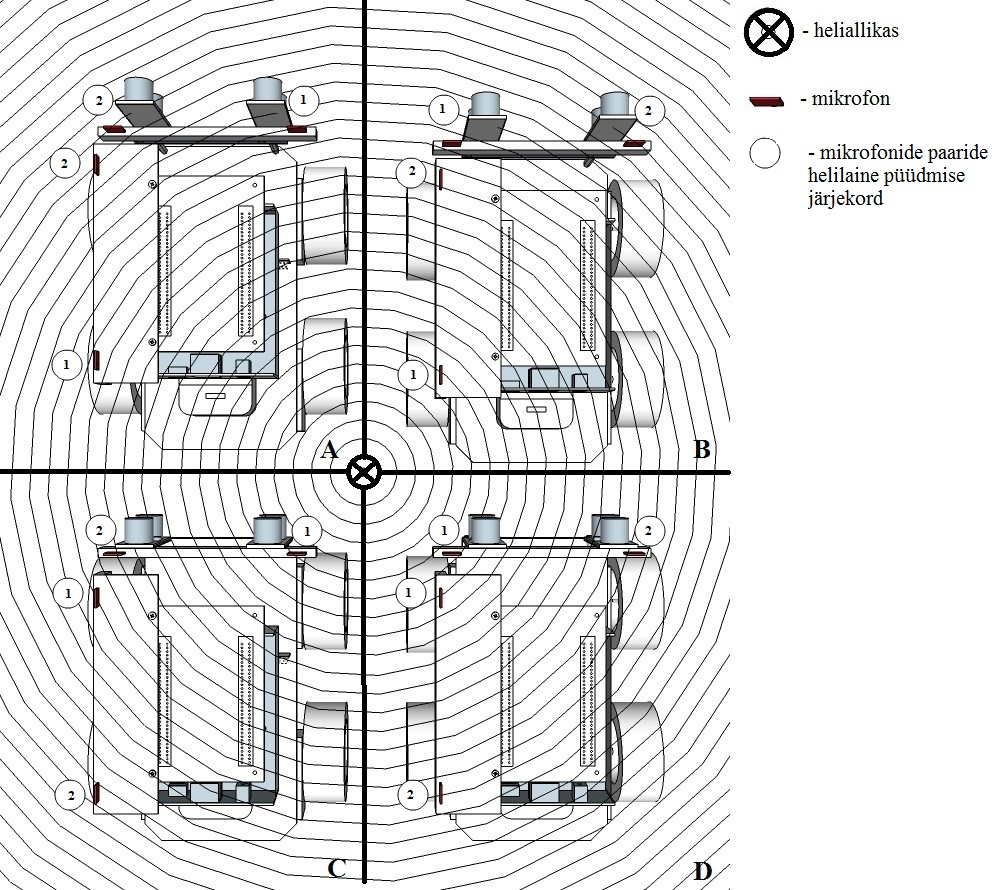


Joonis . INMP401 (ADMP401) mikrofon [11]

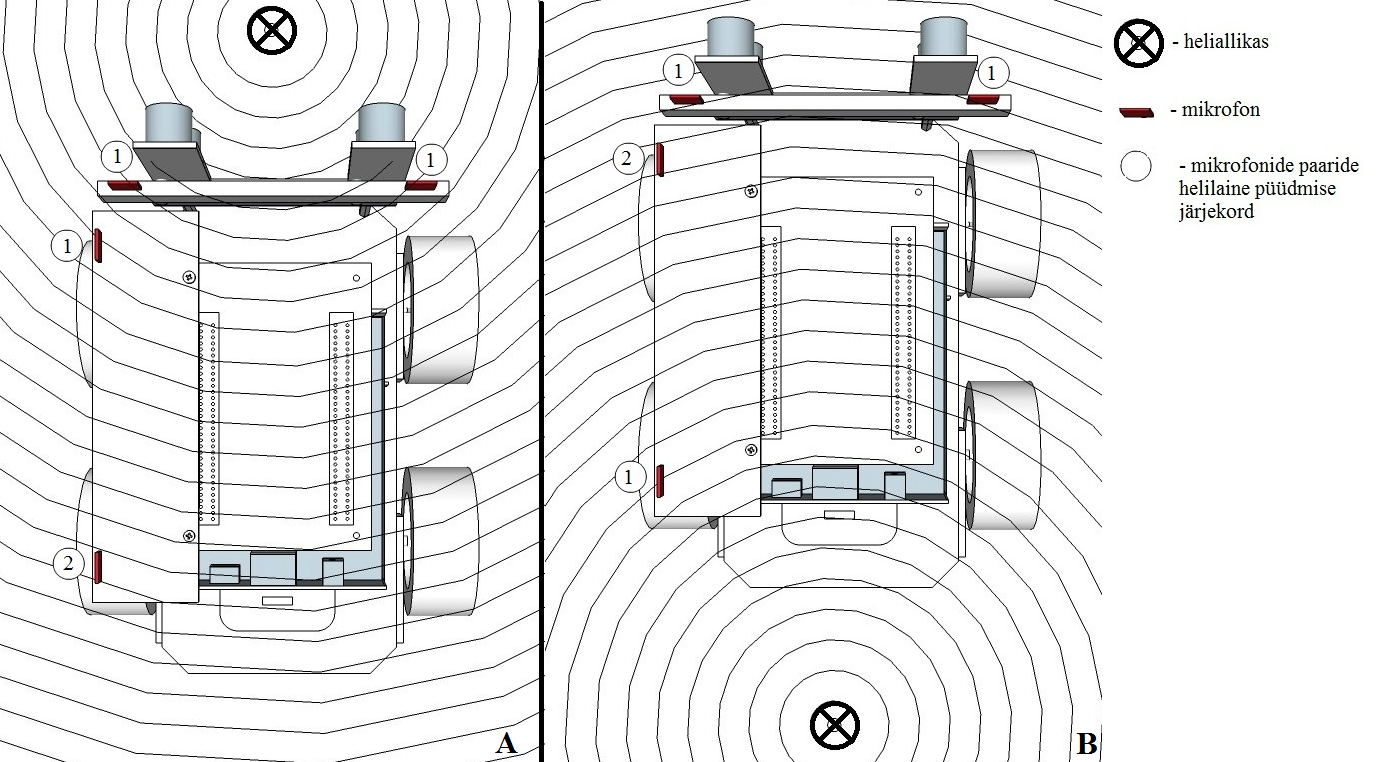
Heliallika positsioneerimise algoritm, mida selles töös kasutatakse, on kirjutatud C keeles Tallinna Tehnikaülikooli Proaktiivtehnoloogiate Teaduslabori poolt.

Heliallika positsioneerimine käib läbi heli hilistumise kahe mikrofoni vahel. Selleks on teada vaja mikrofonide kaugust teineteisest, milleks on 10cm. Heliallika positsiooni mikrofoni paari keskpunkti suhtes saame, kui mõõdame helilaine liikumise aega kahe mikrofoni vahel ehk leiame kui palju hilineb helilaine teise mikrofoni jõudmisel pärast esimesest mikrofoni, mille tulemuseks saadakse heliallika nurk vahemikus -83 o – 83o (algoritmi väljundi täpsus on piiratud mikrofonide vahelise kauguse ja helivõendi sagedusega).

Kasutame kahte mikrofoni paari, millest esimene (robotauto ees) mõõdab heliallika suunda (vasak/parem) ja teine (robotauto vasakul küljel) mõõdab kas heliallikas on robotauto ees või taga. Joonistelt (Joonis 12), (Joonis 13) näeme, mis järjekorras, sõltuvalt heliallika asukohast, mikrofonid helilaineid registreerivad. Kui heliallikas on otse ees või taga, registreerivad robotauto ees olevad anduri samal ajal helilaine, ja sellepärast on vaja roboti küljel olevat mikrofonipaari, mille järgi saab teada, kas heli tuli eest või tagant. Näiteks, kui helilaine tuleb tagant (Joonis 13.B), siis külje tagumine andur registreeris helilaine enne, kui külje esimene andur.



Joonis . Heliallika positsioneerimine: heliallikas- (A) taga paremal (B) taga vasakul (C) otse paremal (D) otse vasakul



Joonis . Heliallika positsioneerimine: heliallikas- (A) ees keskel (B) taga keskel

Olemasolev C heliallika positsioneerimise kood väljastab hetkel ainult kaks väärtust: ees oleva anduripaari heliallika asukoha nurga ja küljeandurite poolt saadud nurga. Et robotauto saaks helile järgneda ilma, et tema servomootorite müra segaks roboti heliallika leidmist, on vaja leida helilävi, millest alates hakkaks robotauto helile reageerima. Helitugevust antud C koodist saab vaadata kolme muutujaga: normeeritud väärtusega, väärtuste summaga ja keskväärtusega. Et leida õige muutuja, millega oleks kõige efektiivsem heliläve tekitada, tuleb teha testid, kus loeme sisse kõik kolm väärtust, samal ajal suurendades heli tugevust.

Testi tulemustest (Joonis 14), (Joonis 15), (Joonis 16) näeme, et normeeritud väärtuse muutujat on kõige parem kasutada heliläve tekitamiseks, kuna vaikuses on muutuja stabiilselt väike ja mida kõrgem on heli, seda rohkem muutuja helile reageerib. Teistest muutujatest on raske leida kohta, kus heliallikas hakkas tööle, mida normeeritud väärtuse muutujast on kerge leida.

Joonis . Heliallika tugevus: keskväärtus

Joonis . Heliallika tugevus: normeeritud väärtus

Joonis . Heliallika tugevus: väärtuste summa

Nüüd, kui on teada muutuja, millega hakkame piirama robotauto kuulmist, saame leida heliläve. Kasutame mõlema mikrofonipaari normeeritud väärtusi heliläve loomisel, kuna nii saame kahekordse helitugevuse kontrolli, kus küljeandurid registreerivad mootorite müra taset efektiivsemalt, kui ees olev anduripaar, mida on ka näha järgnevast testi tulemusest (Joonis 17). Et leida piirkond, kus mootorite müra oleks suuremalt osalt eemaldatud, teeme kolm testi: vaikuses, servomootorite müraga ja servomootorite müra koos heliallikaga. Tulemuseks (Joonis 17) saime piirkonna märgitud punase kastiga, kus on ära kaetud 88.2% mootorite mürast:

* Ees oleva mikrofonide helitugevuse vahemik: 0.015381– -0.015137
* Vasaku külje mikrofonide helitugevuse vahemik: 0.004883 – -0.015137

Joonis . Heli tugevuse mõõtmised

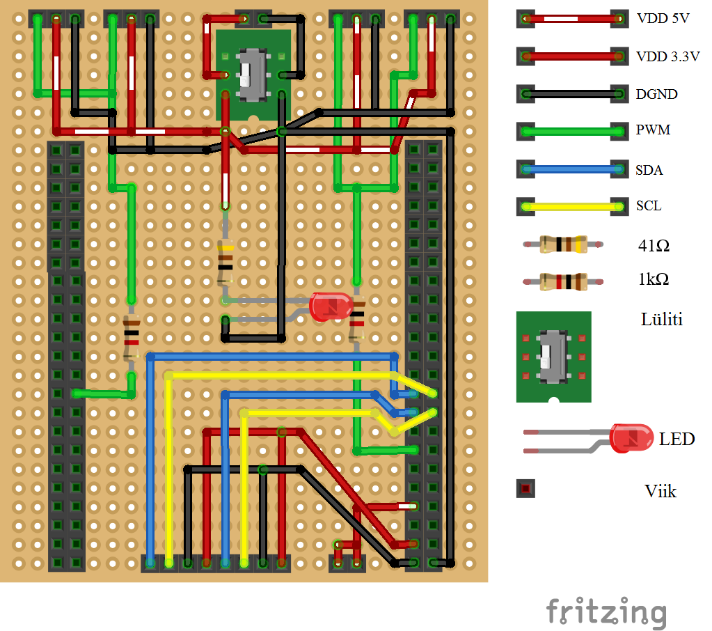
Teades muutujat, mida hakkame kasutama robotauto helile reageerimise piiramiseks, on vaja antud muutuja koos nurga väärtustega programmist väljastada. Pärast C koodi modifitseerimist on programmi väljund järgnev: ees olevate mikrofonide helitugevus (normeeritud väärtus) koos heliallika nurgaga ja vasaku külje mikrofonide helitugevus (normeeritud väärtus) koos heliallika nurgaga.

# Robotauto komponentide ühendused

Algselt olid kõik komponentide vahelised ühendused otseviikudega (Lisa 1 ja Lisa 2), aga need ühendused ei olnud stabiilsed ja kaotasid pidevalt kontakti viikudega. Samuti tänu seadmete suurele arvule oli juhtmete arv suur, eriti toite ja maanduse juhtmed, mida pidi mitme seadme vahel jagama. Lahenduseks oli vaheplaadi (Joonis 18) tegemine, mis eemaldas suurema osa juhtmeid ja lasi teha stabiilsemaid kinnitusi.

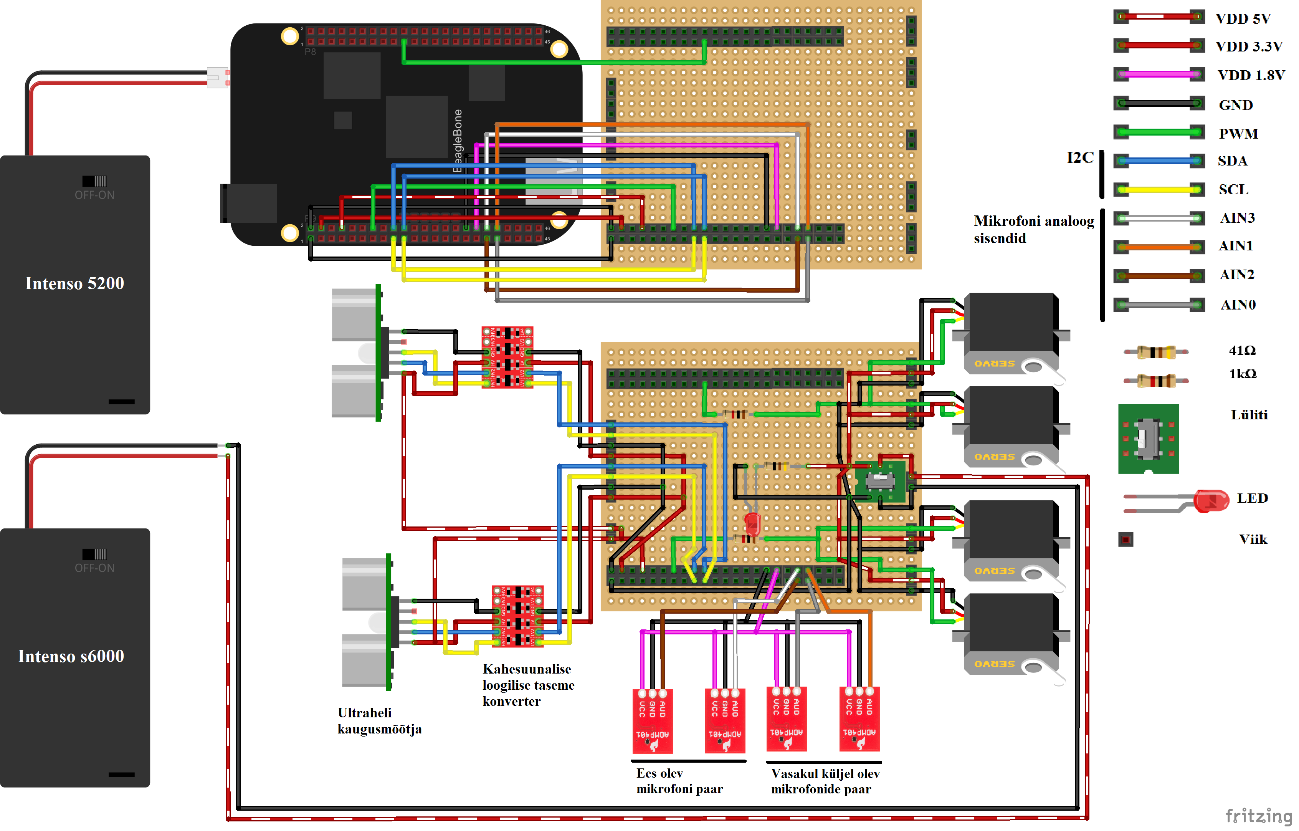
Ühendamisel tuleb arvestada, et servomootorite *PWM* signaali vahele on soovitatav panna 1kΩ takistused, et kaitsta *BBB* miniarvuti *PWM* viiku pingekõikumise eest, mida servomootori koormuse äkiline tõus või lang tekitada võib. Samuti on kasulik servomootorite ja akupanga vahele panna üks valgusdiood koos 41Ω takistusega, mis ei laseks akupangal end välja lülitada mootorite seisuoleku ajal. Samuti aitab mootorite akupanka sees hoida selle maandamine *BBB* miniarvutiga. [12].

Trükkplaadile (Joonis 18) on joodetud nelja servomootori neli kolmest viiku, kahe ultrahelianduri üks kaheksane ja üks kahene (kahesuunalise loogilise taseme konverterile 5V, pinge tõstmiseks) viik. Samuti on lisatud vajalikud takistused, lüliti mootori toite kontrollimiseks, valgusdiood mootorite sees hoidmiseks seisuoleku ajaks, ja on tehtud kõik vajalikud pinge ja maanduse jagamised erinevate seadmete vahel.



Joonis . Trükkplaadi disain

Kõik robotauto komponentide ühendused on näha joonisel 19. Mikrofonide ühendusi ei ole trükkplaadil eraldi viikudesse veetud, kuna mikrofonide asukohad ja kinnitused (Lisa 6) vajasid pikemate juhtmete vedamist trükkplaadi all, mida ei oleks saanud trükkplaadile efektiivsemalt panna, kui otseühendusega.



Joonis . Robotauto komponentide ühendused

Nüüd, kui kõik komponendid on olemas ja trükkplaat on valmis, saame robotauto lõplikult kokku panna (Lisa 7).

# Robotauto juhtimise algoritm

Robotauto liikumist juhitakse Pythoni programmiga (Lisa 5), kuna Python on lihtne ja mugav keel erinevate andmetüüpide töötlemiseks. Töötlemist vajavad nii C programmist saadud heliallika nurgad ja normeeritud muutuja väärtused kui ka ultrahelianduritest saadud väärtused.

Pythoni programmiga pannakse mikrofonide C-programm taustaprotsessina tööle ja suunatakse selle väljundid Python programmi, mis antud andmed ära töötleb ja hakkab saadud informatsiooni põhjal otsuseid tegema.

C programmis oli vaja lisaks andmete väljaprintimisele modifitseerida koodi, et saada teada vigastest lugemistest või programmis tekkinud veast, mis vajab programmi uuesti käivitamist. Selleks on koodis kaks veateate väljundit, millest esimene „*WARNING*“ annab märku, et mikrofonide lugemisel läks midagi valesti ja Python programm peab uuesti andmeid küsima ja „*PRUIO failed!*“, mis teavitab seda, et C programm tuleb sulgeda ja uuesti käivitada.

Python küsib kaks korda heliallika asukoha nurga väärtusi ja siis võtab nendest keskmise, saades stabiilsema suuna. Kui võtta järjest lugemisi, võib mõni muu ajutine heliallikas robotauto suunda ühekordselt mõjutada ja tagajärjeks on roboti jõnksutamine parema ja vasaku suuna vahel. Nüüd saame kasutada C programmist saadud normeeritud muutuja väärtusi ja eelnevalt leitud helitaseme piirkonda (Joonis 17), mida mitte jälgida, et vaadata, kas sellele heliallikale peab robotauto reageerima või mitte (Joonis 20). Kui heliallikas on liiga vaikne siis robotauto ei liigu, aga kui heliallikas on valjem kui ette antud väärtused, siis hakkab robotauto seda järgima.

#Front microfones

if 0.015381 >= soundLevelFront and soundLevelFront >= -0.015137:

#Left microfones

if 0.004883 >= soundLevelSide and soundLevelSide >= -0.015137:

sound=False #Don't follow the sound

else:

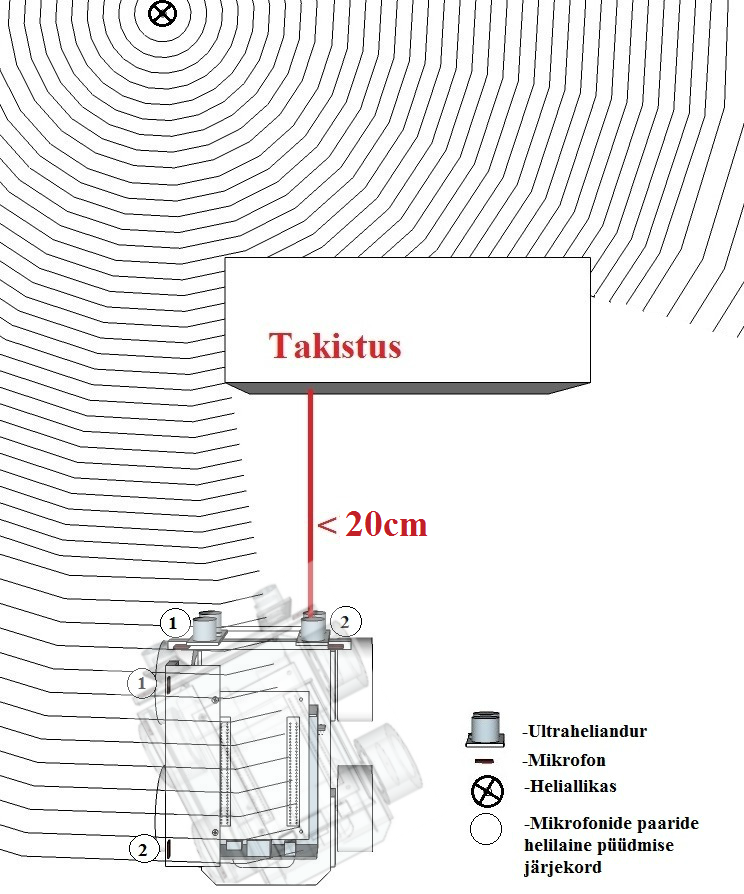
sound=True #Follow the sound

else:

sound=True #Follow the sound

Joonis . Heli tugevuse piiramine

Kui robotauto leiab, et heliallikas on küllalt tugev, et sellele järgneda, vaatab ta ees oleva mikrofoni anduripaari nurga väärtust. Kui nurk on positiivne ja suurem/võrdne kui 18 siis heli tuleb paremalt ja robot pöörab kohapeal end paremale, kui nurga väärtus on negatiivne ja väiksem/võrdne kui -18 siis robot pöörab kohapeal vasakule. Robotauto teab, et heliallikas on otse, kui ees oleva mikrofonipaari nurga väärtus jääb vahemikku -18 – 18 ja vasaku külje mikrofonipaari nurga väärtus on positiivne. Kui vasaku külje mikrofonipaari nurga väärtus on aga negatiivne, siis tähendab see seda, et heliallikas on robotauto taga ja robot peab end ümber pöörama. Ümberpööramise suunda valitakse ees oleva mikrofoni anduripaari nurga järgi. Kui väärtus on suurem/võrdne kui -1, siis pöörab vasakule, kui väärtus on väiksem nullist, pöörab robot paremale. Otse sõitmisel jälgib robot ultraheli kaugusmõõtjatelt saadud kaugusi. Kui kumbki kaugusandur kasutades Pythoni funktsiooni „GetSonarSensorData()“ (Lisa 4) näeb, et takistus on lähemal kui 20cm, siis ta edasi ei sõida ja jääb ootama uue (algne heliallikas on vaikne) või valjema heli (algne heliallikas ei ole vaikne) teket. Tänu sellele, et mikrofonid on ultraheliandurite kõrval saame ära kasutada neid takistustest mööda saamiseks. Kui üks kaugusandur näeb takistust siis robot ei tohi enam edasi sõita, aga võib kohapeal end tugevama heliallika poole pöörata, kuna teel olev takistus summutab helilaineid ja roboti jaoks ei ole enam heliallikas otse saab robot enda suunda muuta vastavalt heli tugevusele (sõltub heliallika asukohast takistuse taga). Jooniselt 21 näeme, et kui takistus on ees ja heliallikas on takistuse taga vasakul pool siis roboti vasakpoolne mikrofon registreerib heli ennem, kui takistuse taga olev mikrofon, mille tagajärjel robot pöörab vasakule takistusest mööda. Kui suuna otsus on tehtud, antakse mootori kiirused ja suunad mootorite funktsioonile (Lisa 3).



Joonis . Takistuse vältimine

Selleks, et robotauto käivitaks automaatselt toite saamisel Pythoni programmi, peame andma *BBB* miniarvutile käsu, et ta selle programmi käivitaks *BBB* käivitumise ajal. Käivitamisel tekkis aga probleem. Python programm käivitati siis, kui BBB polnud veel kõiki oma protsesse käivitanud, ja tööle pandud Python programm jooksis kokku veateatega. Selle probleemi lahenduseks sai pandud koodi algusesse enne mootorite, andurite deklareerimisi ja C koodi käivitamist, et kood ootaks 60 sekundit enne edasi minekut. 60 sekundit on piisav aeg *BBB* täielikuks käivitumiseks.

# Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli ehitada ja programmeerida robotauto, mis on võimeline leidma ruumis oleva liikuva või seisva heliallika ja sellele järgneda, kuniks robotauto tee pole takistatud. Heliallika leidmiseks kasutatakse ProLabi poolt koostatud C programmi.

Nagu töös oli näha, oli terve ehitamise ja programmeerimise protsessi käigus erinevaid väljakutseid, millele sai töö käigus leida lahendusi. Probleeme oli nii riistvara kui ka tarkvara valdkondadest, mootorite liikuma panemisest kuni kahe programmi vahelise suhtlemiseni.

Selle bakalaureusetöö käigus valmis töötav robotauto, mis on võimeline leidma ruumis oleva liikuva või seisva servomootorite mürast tugevama heliallika ja sellele järgneda, kuniks robotauto tee pole takistatud. Robotauto peal jooksevad samaaegselt kaks programmi: heliallika positsioneerimise C-programm ja roboti juhtimise Python-programm.

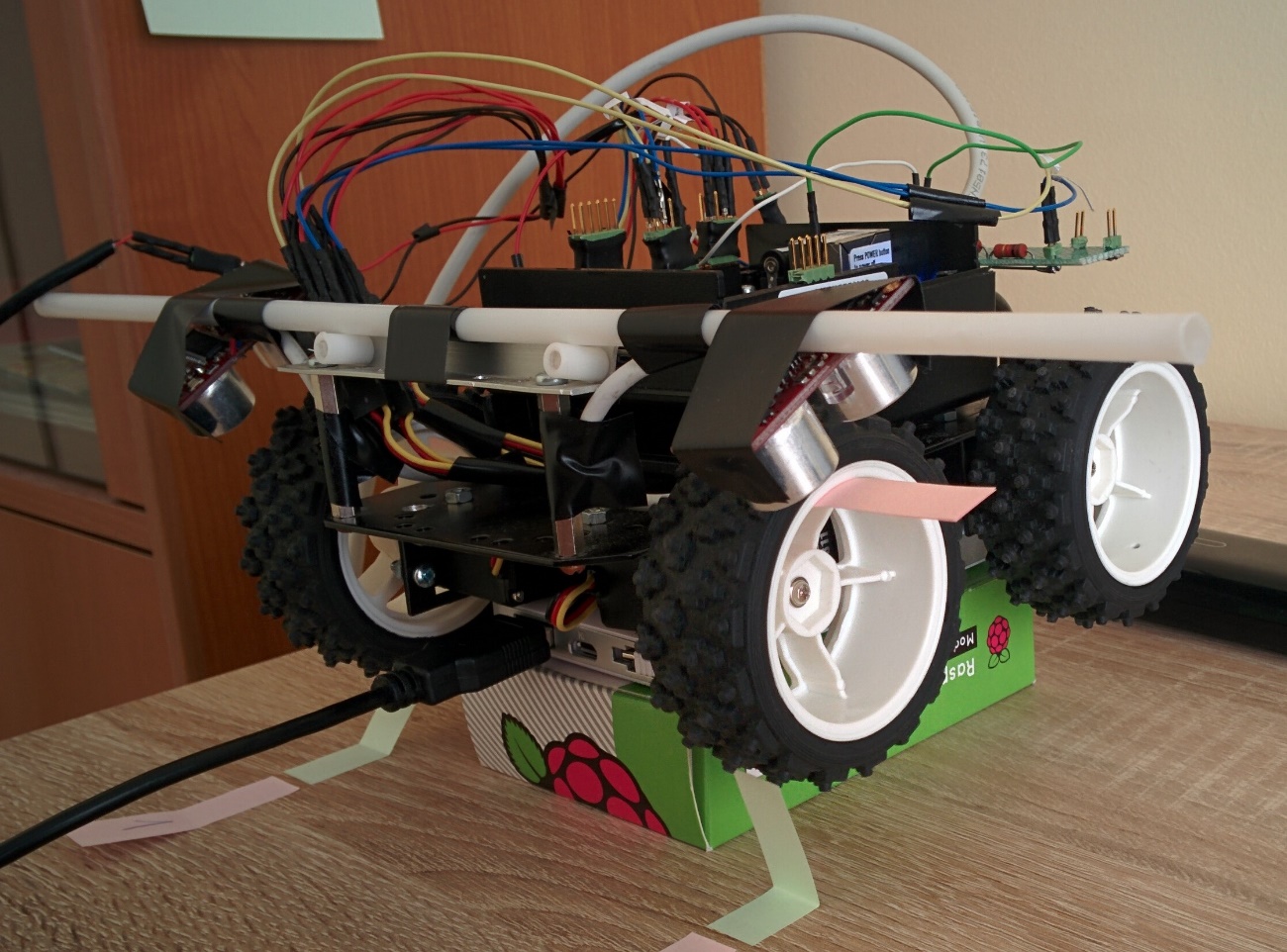
Käesoleva bakalaureusetöö edasiarendamisel saaks robotauto panna jälgima spetsiifilisi helisid, mitte lihtsalt kõige valjemat, ja samuti saaks leida vaiksemad mootorid, mis annaksid suurema helitugevuse piirkonna, mida robotauto saaks jälgida. Huvitavaks lahenduseks oleks veel võimalik lisada robotile kaamera, millega suudab robot heliallika juurde sõita ka siis, kui heliallikas on vaikseks jäänud, kasutades mikrofone, et leida kaamerast saadud pildil olev heliobjekt.

Lõputöö autorile andis tehtud töö palju uusi teadmisi Linux operatsioonisüsteemi, *BBB* miniarvuti kui ka teiste roboti komponentide kasutamise kohta. Sai õppida Pythoni programmeerimise keelt kui ka C keele teadmisi värskendada. Samuti sai suures koguses jootmise ja trükkplaadi disainimise kogemusi.

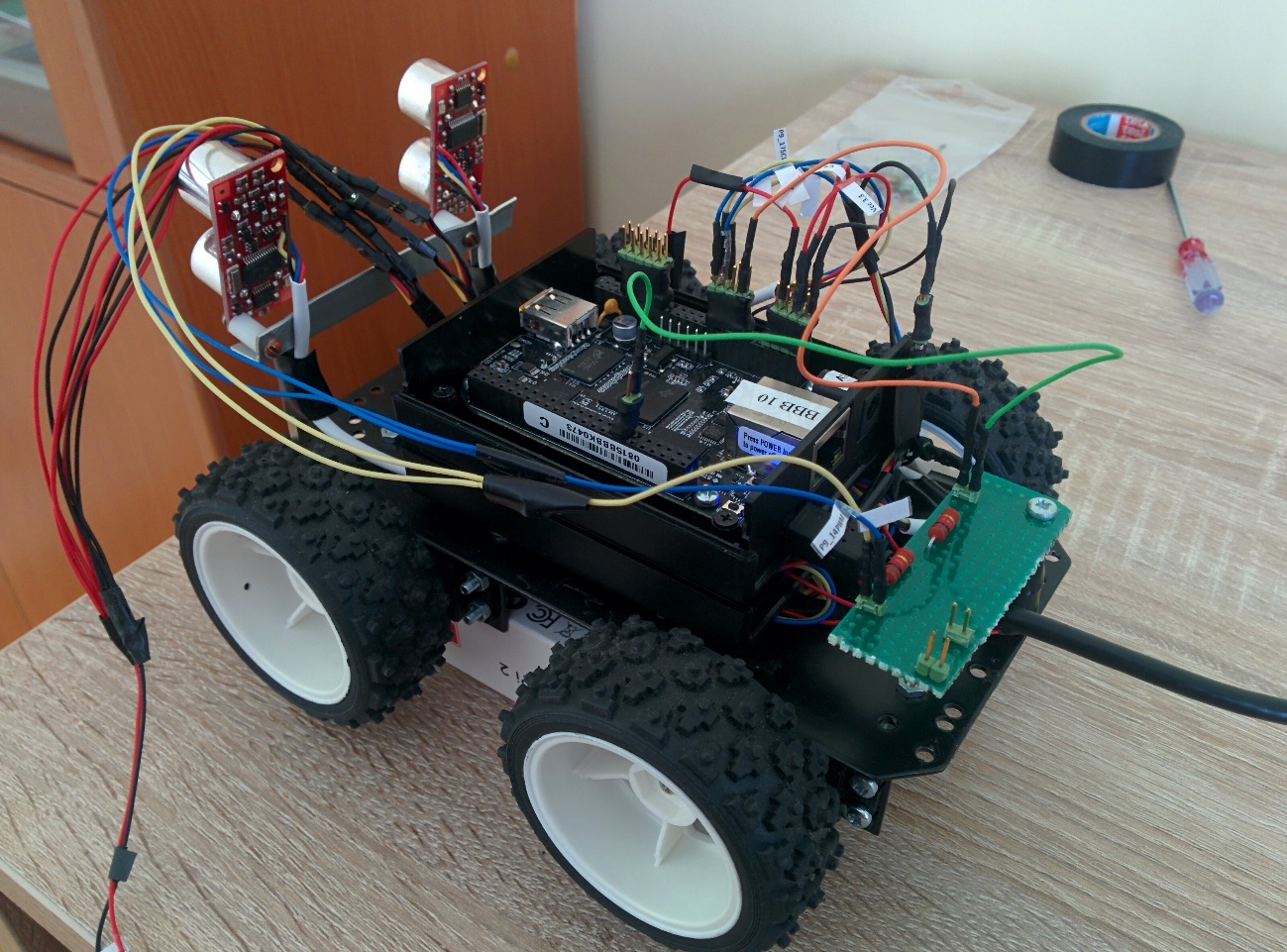
Kasutatud kirjandus

1. BeagleBone Black [WWW] https://beagleboard.org/black (18.05.2016)
2. BeagleBone 101 [WWW] http://beagleboard.org/support/bone101 (18.05.2016)
3. HS-311 Standard [WWW] <https://www.servocity.com/html/hs-311_standard.html#.V1AixJGLS71> (30.05.2016)
4. BeagleBone Black Pins [WWW] http://robotic-controls.com/node/68 (30.05.2016)
5. only 5 PWM pins possible at a time [WWW] https://github.com/adafruit/adafruit-beaglebone-io-python/issues/31 (19.05.2016)
6. General Servo Information [WWW] http://www.horizonhobby.com/pdf/HRC31311S-Servo\_Manual.pdf (18.05.2016)
7. Interfacing the BeagleBoard with an SRF08 Ultrasonic Ranger over I2C [WWW] https://yetanotherhackersblog.wordpress.com/2011/04/07/interfacing-the-beagleboard-with-an-srf08-ultrasonic-ranger-over-i2c/ (18.05.2016)
8. TruSens SRF08 High Performance Ultrasonic Range Finder [WWW] http://www.rapidonline.com/electronic-components/trusens-srf08-high-performance-ultrasonic-range-finder-78-1086 (23.05.2016)
9. SRF08 Ultra sonic range finder Technical Specification [WWW] http://www.robot-electronics.co.uk/htm/srf08tech.html (23.05.2016)
10. Bi-Directional Logic Level Converter Hookup Guide [WWW] https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide (18.05.2016)
11. SparkFun MEMS Microphone Breakout - INMP401 (ADMP401) [WWW] https://www.sparkfun.com/products/9868 (22.05.2016)
12. Controlling a Servo with a BeagleBone Black [WWW] https://cdn-learn.adafruit.com/downloads/pdf/controlling-a-servo-with-a-beaglebone-black.pdf (22.05.2016)

Lisa 1 – Robotauto ümberehitus, et mõõta servomootorite kiirust



Lisa 2 – Robotauto ühendused enne vaheplaati



Lisa 3 –Servomootorite liikumise Python funktsioon

def MotorSpeed(side, Speed): #Input motor side and speed 0–100

if side=="Left":

#(x-in\_min)\*(out\_max-out\_min)/(in\_max-in\_min)+out\_min

if(Speed>0): #Forward

P8\_DC=(Speed-1)\*(93.17-92.67)/(100-1)+92.67

elif (Speed<0): #Reverse

Speed=Speed\*-1

P8\_DC=(Speed-1)\*(39-5)/(100-1)+5

else: #Stop

P8\_DC=100

P8\_DC = float("{0:.2f}".format(P8\_DC))

PWM.set\_duty\_cycle(P8, float(P8\_DC))

if side=="Right":

if(Speed>0): #Forward

P9\_DC=(Speed-1)\*(39-5)/(100-1)+5

elif (Speed<0): #Reverse

Speed=Speed\*-1

P9\_DC=(Speed-1)\*(93.06-92.56)/(100-1)+92.56

else: #Stop

P9\_DC=100

P9\_DC = float("{0:.2f}".format(P9\_DC))

PWM.set\_duty\_cycle(P9, float(P9\_DC))

Lisa 4 – Ultraheli kaugusmõõtjatest kauguse saamise Python kood

bus1 = smbus.SMBus(1) # pins D9 19, 20, /dev/i2c-1

bus2 = smbus.SMBus(2) # pins D9 17, 18, /dev/i2c-2

rng\_left = 0 #Sensor1 left distance value

rng\_right = 0 #Sensor2 right distance value

def write\_byte(value):

bus1.write\_byte\_data(0x70, 0, value)

bus2.write\_byte\_data(0x70, 0, value)

return -1

def sonicrange1():#Get left distance value

range1 = bus1.read\_byte\_data(0x70, 2) #High Byte

range2 = bus1.read\_byte\_data(0x70, 3) #Low Byte

range3 = (range1 << 8) + range2

return range3

def sonicrange2():#Get right distance value

range1 = bus2.read\_byte\_data(0x70, 2) #High Byte

range2 = bus2.read\_byte\_data(0x70, 3) #Low Byte

range3 = (range1 << 8) + range2

return range3

def GetSonarSensorData():

global rng\_left

global rng\_right

write\_byte(0x51)

time.sleep(0.1)

rng\_left = sonicrange1()

rng\_right = sonicrange2()

return 1

Lisa 5 –Python põhitsükkel

while True:

soundAngleAVR, soundDirectionAVR = 0, 0

#Monitore volume level (Follow sounds that are louder than servos)

for i in range(2): # Get 2 volume readings and add them together

getMicData()

soundDirectionAVR = soundDirectionAVR + soundDirection

soundAngleAVR = soundAngleAVR + soundAngle

soundAngle = soundAngleAVR / 2

soundDirection = soundDirectionAVR / 2

GetSonarSensorData() # Get ultrasonic data

if 0.015381 >= soundLevelFront and soundLevelFront >= -0.015137: # Front

sound = soundLevelSide > 0.004883 or soundLevelSide < -0.015137

else:

sound = True # Follow the sound

if not sound: # If volume level is lower than servos then STOP

leftSpeed, rightSpeed = 0, 0

else: # Else follow sound

if soundAngle >= 18: # Turn right on the spot

leftSpeed, rightSpeed = 100, -100

if soundAngle <= -18: # Turn left on the spot

leftSpeed, rightSpeed = -100, 100

if -18 < soundAngle < 18: # straight forward

if soundDirection > 0: # Check if sound is coming from front

if rng\_right < 20 or rng\_left < 20: # Is path obstructed?

leftSpeed, rightSpeed = 0, 0

else:

leftSpeed, rightSpeed = 100, 100

else:

if soundAngle < 0:

leftSpeed, rightSpeed = -100, 100

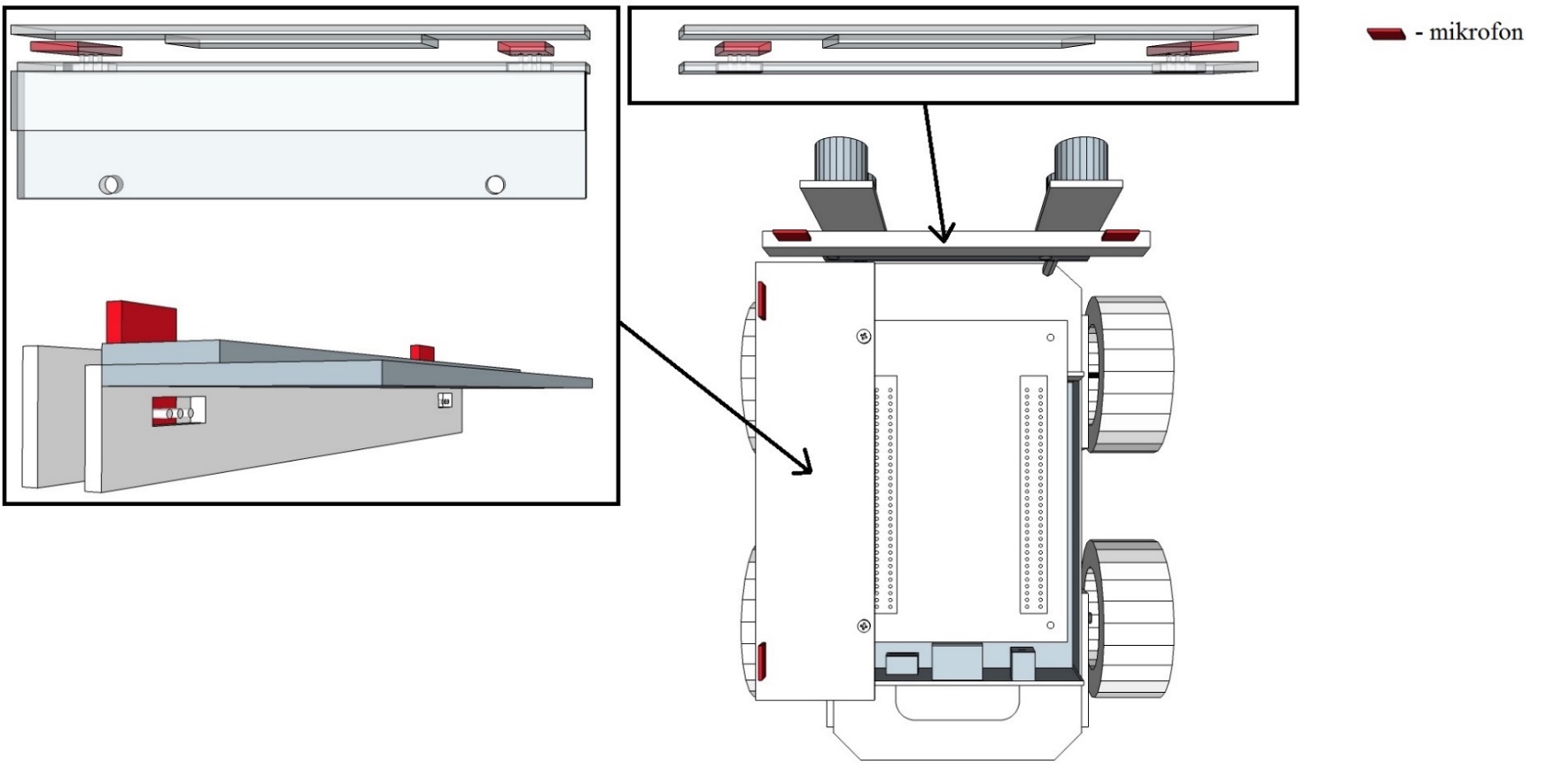
elif soundAngle >= -1:

leftSpeed, rightSpeed = 100, -100

MotorSpeed("Right", rightSpeed) # Change right motor speed

MotorSpeed("Left", leftSpeed) # Change left motor speed

Lisa 6 – Robotauto mikrofonide kinnitused



Lisa 7 – Valmis robotauto

