TAMPEREEN TEKNILLINEN YLIOPISTO

JOUKO KALLIO

HUMANOIDIROBOTTI NAON KÄYTTÖSOVELLUTUKSET VANHUSTEHUOLLOSSA

Diplomityö

Tarkastaja: professori Hannu-Matti Järvinen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 4.kesäkuuta 2014

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
KALLIO, JOUKO: Humanoidirobotti naon käyttösovellutukset vanhustenhuollossa
Diplomityö, 42 sivua, 0 liitesivua
Huhtikuu 2016
Pääaine: Sulautetut Järjestelmät
Tarkastajat: professori Hannu-Matti Järvinen
Avainsanat: Vanhustenhuolto, Nao-robotti, C++, Python, Ohjelmointi, Robotiikka ja Telelääketiede

Opinnäytetyö on tehty osana TEKESin (teknologian ja innovaatioiden kehittämiskeskus) rahoittamaa hoitorobotti-tutkimushanketta, jossa oli tavoitteena selvittää ja pilotoida robotin hyödyntämistä vanhustenhuollossa hoitohenkilökunnan apuvälineenä. Tutkimushankkeessa mukana olivat Tampereen teknillisen yliopiston Seinäjoen yksikön lisäksi LifeIT Oyj, Etelä-Pohjanmaan terveysteknologian kehittämiskeskus ry ja kolme etelä-pohjanmaalaista hoitokotia, jotka olivat Tmi Alavuden Karoliinakoti, Hoivakartano ja Vuosikas Oy. Robottina opinnäytetyössä käytettiin ranskalaisvalmisteista Nao-robottia.

Opinnäytetyössä robotille tehtiin erilaisia ohjelmistoja, joiden avulla robottia voidaan käyttää apuvälineenä vanhusten hoivatyössä. Ohjelmistojen avulla robottia voitiin testata hankkeessa mukana ollessa hoitokodeissa. Robotille tehtiin jumppaohjelma, jota voitaisiin käyttää apuna ikäihmisten kuntouttamiseen. Robotille tehtiin valvontasovellus, joka toimii hoitajien apuna hoivattavien valvonnassa. Valvontasovellusta varten jouduttiin tekemään sisätilanavigointiohjelma, joka avaa ja sulkee puheyhteyden hoivattavan ja hoitajan ja ohjelma joka avaa ja sulkee videoyhteyden hoivattavan ja hoitajan välillä. Ohjelmistoja testattiin robotin oikeassa käyttöympäristössä hankkeessa mukana olleessa hoivakodeissa.

Opinnäytetyössä tehty tutkimus osoitti, että robotiikka voi tarjota hoitohenkilökunnalle hyvän apuvälineen vanhustenhoivatyöhön. Toisaalta on selvää, että robotti ei tule syrjäyttämään ihmistä hoitotyössä, vaan se toimii lähinnä avustajana ja tarjoaa työkalun turvallisuuden lisäämiseen valvontaan ja kulunseurantaan liittyvissä tehtävissä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
KALLIO, JOUKO: Applications of Humanoid robot Nao in elderly care
Master of Science Thesis, 42 pages, 0 Appendix pages
April 2016
Major: Embedded Systems
Examiners: Professor Hannu-Matti Järvinen
Keywords: Elderly care, Nao robot, C++,Python, Programming, Robotics and Telemedicine

Thesis has been written as part of the TEKES (the Finnish Funding Agency for Technology and Innovation) funded care robot research project in which the goal was to define and pilot the robot's use as tool of nursing care for the elderly. In addition to Tampere University of Technology Seinäjoki unit, the research project was participated by LifeIT Ltd, South Ostrobothnia Health Technology Development Centre and three in South Ostrobothnical care homes that were Tmi Alavuden Karoliinakoti, Hoivakartano and Vuosikas Ltd. Robot used in the thesis was Nao robot produced in France.

In the thesis, different software programs were made in order to use the robot as an aid for the elderly care. With these programs the robot could be tested in the nursing homes that were participating the project. For the robot was made an exercise program that could be used to help in the rehabilitation of the elderly. Monitoring application was made intended to help the nurses supervise the residents. Three additional programs were made for the monitoring application: navigation program for indoors, program that opens and closes a voice connection between resident and nurse, and a program that opens and closes video connection between the resident and the nurse. Software were tested in the robot's right operating environment in the nursing homes taking part to the project.

The research made in the thesis showed that robotics can provide a valuable tool for health care personnel caring for the elderly. On the other hand, it is clear that the robot will not displace people in nursing, but it works mainly as an assistant and provides a tool for enhancing the security in tasks concerning supervision and access control.

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

flash-muisti	Flash-muisti on puolijohdemuisti, joka voidaan sähköisesti tyhjentää ja uudelleen ohjelmoida. Flash-muisti on haihtumaton muistityyppi, jossa tieto säilyy jopa 10 vuotta, vaikka virta kytkettäisiin pois. Muistissa ei ole liikkuvia mekaanisia osia, joten se on käytettäessä äänetön.
FTP	FTP (File Transfer Protocol) on TCP(Transmission Control Protocol)-protokollaa käyttävä tiedostonsiirtomenetelmä kahden tietokoneen välille. FTP-yhteys toimii asiakas-palvelin -periaatteella.
GSM	GSM (Global System for Mobile Communications) on matkapuhelinjärjestelmä, jota käytetään maailmanlaajuisesti.
OSX	OSX on Applen kehittämä ja markkinoima käyttöjärjestelmäperhe Macintosh-tietokoneisiin. Se on tullut vuodesta 2002 esiasennettuna jokaisen uuden Macintosh-tietokoneen mukana.
RGB	RGB-värimalli on väritila, jossa eri värejä muodostetaan sekoittamalla keskenään punaisen, vihreän ja sinisen väristä valoa. Värimallin nimi juontuu näiden päävärien englanninkielisistä nimistä: red, green, ja blue.
RTP	RTP (Real-time Transport Protocol) on tietoliikenneprotokolla tosiaikaisen datan kuten äänen ja kuvan siirtoon pakettiverkoissa.
SRAM-muisti	SRAM (Static Random Access Memory) eli staattinen RAM(Random access memory) -muisti on puolijohdetekniikalla toteutettu muistityyppi. SRAM eroaa DRAM(Dynamic Random Access Memory)-muistista siten, että DRAM-muistikennon kondensaattoria on virkistettävä säännöllisesti, koska muutoin se kadottaa tietonsa.
SSH	SSH (Secure Shell) on salattuun tietoliikenteeseen tarkoitettu protokolla. Yleisin SSH:n käyttötapa on ottaa etäyhteys SSH-asiakasohjelmalla SSH-palvelimeen päästäkseen käyttämään toista konetta merkkipohjaisen konsolin kautta. SSH:lla voidaan myös suojata FTP-, HTTP(Hypertext Transfer Protocol)- tai muuta liikennettä, joka toimii samalla tasolla.
TEKES	Tekes on Suomen valtion virasto, joka aktivoi ja rahoittaa yritysten, yliopistojen ja korkeakoulujen sekä tutkimusyksikköjen tutkimus- ja kehitysprojekteja. Tekes on perustettu vuonna 1983. Tekesin vuosibudjetti on noin 500 miljoonaa euroa.

1 JOHDANTO

Työ on tehty osana TEKESin (teknologian ja innovaatioiden kehittämiskeskus) rahoittamaa hoitorobotti-tutkimushanketta, jossa oli tavoitteena selvittää robotin hyödyntämistä vanhustenhuollossa hoitohenkilökunnan apuvälineenä. Tutkimushankkeessa käytettiin ranskalaisvalmisteista Nao-robottia. Tutkimushankeen osapuolina olivat Tampereen teknillisen yliopiston Seinäjoen yksikkö hankeen vetäjänä, LifeIT Oyj, Etelä-Pohjanmaan terveysteknologian kehittämiskeskus ry ja kolme etelä-pohjanmaalaista hoitokotia, jotka olivat Tmi Alavuden Karoliinakoti, Hoivakartano ja Vuosikas Oy.

Työn tavoitteena on laatia Nao-robotille ohjelmistoja, joiden avulla robottia voidaan käyttää apuvälineenä vanhusten hoivatyössä. Tavoitteena on tehdä jumppaohjelma, jota voidaan käyttää vanhusten kuntoutuksessa apuvälineenä. Toisena tavoitteena on tehdä valvontasovellus, jota voidaan käyttää hoitajan apuna hoivattavien valvonnassa erityisesti yön aikana, kun hoitajia on vähän töissä. Valvontasovellusta varten on tarkoitus toteuttaa myös robotille ohjelma, joka hoitaa autonomisen liikkumisen ja navigoinnin sisätiloissa ja ohjelma, joka avaa ja toteuttaa puheyhteyden hoivattavan ja hoitajan välillä.

Työn luvussa kaksi käsitellään vanhustenhuoltoa, väestön ikääntymistä ja vanhustenhuollon kustannuksia. Luvussa kolme esitellään, minkälainen laite käytettävä Nao-robotti on. Luvussa neljä tarkastellaan, minkälainen laite Nao on ohjelmoijan näkökulmasta katsottuna. Luvussa viisi tehdään selkoa, minkälaisia työkaluja on käytettävissä Nao ohjelmointiin. Luvussa kuusi esitellään, minkälaisia ohjelma robotille on tehty. Luvussa seitsemän selostetaan, miten toteutettujen ohjelmistojen testaus suoritettiin.. Luvussa kahdeksan kerrotaan, minkälaisia tuloksia opinnäytetyötä tehdessä on saatu aikaan. Luvussa yhdeksän kerrotaan, minkälaisia tulevaisuuden kehitys ideoita voisi olla. Luvussa kymmenen tehdään yhteenveto opinnäytetyöstä.

2 VANHUSTENHUOLTO

Tässä luvussa on katsaus siihen, miten vanhustenhuolto toimii Suomessa ja mitkä ovat sen kustannukset. Luvussa kerrotaan myös väestön ikääntymisestä ja sen vaikutuksista. Suomessa vanhustenhuolto jakaantuu kahteen päätoimintamalliin riippuen vanhusten kunnosta, jotka ovat avo- tai laitoshoito.

2.1 Väestön ikääntyminen

Ikääntymistä voidaan tarkastella joko biologisesta, psyykkisestä tai sosiaalisesta näkökulmasta. Biologinen ikääntyminen ilmenee asteittaisena kehon fyysisenä rappeutumisena iän myötä. Psyykkinen ikääntyminen ilmenee muistissa ja oppimiskyvyssä tapahtuvissa muutoksissa sekä persoonallisuuden kehittymisessä ikääntymisen seurauksena. Sosiaalisena ikääntymisenä ymmärretään kaikkia niitä muutoksia, joita ilmaantuu ikääntymisen myötä yksilön ja yhteiskunnan välisessä vuorovaikutuksessa. Pohjimmiltaan ikääntyminen on biologinen prosessi, mutta ihmisten arkielämässä se tulee näkyviin psyykkisen ja sosiaalisen vanhenemisprosessin kautta. Vanhenemisprosessi ja varsinkin sen kokeminen ovat yksilöllisiä. Ikääntymiseen liittyvät toiminnalliset vajavaisuudet alkavat ilmaantua teollisuusmaissa noin 75-80 vuoden iässä, mutta yksilöiden välillä on suuria eroa.[1]

Väestörakenne on muuttumassa niin, että ikääntyneiden osuus väestöstä kasvaa voimakkaasti. Tämä on seurausta sekä ihmisten eliniän pidentymisestä että syntyvyyden pienentymisestä. Taulukosta 2.1 nähdään miten Suomen väestön ikärakenne on muuttumassa vuosien 1900 ja 2030 välisenä ajanjaksona väestöennusteiden mukaan.[2]

Taulukko 2.1 Väestön ikärakenne Suomessa 1900 – 2030 [2]

Vuosi	0-14 osuus %	15-64 osuus %	64-74 osuus %	75- osuus %	Väkiluku
1900	35,0	59,6	3,9	1,5	2656
1950	30,0	63,3	4,7	2,0	4030
1980	20,2	67,8	7,9	4,1	4788
2000	18,1	67,0	8,4	6,5	5181
2030	15,5	58,8	12,6	13,1	5250

2.2 Laitoshoito

Laitoshoito pitää sisällään ihmisen iän ja kunnon vaativan hoidon, huolenpidon ja asianmukaisen kuntoutuksen. Siihen kuuluvat myös täydellinen ylläpito, mikä sisältää ruokahuollon, puhtauden, vaatetuksen, terveyden- ja sairaanhoidon sekä ihmisen henkisen, hengellisen ja sosiaalisen hyvinvoinnin kannalta tarpeelliset muut palvelut.[3]

Henkilö voi olla laitoshoidossa osavuorokautisesti, lyhytaikaisesti tai pitkäaikaisesti. Lyhytaikaisessa laitoshoidossa tuetaan vanhuksen selviytymistä kotona tarjoamalla mahdollisuus ajoittaisiin hoitojaksoihin ja kuntoisuuden arviointiin. Sillä on tarkoitus siirtää ja ehkäistä pysyvän laitoshoidon tarvetta. Jaksottaisessa hoidossa vanhus tulee laitokseen hoitoon 1-2 viikon ajaksi ja tätä voidaan toistaa tarpeen mukaan 1-2 kuukauden välein. Laitoshoitojakson aikana vanhuksen hoidon asianmukaisuus tarkistetaan ja vanhusta kuntoutetaan selviytymään avohoidossa. Viikkohoidossa vanhus on arkipäivät laitoshoidossa ja viettää viikonloput avohoidossa. Pitkäaikaista laitoshoitoa annetaan vanhuksille, joille ei enää voida järjestää heidän tarvitsemaansa ympärivuorokautista hoitoa avohoidon palvelujen avulla, mutta jotka eivät kuitenkaan ole sairaalamaisen hoidon tarpeessa. Kunnissa, missä ei ole riittävästi laitospaikkoja vanhuksia varten, vanhuksia hoidetaan terveyskeskuksien vuodeosastoilla, vaikka heidän terveydentilansa ei tätä vaatisi.[3]

2.3 Avohoito

Vanhustenhoidossa avohoidolla tarkoitetaan niitä tehostetun avohoidon muotoja, joilla pyritään edistämään vanhusten kotona selviytymistä. Niitä ovat esimerkiksi kotisairaanhoito, kotipalvelut ja palveluasuminen.[4]

Vanhustenhoidossa kotipalvelulla tarkoitetaan vanhuksille kotiin tarjottavia palveluita. Niiden tarkoitus on tukea ja auttaa vanhuksia, joiden toimintakyky on alentunut tai jotka sairaudesta johtuen tarvitsevat apua ja tukea selviytyäkseen kotona arkipäivän askareista, kuten ruoanlaitosta ja henkilökohtaisen hygienian hoitamisesta**.** Kotipalvelun työntekijät ovat pääasiassa kodinhoitajia, kotiavustajia ja lähihoitajia.[5]

Vanhustenhoidossa kotisairaanhoidolla tarkoitetaan asiakkaan kotona lääkärin määräämiä sairaanhoidollisia toimia, näytteen ottoa, lääkityksen valvontaa ja vanhuksen terveydentilan tarkkailemista. Omaisten tukeminen on osa kotisairaanhoitoa. Kotisairaanhoidon henkilöstö on koulutukseltaan pääasiassa sairaanhoitajia.[5]

Vanhustenhoidossa palveluasuminen on tarkoitettu vanhuksille, jotka tarvitsevat runsaasti ympärivuorokautista hoivaa ja apua. Palveluasumisen tarkoituksena on saada vanhukset asumaan mahdollisimman pitkään itsenäisesti oman elämäntyylinsä ja itsemääräämisoikeutensa säilyttäen. Vanhusten palveluasumiseen sisältyy muun muassa ruokahuoltoon, terveyden ylläpitämiseen, sairauksien hoitoon ja virkistystoimintaan liittyvät palvelut. Palveluasumisen maksu määräytyy asiakkaan maksukyvyn mukaan. [5]

2.4 Vanhustenhuollon kustannukset

Kunnat vastaavat Suomessa vanhustenhuollon järjestämisestä. Ne voivat tuottaa vanhustenhuollon palvelut yksin tai muiden kuntien kanssa yhteistyössä. Kunnat voivat myös ostaa palveluita muilta kunnilta, järjestöiltä tai yksityisiltä palveluntuottajilta.[6]

Taulukosta 2.4 nähdään, että vanhustenhuollon kustannukset olivat vuonna 2005 yhteensä 2436,1 miljoonaa euroa vuodessa ja kustannukset olivat nousseet vuoden 2010 loppuun mennessä 3315,6 miljoonaan euroa. Vuosien 2005–2010 välillä vanhustenhuollon kustannukset ovat nousseet 36,1 prosenttia. Vanhustenhuollon käyttömenoihin käytettiin vuonna 2005 2381,6 miljoonaa euroa ja vuoteen 2010 mennessä käyttömenot olivat nousseet 3242,5 miljoonaan euroon. Vanhustenhuollon investointeihin käytettiin

Taulukko 2.4 Vanhustenhuollon kustannukset Suomessa vuodesta 2005 vuoteen 2010 [7]

Toiminto	2005	2006	2007	2008	2009	2010
1. Vanhustenhuollon käyttömenot	**2381,6**	**2528,6**	**2719,8**	**2958,0**	**3177,8**	**3242,5**
1.1 Vanhusten laitoshoitopalvelut	699,4	731,1	767,6	763,6	820,5	794,7
1.1.1 Julkisten palveluntuottajien tuottamat palvelut	531,7	574,6	606,3	595,6	658,5	640,8
1.1.2 Yksityisten palveluntuottajien tuottamat palvelut**	167,6	156,5	161,3	168,0	162,1	153,9
1.2 Julkiset perusterveydenhuollon pitkäaikaishoidon palvelut (vähintään 90 vrk) (väh. 65-vuotiaat)	445,5	429,4	454,5	497,5	475,2	429,8
1.3 Kotipalvelut*	476,9	509,8	520,9	563,6	605,6	627,7
1.3.1 Julkisten palveluntuottajien tuottamat palvelut*	439,7	472,1	484,9	521,9	565,7	583,6
1.3.2 Yksityisiltä palveluntuottajilta ostetut palvelut*	37,3	37,8	36,0	41,7	39,9	44,1
1.4 Muut vanhusten palvelut*	759,8	858,2	976,9	1133,4	1276,5	1390,2
1.4.1 Julkisten palveluntuottajien tuottamat palvelut*	323,3	330,2	386,2	449,9	499,6	556,1
1.4.2 Yksityisiltä palveluntuottajilta ostetut palvelut*	436,4	528,0	590,7	683,5	776,8	834,1

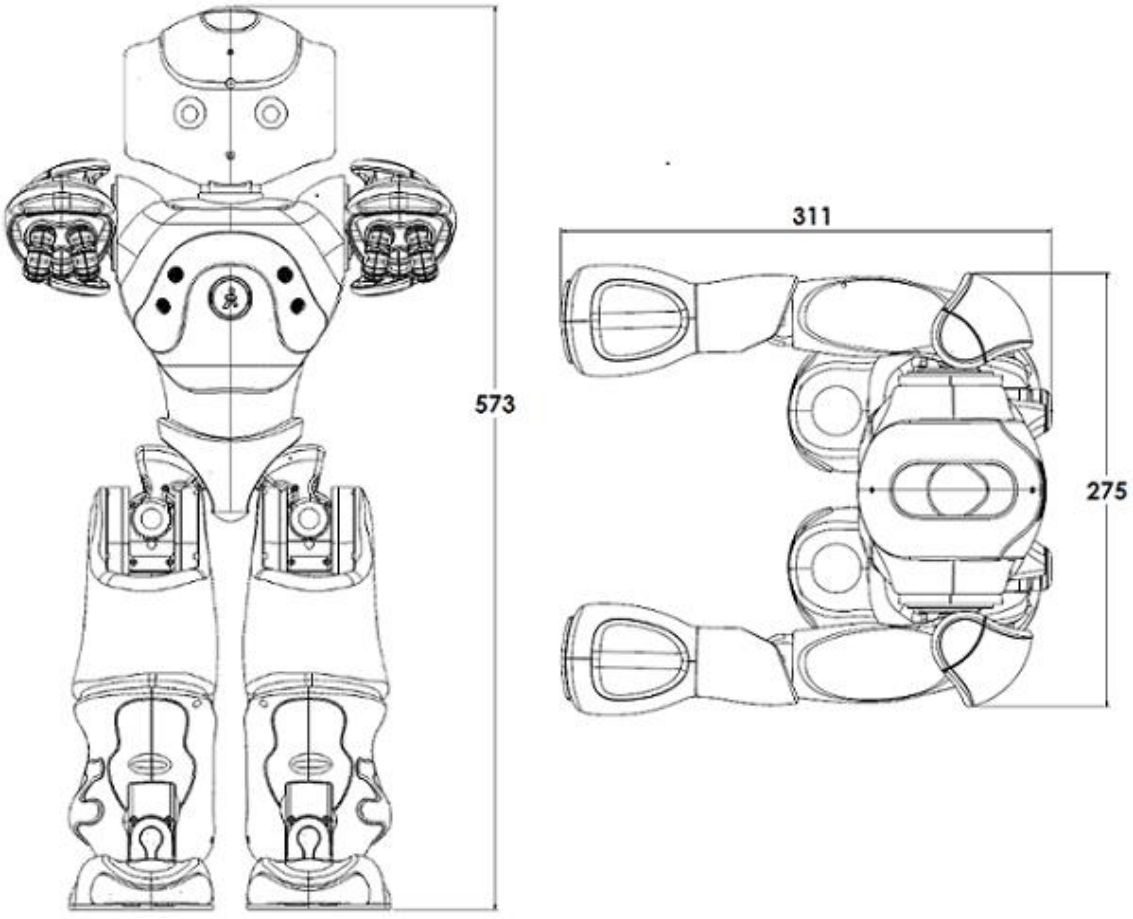
2. Vanhustenhuollon investoinnit	**54,5**	**52,8**	**63,3**	**72,5**	**63,8**	**73,4**
2.1 Kuntien vanhusten laitoshoitoinvestoinnit	53,7	52,6	63,2	72,4	63,5	72,0
2.2 Vanhainkotikuntayhtymien investoinnit	0,8	0,2	0,2	0,1	0,3	1,3
Vanhustenhuollon menot yhteensä	**2436,1**	**2581,3**	**2783,2**	**3030,5**	**3241,6**	**3315,8**

* Luokka ei sisälly SHA-tilastoinnin mukaisiin terveydenhuoltomenoihin, mutta sisältyy terveydenhuoltoon liittyvien toimintojen menoihin luokkaan HC.R.6.1
** Luokka sisältää kuntien ja kuntayhtymien ostopalvelut yksityisiltä, Valtiokonttorin pitkäaikaishoidon ostot sekä Raha-automaattiyhdistyksen vanhustenhuollon toiminta-avustukset
*** Kotitalouksien itsenäisesti hankkimat palvelut ovat tarkastelun ulkopuolella

vuonna 2005 54,5 miljoonaa euroa ja vuonna 2010 investointeihin käytettiin 73,4 miljoonaa euroa. Vuosien 2005–2010 välillä vanhustenhuollon investoinnit ovat nousseet 34,7 prosenttia.[7]

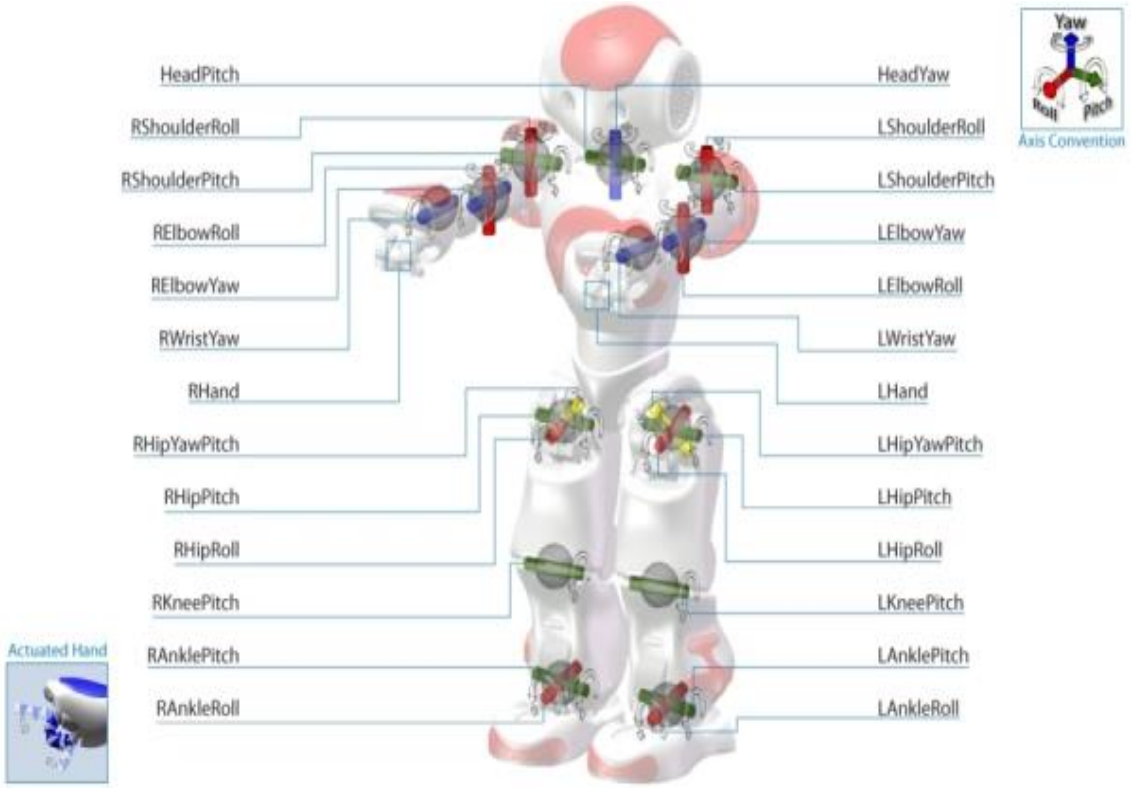
3 NAO

Nao on keskikokoinen ihmisen kaltainen robotti, joka on noin 58 senttimetriä pitkä ja painaa noin 5,2 kilogrammaa (kuva 3.1). Naon ulkokuori on valmistettu teknisestä muovista. Valmistajan mukaan akku kestää noin 90 minuuttia jatkuvaa liikkumista.[8]



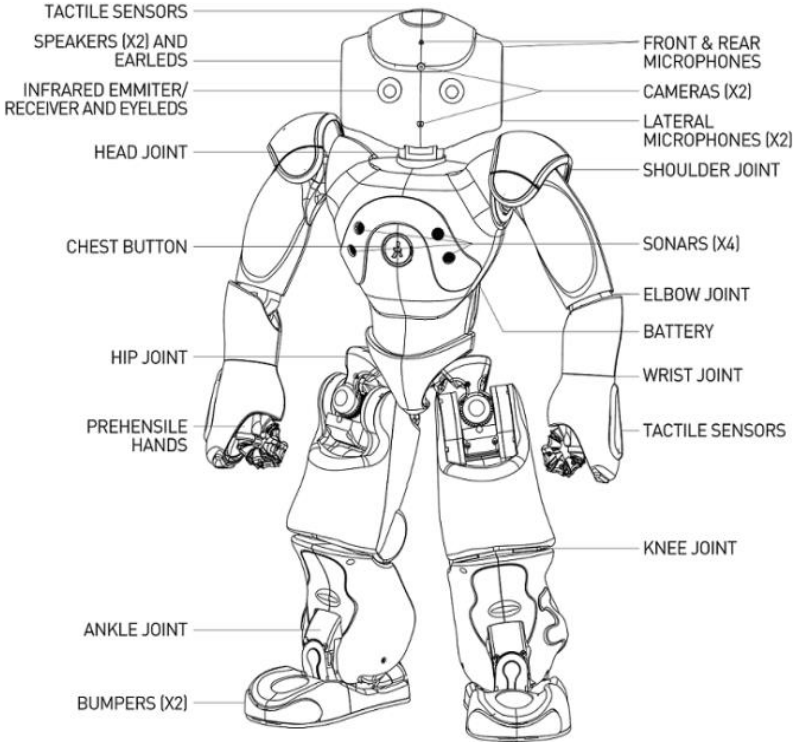
Kuva 3.1

Naolla on 25 sähköisesti ohjattavaa niveltä ja jokaisessa nivelessä on Hall-anturi, jonka tehtävänä on mitata, missä kulmassa nivel kulloinkin on. Nivelten sijainnit näkyvät kuvassa 3.2.[8]



Kuva 3.2

Robotin silminä toimii kaksi 640×480 pikselin CMOS-kennolla varustettua digitaalikameraa. Korvina toimii neljä mikrofonia ja äänilähteenä kaksi kaiutinta. Niiden sijainti on esitetty kuvassa 3.3. [8]



Kuva 3.3

Robottiin voi muodostaa verkkoyhteyden joko langattoman lähiverkkotekniikan tai ethernet-tekniikan avulla. [8]

Robotin aisteina toimii monenlaisia antureita, joilla voidaan aistia robotin ympäristöä ja havaita siinä tapahtuvia muutoksia. Siinä on yksi kiihtyvyysanturi, joka mittaa kiihtyvyyksiä kolmen akselin suhteen ja lisäksi siinä on kaksi kappaletta gyroskooppeja, joka mittaavat robotin asentoa yhden akselin suhteen. Robotissa on sisäänrakennettu 2-kanavainen kaikuluotain esteiden havaitsemista varten. Kaikuluotaimella voi havaita esteen luotettavasti puolentoista metrin päästä. Robotin molemmissa jaloissa on neljä paineanturia, jotka tunnistavat esimerkiksi sen, koska kyseinen jalka on maassa. Robotissa on kahdeksan kosketuskytkintä. Päässä on kolme, rinnassa yksi, molemmissa käsissä yksi ja molemmissa jaloissa yksi anturi. Robotin molemmissa silmissä on infrapunalähetin ja vastaanottimet. [8]

Robotissa on 19 RGB (Red,Green and Blue) lediä ja 20 sinistä lediä, jotka sijaitsevat niin, että molemmissa silmissä on 8 RGB lediä, rinnassa yksi RGB led, molemmissa jaloissa yksi RGB ledi ja molemmissa silmissä 10 sinistä lediä. [8]

Robotin aivoina toimii Intel Atom Z530 1600 Mhz prosessori, jossa on 512 kilotavun välimuisti. Atom on intelin valmistama kannattavia, älypuhelimia ja sulautettuja järjestelmiä suunniteltu x86-arkkitehtuuriin perustuva 32-bittinen prosessori. Naossa olevan Atom Z530 kellotaajuus on 1600 Mhz ja tavallisesti se kuluttaa sähköä valmistajan antamien tietojen perusteella 0,96 wattia. Prosessori perustuu 45 nanometrin valmistustekniikkaan. [8,9]

Robotin muistina toimii 2 GB flash-muisti ja 256 MB SDRAM-muisti. Flash-muisti toimii robotin pysyväismuistina, eli muistina, johon robotissa suoritettavat ohjelmat ja käyttöjärjestelmä ovat tallennettuna. SDRAM-muisti toimii keskusmuistina, joka toimii suorituksessa olevien ohjelmien ja käyttöjärjestelmän työmuistina. Pysyväismuistia voi laajentaa maksimissaan 8 GB muistikortilla.[8]

Robotissa on NAOqi OS-käyttöjärjestelmä, joka perustuu Gentoo Linux -jakeluun. NAOqi OS on robottia varten suunniteltu Linux-pohjainen 32-bittinen käyttöjärjestelmä, jonka päällä käyttäjien tekemät ohjelmat pyörivät. Käyttöjärjestelmä on avoimen lähdekoodin ohjelma. NAOqi OS on käytössä kaikissa Aldebaran Roboticsin roboteissa. NAOqi OS on avoimen lähdekoodin ohjelmisto.[8]

Robotin voimanlähteenä toimii kuusikennoinen litiumioniakku, jonka kapasiteetti on 48,6 wattituntia, maksimi ulostulovirta laturissa on 2,3 ampeeri ja muuten 2 ampeeria ja nimellisjännite 21,6 volttia. Akulla saavutetaan valmistajan antamien tietojen mukaan 60 minuutin käyttöaika aktiivisessa käytössä ja 90 minuutin käyttöaika normaalikäytössä.

Akun latausaika on 5 tuntia, suositus latausvirta on 2,3 ampeeria ja maksimi latausjännite on 25,5 volttia.[8]

4 NAO OHJELMOINTIALUSTANA

Tämä luvussa luodaan katsaus siihen, minkälaisilla ohjelmointikieliellä Naoa voidaan ohjelmoida. Luvussa tutustutaan tarkemmin Pytho-ja C++-ohjelmointikieliin, joita käytettiin Naon ohjelmointiin. Luvussa kerrotaan myös NAOqi ohjelmistokehyksestä, joka helpottaa Naon ohjelmointia tarjoamalla paljon erilaisia rajapintoja ohjelmoijan käyttöön.

4.1 Nao-ohjelmointiin käytettävät ohjelmointikielet

Nao voidaan ohjelmoida käyttäen ainakin kuutta eri ohjelmointikieltä: C++, Python, Java, .Net, Matlab ja Urbi. Robotin valmistajan dokumentaatiosta löytyvät koodi esimerkit ja rajapintadokumentaatio vain C++:aa ja Pythonia varten. Tässä työssä on käytetty ohjelmointikielinä Pythonia ja C++:aa. [10]

4.1.1 Python-ohjelmointikieli

Python on monipuolinen, oliopohjainen, dynaamisesti tyypitetty ja tulkattava ohjelmointikieli. Kielen tärkeimpiä suunnittelun periaatteita on ollut yhdistää helposti luettava lähdekoodi tehokkaaseen ilmaisuun. Kieli on suunniteltu yleiskäyttöiseksi ja sen vakiokirjastot kattavat monien eri sovelluskohteitten tarpeet. Vakiokirjastojen lisäksi Pythoniin on saatavilla paljon lisäkirjastoja Internetistä erilaisten sovelluskohteiden tarpeisiin. Ohjelmamoduuleja ja kirjastoja voidaan toteuttaa myös muilla ohjelmointikielillä kuten esimerkiksi C:llä tai C++:lla. Pythonilla kirjoitetut ohjelmat toimivat Windows-, Linux- ja OSX-ympäristöissä ja lisäksi monissa muissa käyttöjärjestelmäympäristöissä, esimerkiksi Symbian- ja Android-ympäristössä. Pythonia jaetaan avoimella, Open Source Iniativen hyväksymällä lisenssillä ja se sallii kielen ja sen kirjastojen käyttämisen ilmaiseksi myös kaupallisissa sovelluksissa ilman, että lähdekoodia pitää julkaista.[11,12,13,14]

Pythonin ensimmäinen versio on syntynyt 1980-luvun loppupuolella ja Naon ohjelmoinnissa käytettävä Python 2.6 on julkaistu vuonna 2008. Kielen alkuperäisenä kehittäjänä ja isänä on ollut Guido van Rossum, joilla on ollut vahva rooli kielen kehityksessä koko sen historian ajan. [15,16]

Python on saanut nimensä brittiläisen komediaryhmän Monty Pythonin tunnetun Monty Python lentäväsirkus -televisiosarjan mukaan. Nimi oli kehittäjiensä mukaan tarpeeksi

lyhyt, mutta samalla riittävän yksilöivä ja tarpeeksi mysteerinen. Lisäksi kielen dokumentaatiossa olevissa esimerkkikoodeissa on yritetty välttää liiallista vakavuutta satunnaisilla viittauksilla ryhmän tuotantoon. [12,17]

4.1.2 C++-ohjelmointikieli

C++-ohjelmointikieli on Bjarne Stroustrupin C-kielen pohjalta kehittämä ohjelmointikieli. C++-kieli on kehitetty C-kielestä lisäämällä siihen esimerkiksi olio-ohjelmointiin ja geneerisyyteen liittyviä ominaisuuksia. Kielen ensimmäinen versio on julkaistu vuonna 1983. Kielen alkuperäinen nimi on C with Classes, joka jälkikäteen muutettiin C++:ksi. [18]

C++ on yksi maailman suosituimmista ohjelmointikielistä. Sitä käytetään niin sovellusohjelmoinnissa, järjestelmäohjelmoinnissa, sulautettujen järjestelmien ohjelmoinnissa, korkeaa tehokkuutta vaativien palvelin- ja asiakassovellusten ohjelmoinnissa sekä peliohjelmoinnissa. [19,20,21]

C++ on staattisesti tyypitetty, vapaamuotoinen, monimuotoinen ja yleiskäyttöinen ohjelmointikieli ja sitä pidetään keskiaktiivisena kielenä, koska siinä on korkean tason ja matalan tason kielen ominaisuuksia. Monimuotoisuus tarkoittaa ohjelmointikielistä puhuttaessa sitä, että kielessä voi käyttää erilaisia ohjelmointitapoja, esimerkiksi olio-ohjelmointia tai funktionaalista ohjelmointia. [22]

4.2 NAOqi

NAOqi on ohjelmistokehys, jota käytetään ohjelmien tekemiseen Nao-robotille. Ohjelmistokehys on ohjelmistorunko, jota voidaan täydentää eri tavoin erilaisia tarkoituksia varten. Ohjelmistokehys on ohjelmistotuote, jossa on aukkoja ennalta odotettuja täydennyksiä varten. Ne ovat hyvin yleinen tekniikka tuoterunkoarkkitehtuurien toteuttamiseen oliomaailmassa. Ne ovat olioperustainen tapa toteuttaa tuoterunko. Niillä on sama perustavoite kuin tuoterungolla: laajamittainen ja systemaattinen ohjelmistojen uudelleenkäyttö. Ohjelmistokehys ei itsessään yleensä ole suorituskelpoinen ohjelma. Haluttu ohjelmisto saadaan täyttämällä kehyksen aukot uudella koodilla, joka toteuttaa ohjelman toiminnallisuuden muuttamatta kehyksen tarjoamaa arkkitehtuuria. Se on oikeastaan vain luokka-, komponentti- tai rajapintakokoelma, joka toteuttaa jonkin ohjelmistojoukon yhteisen arkkitehtuurin ja perustoiminnallisuuden.[23,24]

4.2.1 NAOqi moduulit

NAOqi sovelluskehys voidaan jakaa seitsemään moduuliin, jotka jakaantuvat alimoduuleihin. Ne ovat NAOqi Core, NAOqi Motion, NAOqi Vision, NAOqi Audio, NAOqi Sensors, NAOqi Trackers ja DCM. Taulukossa 4.2.1.1 on kuvattu moduulien tehtävät [25]

Taulukko 4.2.1.1

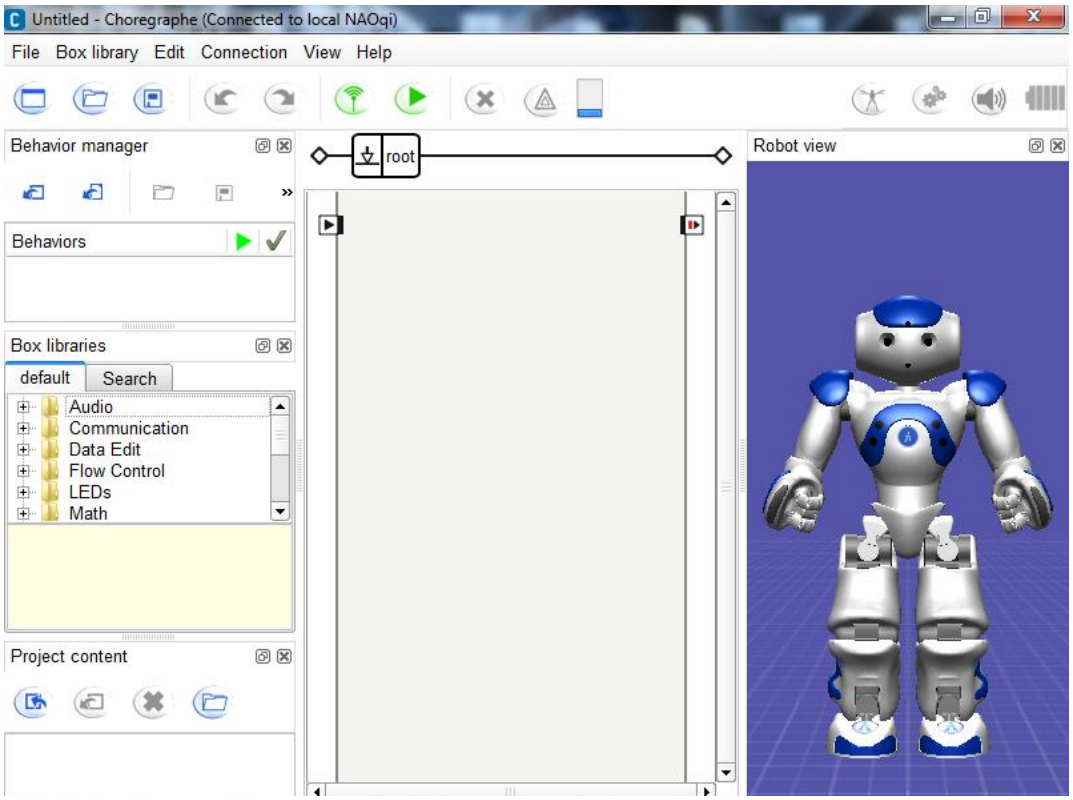
Moduuli nimi	moduulin tehtävät
NAOqi Core	Muistinhallinta ja robotin muistiin tallennettujen ohjelmien käsittely.
NAOqi Motion	Liikkuminen ja liikkeiden nauhoittaminen.
NAOqi Vision	Hahmontunnistus ja kameroiden ohjaus.
NAOqi Audio	Mikrofonien ja kaiuttimien ohjaamine.n
NAOqi Sensors	Anturien ja ledien ohjaaminen.
NAOqi Trackers	Punaisen pallon ja ihmisen kasvojen seuraaminen.
DMC	Toimilaitteiden arvojen lukeminen esimerkiksi robotin kiihtyvyysanturien ja gyroskooppien arvojen lukemista varten.

5 NAO-OHJELMOINTIIN KÄYTETTÄVÄT TYÖKALUT

Tässä luvussa luodaan katsaus siihen, minkälaisia työkaluja on helpottamassa ohjelmistojen tekoa Naolle ja helpottamaan ohjelmistojen testausta. Luvussa tutustutaan Choregraphe-ohjelmistokehityspakettiin, Notepad++-tekstieditoriin, Monitor-apuohjelmaan ja NAOsim-simulaattoriin. Luvussa kerrotaan, miten edellä mainitut ohjelmistot auttavat ohjelmien kehittämistä Naolle ja mitä kyseisillä ohjelmistolla voi yli päätään tehdä.

5.1 Choregraphe

Choregraphe on graafisella käyttöliittymällä varaustettu ohjelmistonkehityspaketti, joka on osa Aldebaran Robotics ohjelmistoa. Sitä käytetään Naon ohjelmoinnissa, ohjelmien siirtämisessä Naon muistiin ja ohjelmien debuggaamisessa. Kuvassa 5.1.1 näkyy Choregraphe-ohjelman käyttöliittymän näkymä. [26]



Kuva 5.1.1

Choregraphella voidaan tehdään Naolle ohjelmia joko yhdistelemällä ohjelman mukana tulevia valmiilta ohjelmakirjastoja graafisessa käyttöliittymässä toisiinsa, yhdistelmällä valmiita kirjastoja itse tehtyjen kirjastojen kanssa graafisessa käyttöliittymässä tai yhdistelemällä vain itse tehtyjä kirjastoja graafisessa käyttöliittymässä toisiinsa. Omia kirjastoja voi tehdä Choregraphessa Python- tai Urbi-ohjelmointikielillä. Kirjastoja tehdessä on käytössä NAOqi-ohjelmointikehyksen tarjoamat rajapintafunktiot. Choregraphessa tehtyjä ohjelmia voidaan ajaa neljällä eri tavalla: siirtämällä tehty ohjelma robotin muistiin ja suorittaa ohjelma robotissa, ajamalla ohjelma etäyhteyden yli robotissa, ohjelmia voidaan myös ajaa Choregraphissa paikallisesti niin sanotussa debuggaus-tilassa tai ohjelmia voidaan ajaa myös NAOsim-nimisessä simulaattorissa. Paikallisesti ohjelmaa ajettaessa robotin liikkeet näkyvät Choregraphessa olevassa Robot view -näkymässä. Paikallisesti ajettavissa ohjelmissa ei voida suorittaa kuin NAOqi Motion - ja NAOqi Core -moduulien rajapintafunktioiden sisältämiä toiminnallisuuksia. [26]

Choregraphella tehtyjen ohjelmien siirto robotin muistiin tapahtuu Choregraphissa olevalla Behavior manager -työkalulla. Robotin muistiin tallennuttuja ohjelmia voidaan käynnistää ohjelmallisesti toisella ohjelmalla tai painamalla jotain robotin kahdeksasta kosketuskytkimestä.[26]

5.2 Notepad++

Notepad++ on avoimeen lähdekoodin pohjautuva tekstieditori ja koodieditori. Notepad++ on saatavilla vain Windows-käyttöjärjestelmillä varustettuihin koneisiin. Se perustuu Scintilla-komponenttiin, joka on vapaan lähdekoodin tekstitieditorikompomentti. Editori tukee useiden eri ohjelmointikielien syntaksi korostusta, esimerkiksi C++, C ja Python.

Tässä työssä käytettiin Notepad++ editoria Python ja C++ ohjelmointikielisten ohjelmien toteuttamiseen. Hyvän työkalun Pyhtonin koodaamiseen Notepad++:sta tekee se, että siinä on hyvä Python-ohjelmointikielen syntaksikorostus ja hyvä tuki Python sisennyksiin perustuvaan koodilohkojakoon. Ainoana puutteena on se, ettei siihen saa integroitua Python-tulkkia, joka tarvitaan Python-koodin kääntämiseen tavukoodiksi ja tavukoodin suorittamiseen.

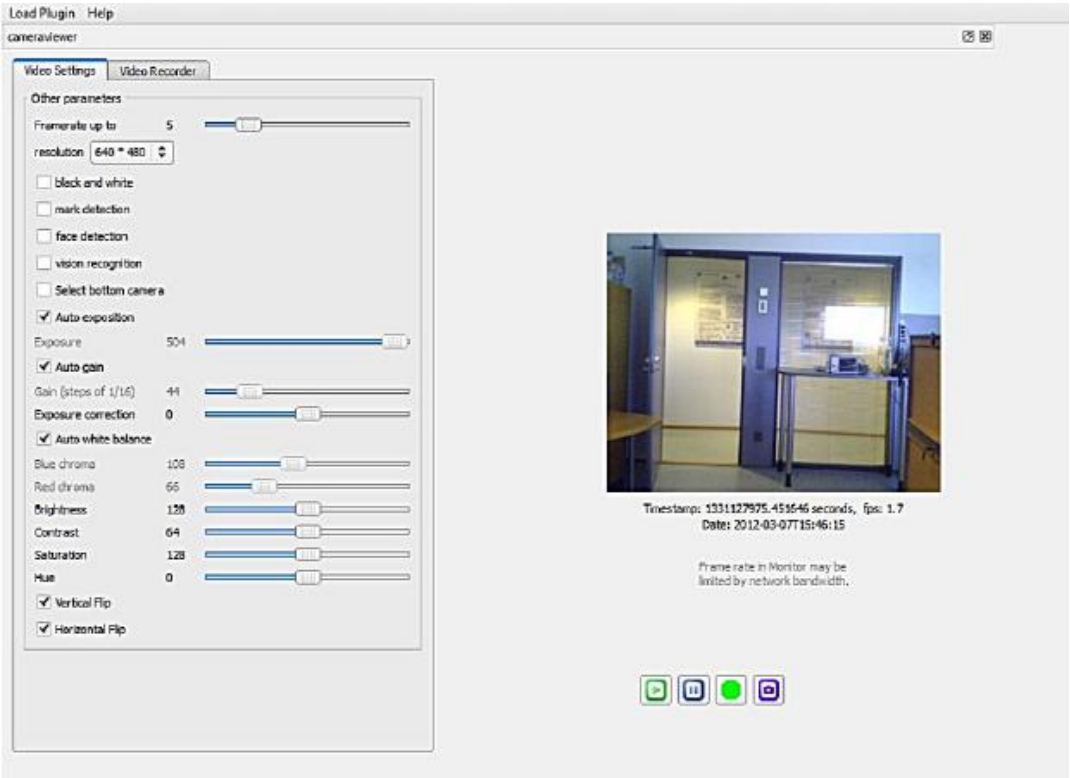
5.3 Monitor

Monitor on työpöytäsovellus, joka asennetaan Choregraphe-ohjelmistokehityspaketin asennuksen yhteydessä osana Aldebaran Robotics -ohjelmistoa. Monitor antaa käyttäjälle

tietoa siitä, mitä robotti näkee ja aistii. Monitori sisältää kolme erilaista liitännäistä, jotka ovat Camera Viewer, Memory Viewer ja Laser Viewer, joista Laser Viewer on käytössä vain Laser Head NAO:n omistajille. Kuvassa 5.2.1 on Monitor ohjelman aloitusnäkymä. [27]

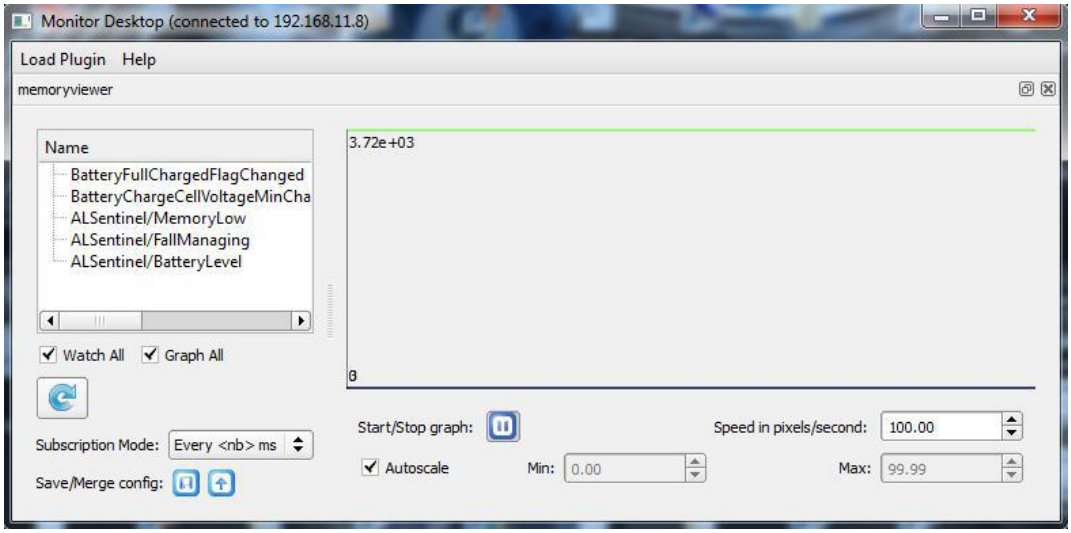
Kuva 5.2.1

Camera viewer -liitännäisellä voi katsella robotin kameroiden lähettämää kuvaa ja tallentaa videokuvaa tietokoneen muistiin. Liitännäisestä voi muokata kuvan resoluutiota ja videon kuvataajuutta. Videossa kuvataajuudella tarkoitetaan sitä, kuinka monta kuvaa videossa on sekuntia kohti. Liitännäisessä voidaan myös säätää kuvan kirkkautta, värikylläisyyttä ja terävyyttä. Liitännäisen avulla voi myös tutkia, miten robotin kasvojen tunnistusalgoritmi toimii ja samalla voi myös testata, miten maanmerkkien tunnistusalgoritmi toimii. Kuvassa 5.2.2 on Monitor-ohjelman Camera viewer -liitännäisen käyttöliittymänäkymä. [27]

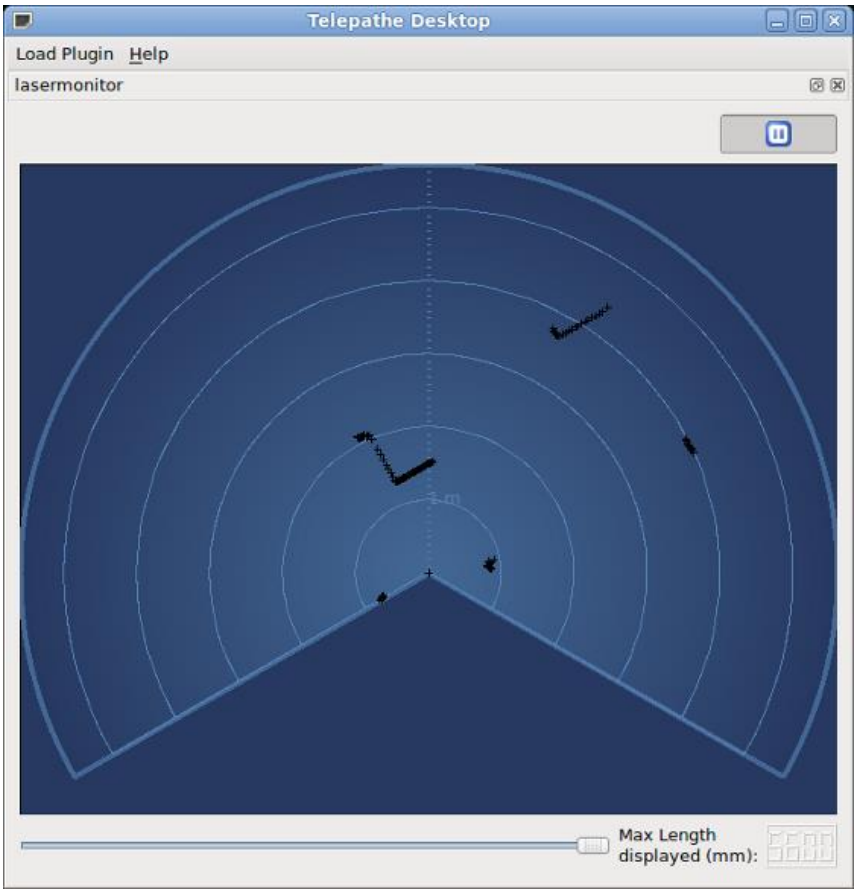


Kuva 5.2.2

Memory Viewerin avulla voidaan tarkastella kuvaajien muodossa, minkälaista dataa robotin eri antureista saadaan ulos. Sen avulla voidaan saada kaikkien robotin anturien data yhtä aikaa näkyville samaan kuvaajaan tai käyttäjien valitsemien anturien välittämä data. Kuvassa 5.2.3 on Monitor-ohjelman Memory Viewer -liitännäisen käyttöliittymänäkymä. [27]



Kuva 5.2.3

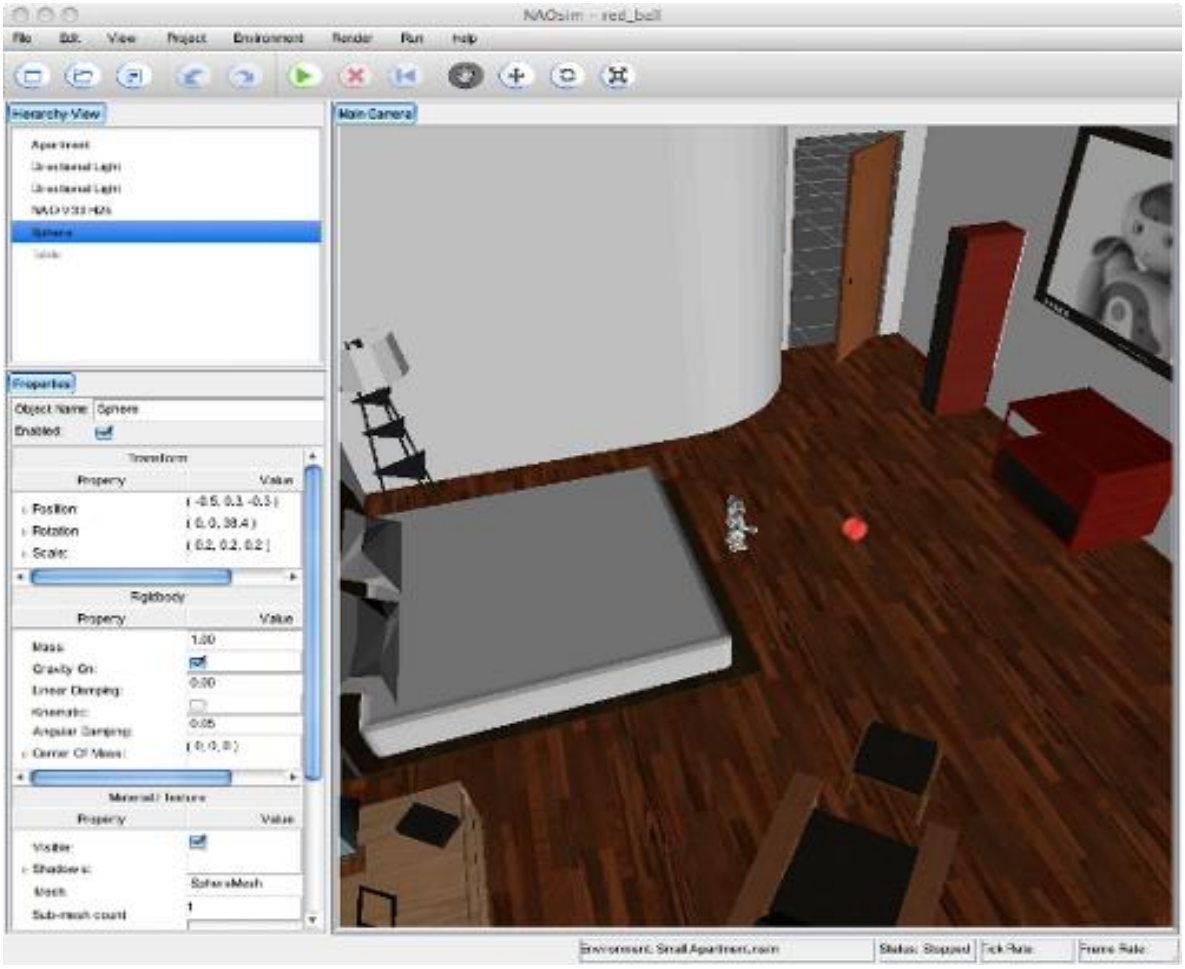


Kuva 5.2.4

Laser Viewer -liitännäinen tekee tutkanäkymää Laser Head NAO:n mittaamasta tiedosta. Laser Viewer -näkymästä voidaan nähdä, kuinka kaukana esteet ovat robotista ja ovatko ne robotin kulkureitillä. Kuvassa 5.2.4 on Monitor-ohjelman Laser Viewer -liitännäisen käyttöliittymänäkymä. [27]

5.4 NAOsim

NAOsim on simulaattori, jossa voidaan testata turvallisesti Nao-robotille tehtyjä ohjelmia virtuaalimaailmassa. NAOsimissä voidaan rakentaa erilaisia virtuaaliympäristöjä, joissa voidaan testata robotille tehtyjä ohjelmia ja voidaan turvallisesti katsoa, miten robotti reagoi erilaisiin virtuaaliympäristön ärsykkeisiin. Kuva 5.3.1 on NAOsimin käyttöliittymästä. NAOsim tukee suurinta osa NAOssa olevista antureista ja toimilaitteista. Tuetut anturit ja toimilaitteet näkyvät taulukossa 5.3.1[28]



Kuva 5.3.1

Taulukko 5.3.1

Raajojen asentoa mittaavat anturit
Jalkapohjan paineanturit
Jalkojen kosketuskytkimet
Molemmat kamerat
Raajoja liikuttavat moottorit
Inertia-yksikkö

6 TOTEUTETUT OHJELMAT

Tässä luvussa luodaan katsaus siihen, minkälaisia ohjelmia diplomityötä tehtäessä on toteutettu ja millä perusteella toteutettavat ohjelmat ovat valittu. Diplomityötä tehdessä Naolle tehtiin seuraavat ohjelmat: tuolijumppa ohjelma, sisätilanavigointi ohjelma ja valvontasovellus. Sisätilanavigointiohjelma on osa valvontasovellusta. Valvontasovellukseen kuului myös muutama pienempi ohjelma, jotka ovat hälytyksen hakuohjelma, kuvan välitysohjelma ja puheen välitysohjelma. Tuolijumppaohjelma rakentuu useista pienistä jumppaliikeohjelmista, jotka ovat jalan nosto ja nilkan ojennusliike, käden nosto ja sivulleojennusliike, käsien heilutusliike, ranteen pyöritysliike, taputusliike, uintiliike ja aaltoliike. Luvussa kerrotaan myös, miten ohjelmat ovat toteutettu Naolle.

6.1 Toteutettujen ohjelmien valinta perusteet

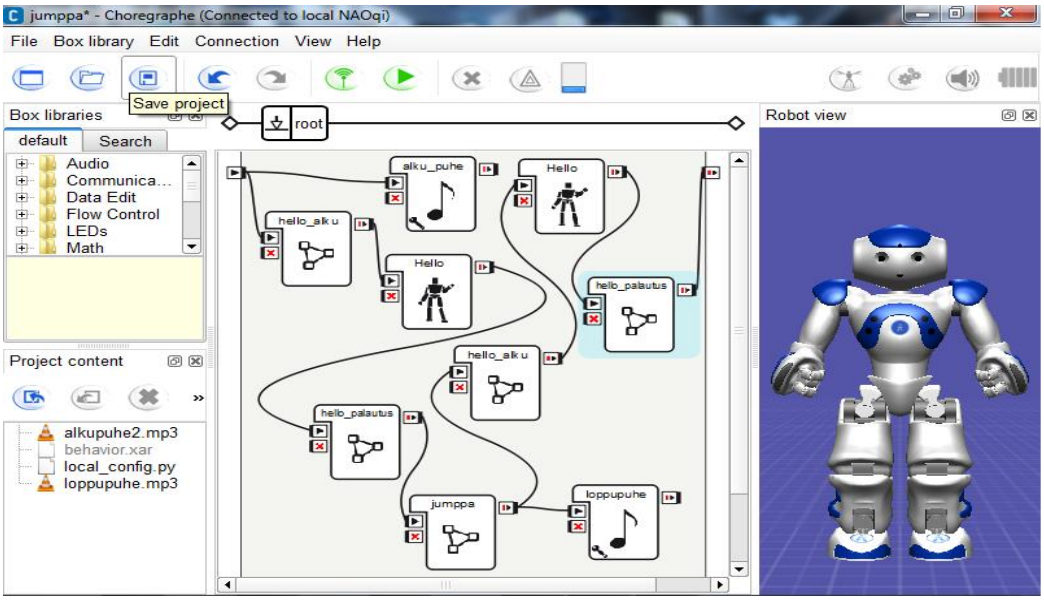
Tutkimushankkeessa mukana olleiden hoitokotien toiveilla oli suuri painoarvo siinä, minkälaisia ohjelmia Naolle tehtiin tutkimushankkeen aikana. Tutkimushankkeessa mukana olleiden hoitolaitoksien toiveiden perustella päätimme toteuttaa robotille tuolijumppaohjelman ja valvontasovellusohjelman. Valvontasovellusta varten piti robotille toteuttaa sisätilanavigointiohjelma.

6.2 Tuolijumppaohjelma

Naolle on opetettu useita jumppaliikkeitä, jotka tehdään tuolilla istuen. Robotti tekee liikkeet käsiensä ja jalkojensa avulla ja antaa äänitettyjä neuvoja kustakin liikkeestä. Jumppa ohjelma kestää noin 15 minuuttia ja se sisältää viisi toistoa jokaisesta liikkeestä. Jumppa sisältää seitsemän erilaista käsillä ja jaloilla suoritettavaa liikettä, jotka tehdään musiikin tahtiin. Jumpan kestoa ja liikkeiden toistomääriä voidaan helposti muokata jumppaajien kunnon ja mieltymysten mukaan. Jumppa ohjelmasta voidaan helposti muodostaa 7! erilaista versiota.[29]

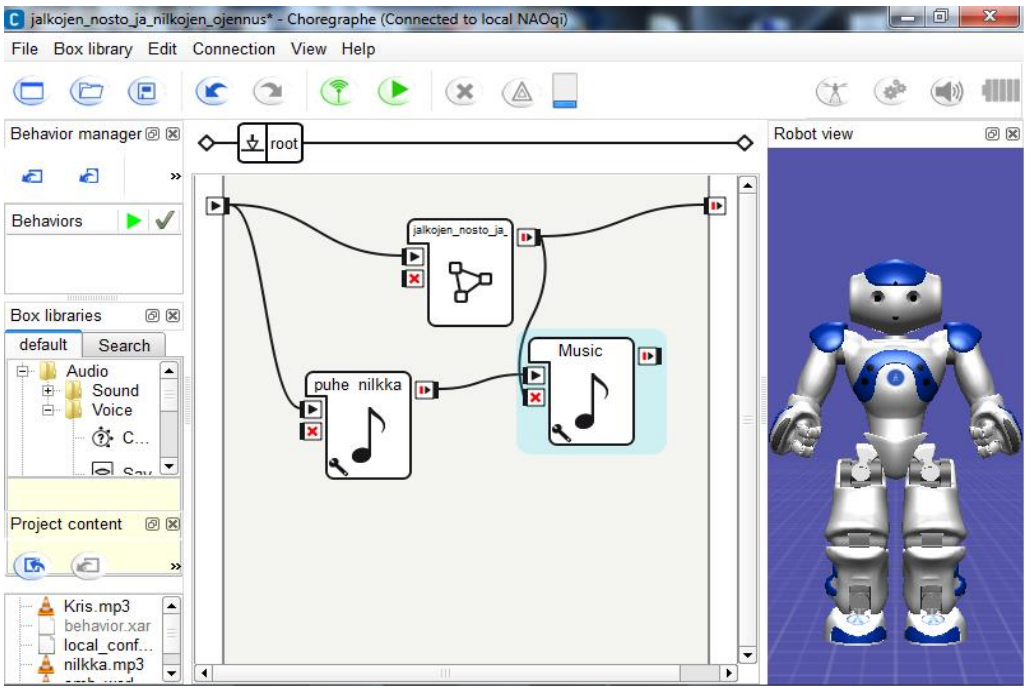
Ohjelma on toteutettu yhdistelemällä Choregraphen sisältämiä valmiita kirjastoja ja itsetehtyjä Pythonilla kirjoitettuja kirjastoja. Niiden yhdistäminen on tehty Choregraphia hyväksikäyttäen. Jokainen jumppaliike on tallennettu erillisenä Choregraphilla tehtynä ohjelmana robotin muistiin.

Kuvassa 6.2.1 nähdään, miten jumppaohjelma on toteutettu Choregraphi ohjelmisto työkalun avulla. Kuvassa 6.2.1 nähtävän jumppakirjaston tehtävä on käynnistellä jumppaliike ohjelmia, jotka on tallennettu robotin muistiin. Jumppaliikeohjelmia käynnistellään satunnaisessa järjestyksessä kuitenkin niin, että jokainen liike toistuu jumppaohjelman aikana vain kerran.



Kuva 6.2.1: Ghoregraphin avulla toteutettu jumppaohjelma

Kuvassa 6.2.2 nähdään, miten miten jalan nosto ja nilkan ojennus liike on toteutettu Choregraphe-ohjelman avulla yhdistelemällä ohjelman valmiita kirjastoja itse tehtyihin kirjastoihin. Kaikki muut jumppaohjelman liikkeet ovat toteutettu samalla tavalla.



Kuva 6.2.2 Ghoregraphin avulla toteutettu jalan nosto ja nilkan ojennusliike

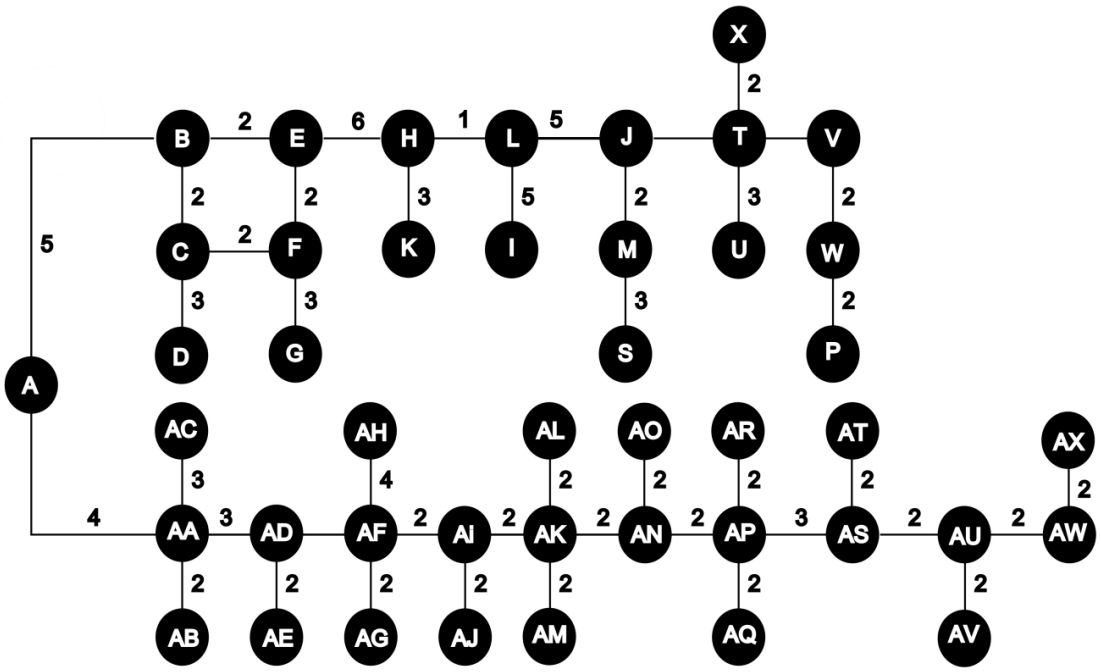
6.3 Sisätilanavigointi

Jotta robotti voisi liikkua itsenäisesti sisätiloissa, siihen on ohjelmoitu karttapohjainen navigointijärjestelmä. Kun robottiin on tallennuttu tarkka kartta alueesta, se voi navigoida paikasta toiseen kartan avulla esimerkiksi sairaaloissa ja hoitokodeissa. Navigointi edellyttää, että robotti tietää lähtöpaikkansa tarkasti. [30]

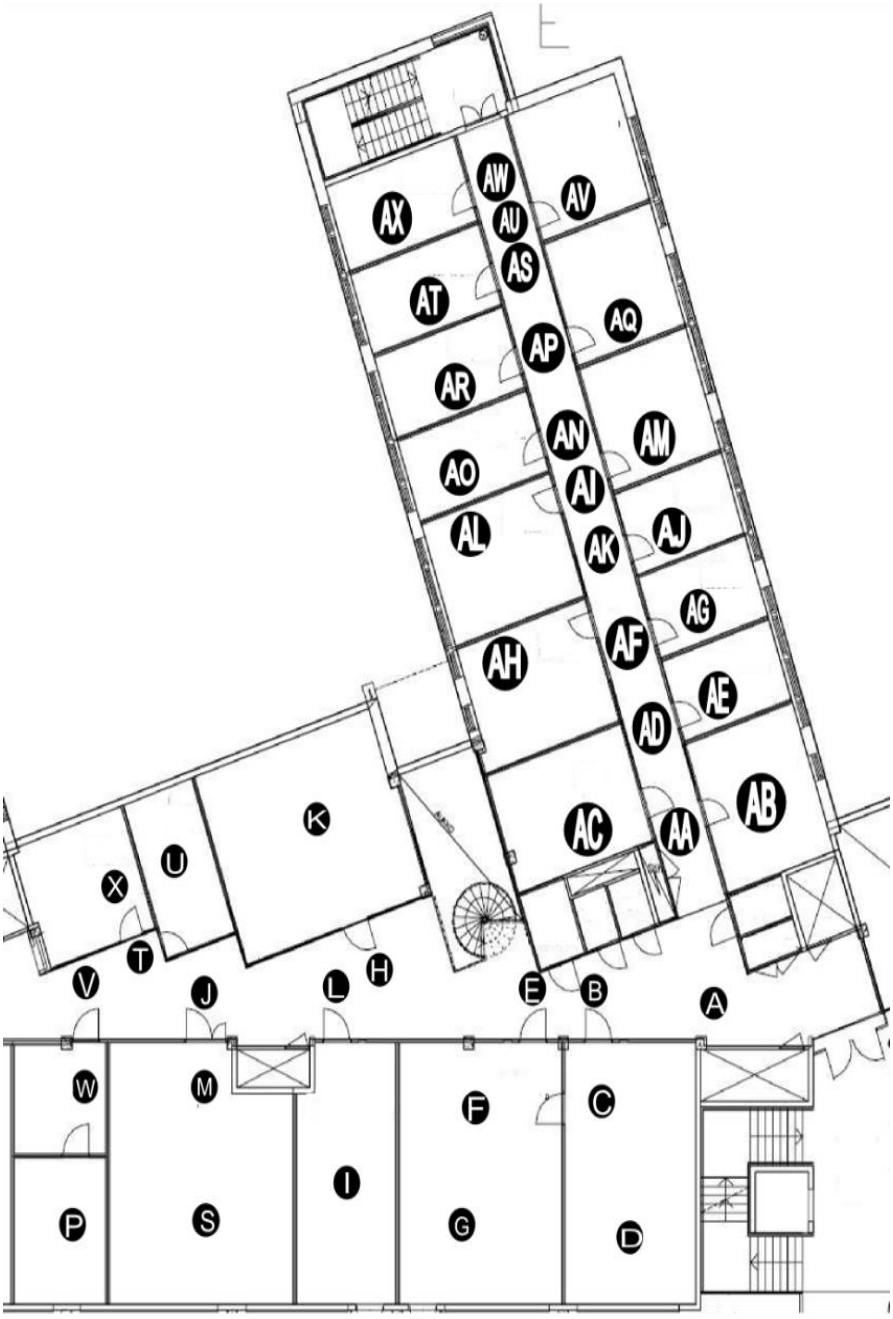
6.3.1 Navigointialgoritmi

Reitit on tallennettu robotin muistiin graafina. Kuvassa 6.2.1.1 näkyy pohjapiirroksesta (kuva 6.2.1.2) tehty graafi. Graafin kaarien painot kuvaavat pisteiden välisiä etäisyyksiä. Lyhimmän reitin etsimiseen käytetään Dijkstran algoritmia. Algoritmi on hollantilaisen Edsger Dijkstran kehittämä graafialgoritmi. Algoritmilla etsitään lyhin reitti graafin yhdestä pisteestä kaikkiin muihin graafin pisteisiin. Algoritmi toimii graafeilla, joiden kaarien painot ovat positiivisia. Algoritmi on hyvin yleisessä käytössä tietojenkäsittelytieteiden alalla. [30,31]

Kuvissa 6.2.1.1 ja 6.2.1.2 näkyvistä pisteistä piste A on robotin lähtöpiste ja pisteet B-AX ovat reittipisteitä, joita käytetään navigoinnissa. Kuvassa 6.2.1.1 pisteiden välissä olevat numerot ovat kaarien painoja, jotka kuvaavat pisteiden absoluuttisia etäisyyksiä rakennuksessa. Etäisyydet ovat metreinä. Painot voivat myös etäisyyksien lisäksi kuvata kuinka haastavia reitit pisteiden välillä ovat robotin kuljettavaksi, mutta tässä tapauksessa jokainen pisteiden välinen reitti on yhtä haastava robotin kuljettavaksi.[30]



Kuva 6.2.1.1 Graafi, joka on tehty kuvan 6.2.1.2 kartasta.

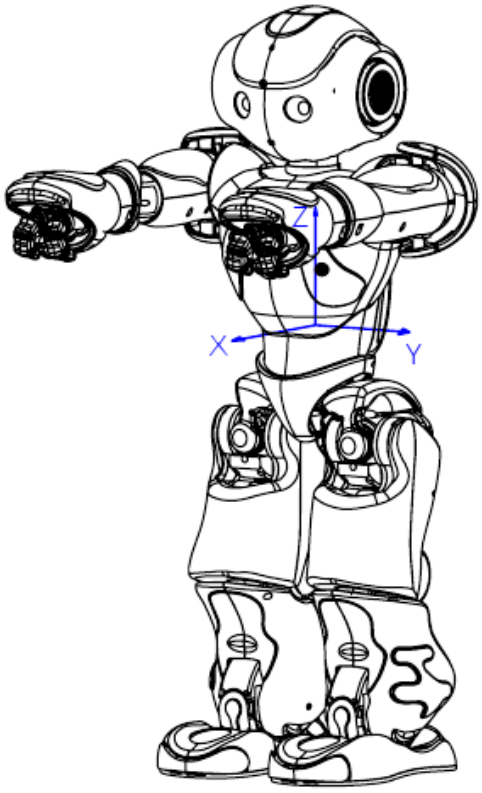


Kuva 6.2.1.2 Esimerkkikartta hoitokodista, jossa robottia voidaan käyttää. A on robotin lähtöpiste ja pisteet B-AX ovat robotin reittipisteitä.

6.3.2 Kävelyn oikaisualgoritmi

Kävelyn oikaisualgoritmin tarkoitus on varmistaa, että robotti kävelisi suoraan navigoidessaan sisätiloissa. On olemassa monia syitä, miksi robotti ei osaa kävellä nativisti suoraan. Suurin syy on robotin sähköisessä rakenteessa, kun robotista puuttuu z-suunnan gyroskooppi. Z-suunnan gyroskoopin avulla olisi helppo havaita robotin poikkeamat kul-

kusuunnasta. Robotin koordinaatisto näkyy kuvassa 6.2.2.1. Robotin natiivikävelyn poikkeama oikeasta kävelylinjasta ei ole järjestelmällinen, joten sitä ei voi korjata vakiokertoimilla. Sen vuoksi kehitettiin algoritmi asian korjaamiseksi.[30,32]

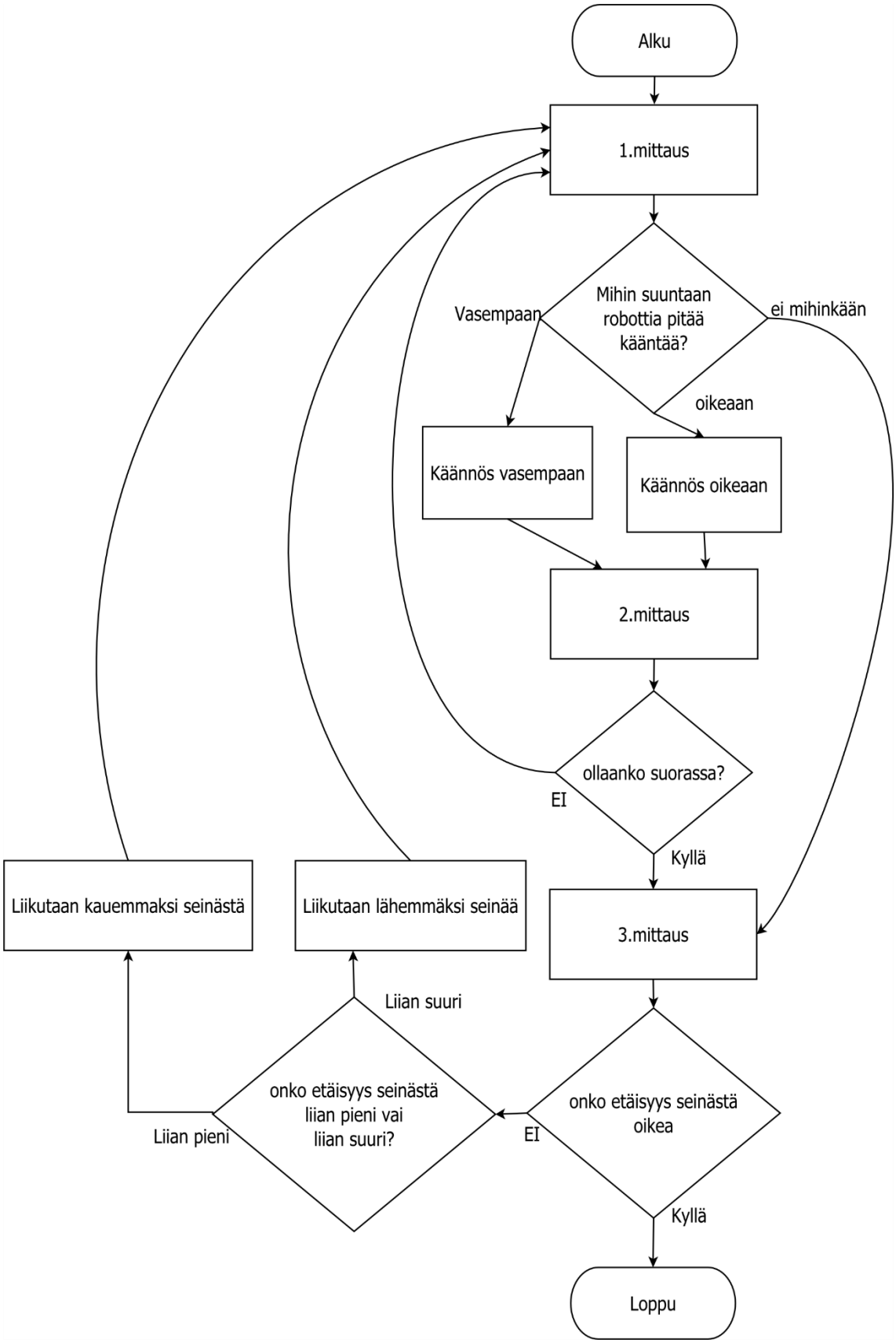


Kuva 6.2.2.1 robotin koordinaatisto

Algoritmi olettaa, että käveltävän käytävän seinät ovat kohtisuorassa robotin kulkusuuntaan nähden. Algoritmissa käytetään etäisyyden mittaamiseen seinästä ja robotin suoristamiseen robotissa olevia ultraääniantureita. Ultraäänianturit sijaitsivat robotin rinnan vasemmassa ja oikeassa reunassa. Anturien tarkka sijainti näkyy kuvassa 2.3. [30]

Kuvassa 6.2.2.2 on oikaisualgoritmin vuokaavio. Vuokaaviosta nähdään, mitä vaiheita algoritmi sisältää ja miten ne liittyvät toisiinsa. Kuvassa näkyvän 1. mittauksen tehtävänä on mitata, onko robotti kohti suorassa seinään nähden. Mittaus tapahtuu vertailemalla robotin rinnassa olevien kahden ultraäänianturin mittaustuloksia toisiinsa, kun anturien tulos on sama, robotti on kohtisuorassa seinään nähden. Kuvassa näkyvän 2. mittauksen tehtävä on mitata, onko robotti oikealla etäisyydellä seinästä ja mittaus suoritetaan ultraäänianturin avulla. [30]

Algoritmin toimivuutta testattiin käytävällä, jossa robotti käveli kymmenen metriä suoraan, minkä jälkeen mitattiin, kuinka paljon robotti poikkeaa kymmenen metrin matkalla oikeasta kävelylinjasta. Testi tehtiin kolmella eri kävelyohjelmalla ja jokainen testi ohjelma suoritettiin kymmenen kertaa.



Kuva 6.2.2.2 Vuokaavio kävelyn oikaisualgoritmista

Ensimmäisessä ohjelmassa oikaisualgoritmi suoritettiin 1,5 metrin välein. Toisessa ohjelmassa oikaisualgoritmi suoritettiin metrin välein. Kolmannessa ohjelmassa oikaisualgoritmia ei suoritettu kävelyn aikana. Taulukossa 6.2.2.1 on esitetty edellä mainitun mittauksen tulokset. Taulukossa olevan sivuttaissuuntaisen virheen mittayksikkö on metri. [30]

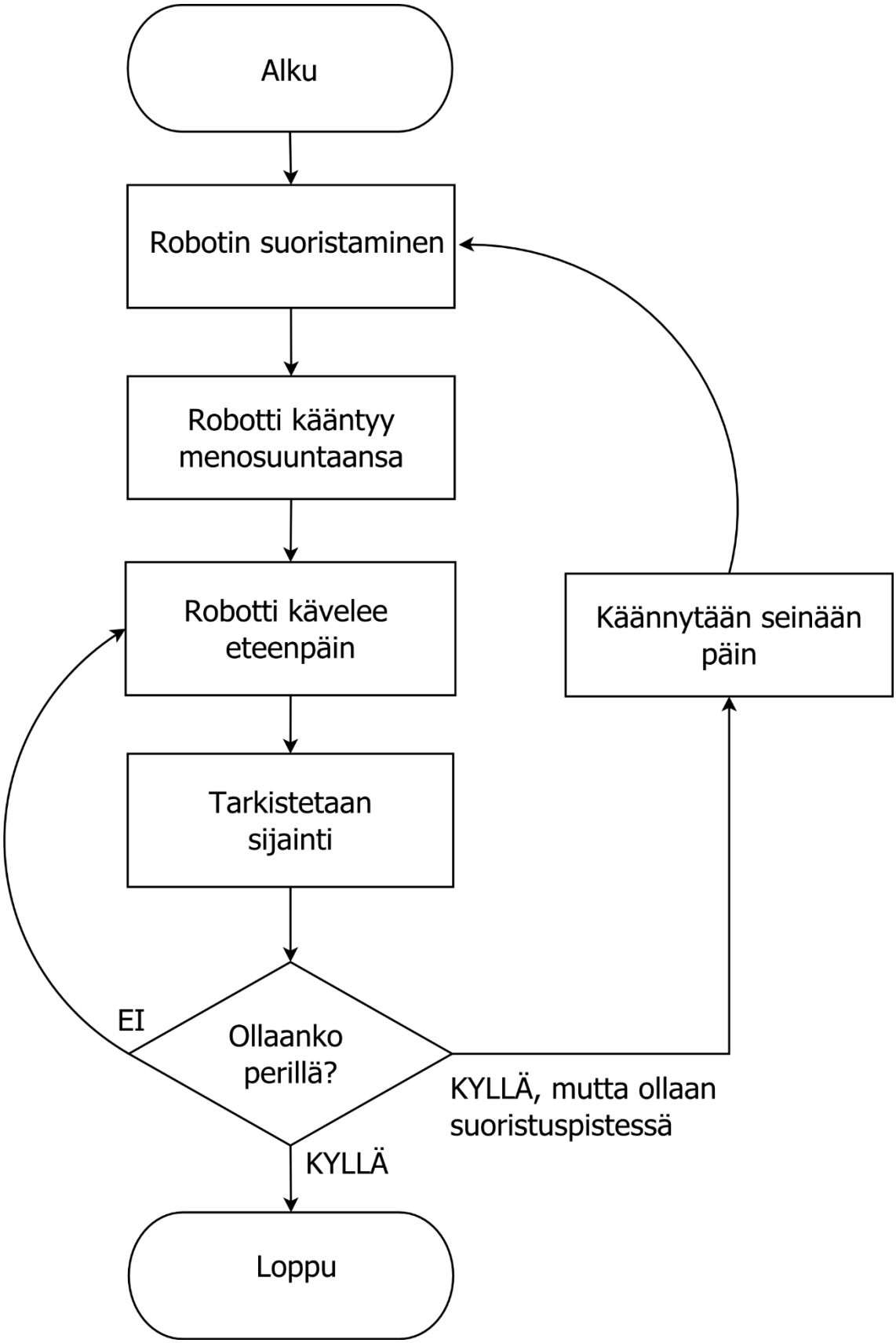
Taulukko 6.2.2.1 Oikaisu algoritmin testauksen mittaustulokset [30]

Mittaus	Oikaisu 1,5 metrin välein	Oikaisu 2,0 metrin välein	Ei oikaisua käytössä
1	0,25	0,3	1,67
2	0	0,05	1,56
3	0,2	0,5	1,88
4	0,1	0,3	1,53
5	0	0,45	1,34
6	0,1	0,25	2,5
7	0	0	1,53
8	0	0,6	2,5
9	0,05	0,5	3
10	0,15	0,55	2,5
Keskiarvo	0,085	0,35	2,001
Mediaani	0,08	0,38	1,78
Keskihajonta	0,09	0,21	0,57

Tuloksesta nähdään, että robotin natiivikävelyn käyttäminen aiheuttaa varsin huomattavan virheen sivuttaissuunnassa jo kymmenen metrin matkalla. Suorittamalla oikaisu algoritmi kahden metrin välein päästään jo 35 senttimetrin tarkkuuteen kymmenen metrin kävely matkalla. Suorittamalla oikaisu algoritmi puolentoista metrin välien päästään jo alle kymmenen senttimetrin tarkkuuteen kymmenen metrin kävelymatkalla. Tuloksesta voidaan päätellä, että suorittamalla oikaisu algoritmin puolentoista metrin välein saavutetaan riittävä kävely tarkkuus moneen tarkoitukseen. [30]

6.3.3 Navigointialgoritmin ja kävelyn oikaisualgoritmin yhdentäminen

Kuvassa 6.2.3.1 näkyy vuokaavion muodossa, miten navigointialgoritmi ja kävelyn oikaisualgoritmi on yhdistetty toisiinsa. Vuokaavio kuvaa yhden kuvassa 6.2.1.1 olevien reittipisteiden välin kulkemista. Eli esimerkiksi robotin siirtymistä pisteestä A pisteeseen B. Vuokaaviossa näkyvä robotin suoristaminen-laatikko kuvaa, missä kohtaa ohjelmaa suoritetaan kävelyn oikaisualgoritmi. Suorituspisteellä tarkoitetaan pistettä, missä oikaisualgoritmi suoritetaan ja suoristuspisteiden etäisyys toisistaan kertoo sen, kuinka usein oikaisualgoritmi suoritetaan. Vuokaaviossa näkyvässä ”tarkistetaan sijainti” -laatikossa tarkistetaan, onko robotti kulkenut pisteiden välisen kaaren painon ilmoittaman matkan vai ollaanko pisteessä suorituspisteessä. [30]

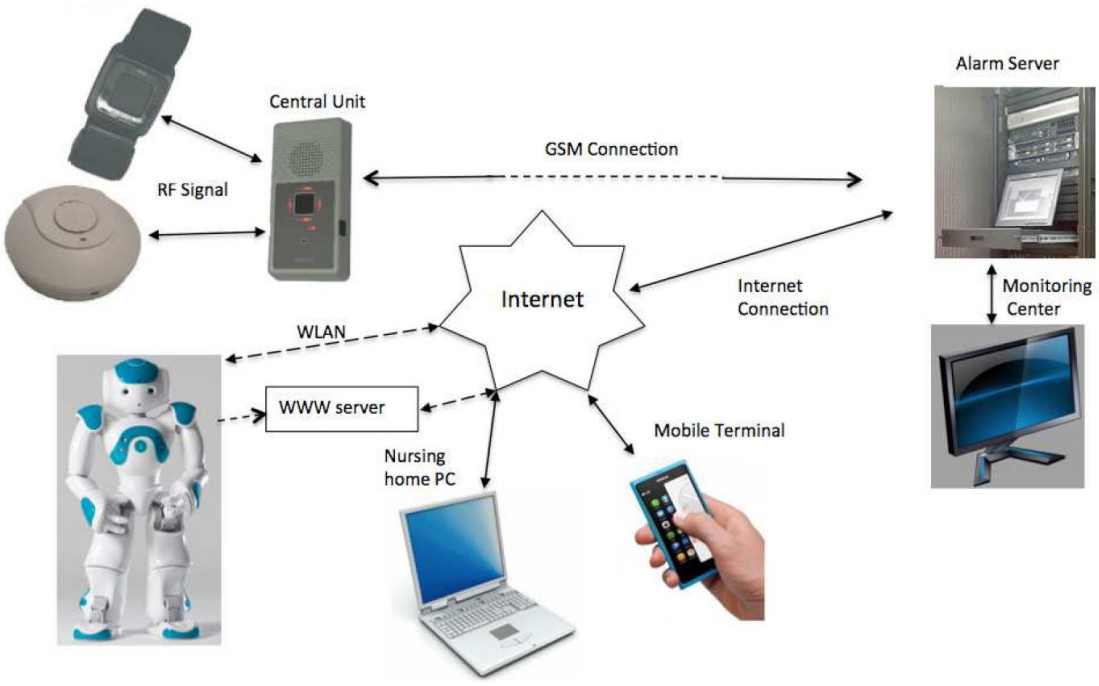


Kuva 6.2.3.1 Vuokaavio Navigointialgoritmin ja kävelyn oikaisualgoritmin yhdistämisestä

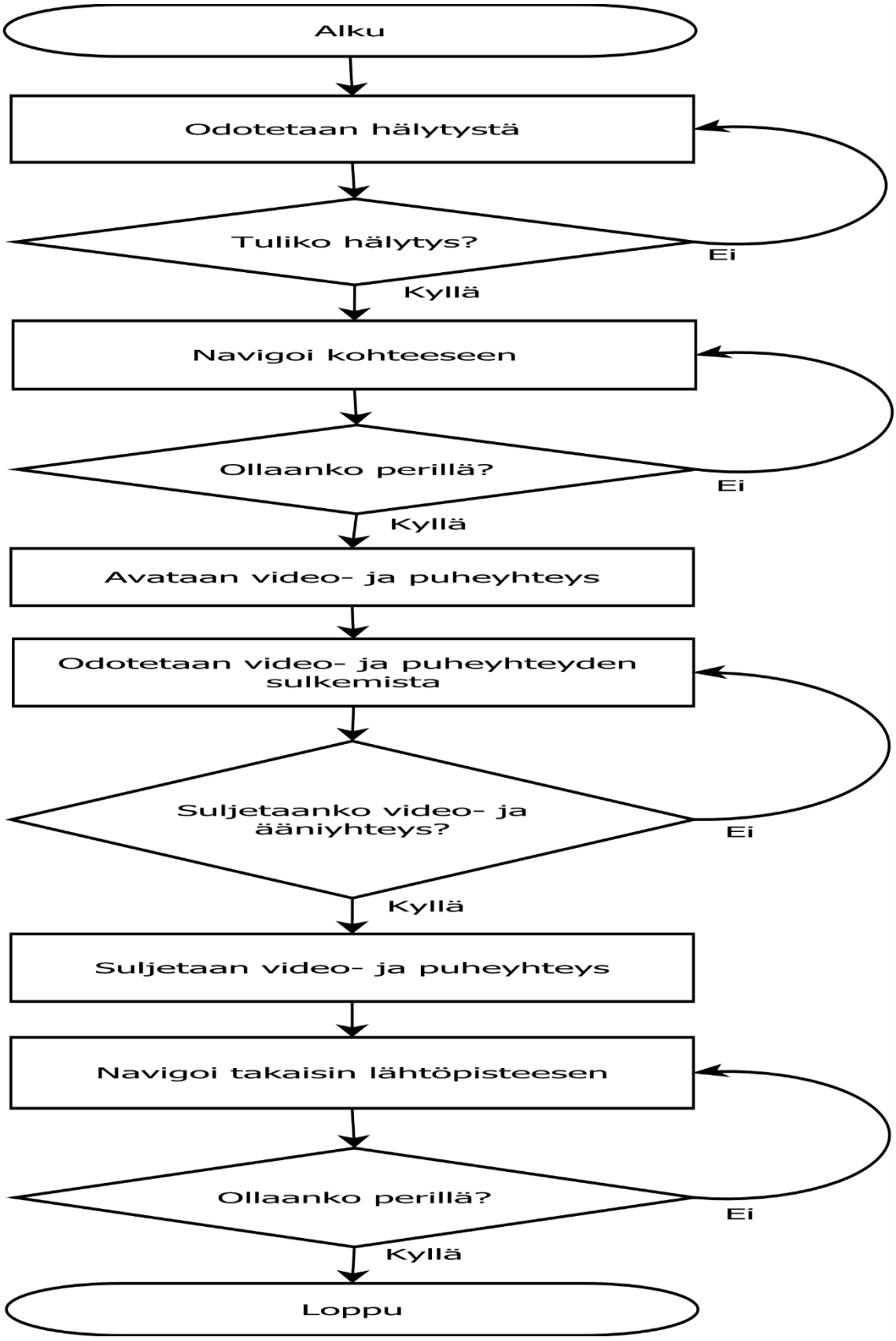
6.4 Valvontasovellus

Valvontasovellus muodostuu kolmesta erillisestä sisätilanavigointiohjelmasta, kuvan välitysohjelmasta, puheen välitysohjelmasta ja hälytysten hakuohjelmasta. Sisätilanavigointiohjelman toiminta on esitelty aliluvussa 6.2, kuvan välitysohjelman toiminta esitellään aliluvussa 6.3.1, puheen välitysohjelman toiminta esitellään aliluvussa 6.3.2 ja hälytyksien hakuohjelma esitellään luvussa 6.3.3.

Valvontasovellutuksen toteuttamiseen käytettiin NAO-robotin sisältämiä instrumentteja (kamera, kaiuttimet ja mikrofonit). Robotissa oleva WLAN-sovitin mahdollistaa puheen ja äänen siirtämisen lähiverkon tai Internetin yli etävalvomoon ja mahdollistaa hälytystietojen hakemisen hälytyspalvelimelta. Kuvassa 6.3.1 on esitelty järjestelmän osat. Hälytysjärjestelmään voi kuulua huoneissa olevia hälytysnappeja, ovihälyttimiä ja rannehälyttimiä, jotka ovat langattomasti yhteydessä keskusyksikköön. Käytetyt hälyttimet olivat Evaronin valmistamia. Ovihälyttimen tyyppi on PL120, rannehälyttimen tyyppi on PL100 ja keskusyksikön tyyppi on TP201. Keskusyksiköstä hälytys siirtyy GSM (Global System for Mobile Communications)-verkkoa siirtotienä käyttäen hälytyspalvelimelle. Palvelin välittää hälytyksen tiedot sovittuun sähköpostilaatikkoon, jota robotti jatkuvasti lukee. Palvelimen välittämässä tiedossa on tarvittavat paikkatiedot, jotta robotti voi navigoida navigointialgoritminsa avulla hälytyspaikalle ja avata kuva ja kaksisuuntaisen puheyhteyden kohteen ja valvomon välillä (kuva 6.3.3).[33,34]



Kuva 6.3.1 Tiedon kulku hälytysjärjestelmässä [33]



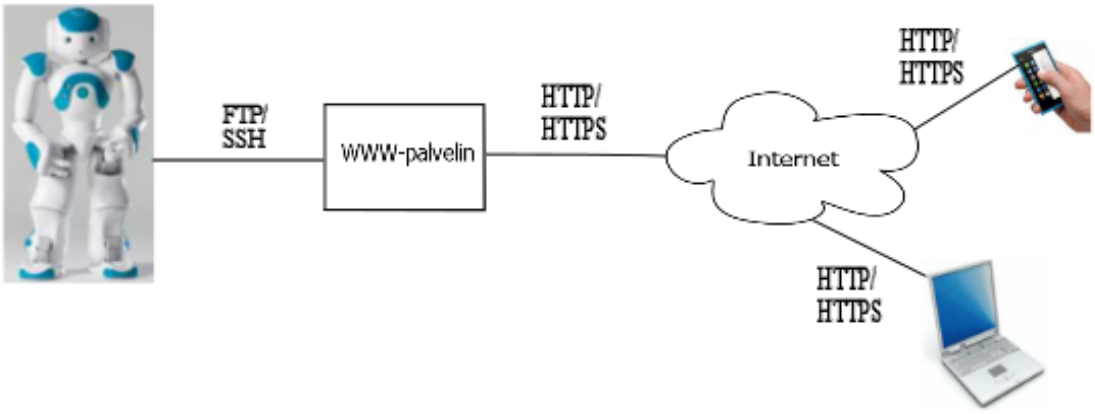
Kuva 6.3.2 Vuokaavio valvontasovelluksen toiminnasta

Kuvassa 6.3.2 on valvontasovelluksen toiminnasta vuokaavio. Vuokaaviosta nähdään, mitä ohjelmia se sisältää ja miten ne liittyvät toisiinsa. Vuokaavion odotetaan hälytystä - laatikossa robotti istuu lepotilassa ja aliluvussa 6.4.3 esiteltävä hälytyksen hakuohjelma odottaa hälytystä. Hälytyksen tullessa käynnistyy aliluvussa 6.4 esitelty sisätilanavigontiohjelma, jonka avulla robotti navigoi paikkaan mistä hälytys on annettu. Kun robotti on navigoinut kohteeseen, se käynnistää kuvan välitysohjelman ja puheen välitysohjelman. Kuvan välitysohjelman toiminta esitellään aliluvussa 6.4.1 ja puheen välitysohjelman toiminta aliluvussa 6.4.2. Tämän jälkeen robotti odottaa sekä kuvan että puheen välitysohjelman sulkemista. Niiden sulkemisen jälkeen robotti käynnistää sisätilanavigointiohjelman ja alkaa navigoida kohti paikkaa, mistä se oli lähtenyt liikkeelle hälytyksen tullessa.

Kuva 6.3.3 Kuvan siirto robotilta tietokoneelle asukkaan huoneesta

6.4.1 Kuvan välitysohjelma

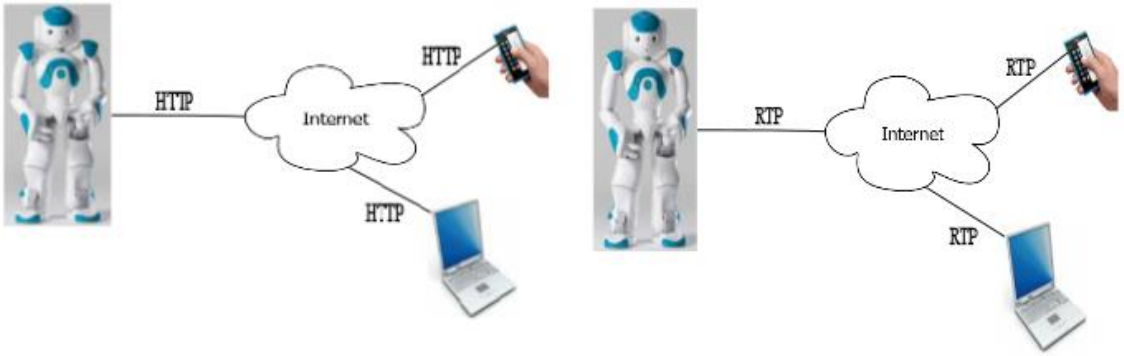
Robotti ottaa kuvia tietyin aikavälein ja tallentaa kuvat muistiinsa, josta ne siirretään Internet-palvelimelle käyttäen joko FTP-yhteyttä (File Transfer Protocol) tai SSH-yhteyttä (Secure Shell). Tällä tavalla saadaan robotin kuvaamat kuvat lähes reaaliaikaisesti Internet-palvelimelle katseltavaksi (kuva 6.3.1.1). Kuvia voidaan katsella palvelimella olevilta Internet-sivuilta millä tahansa selainohjelmalla, jolla on pääsy palvelimen Internet-osoitteeseen. Internet-sivuille on vielä toteutettu JavaScript-koodilla script, jonka tehtävänä on päivittää kuva automaattisesti tietyin aikavälein niin, että Internet-sivulla näkyy viimeisin robotin ottama kuva. Tällä tavalla saadaan robotin ottamat kuvat lähes reaaliaikaisesti Internetiin. [33]



Kuva 6.3.1.1 Robotin ottaman kuvan siirtäminen robotilta WWW-palvelimen kautta päätelaitteeseen

6.4.2 Puheen välitysohjelma

Puheen välitysohjelmassa käytetään hyväksi Naon käyttöjärjestelmässä olevaa multimediakehystä nimeltä GStreamer. Se hoitaa robotin päässä tarvittavat operaatiot, mitkä tarvitaan äänen siirtämisessä robotilta Internetin yli haluttuun kohteeseen ja toisin päin. Puhe välitetään Internetissä robotilta haluttuun kohteeseen käyttäen RTP-protokollaa (Real-time Transport Protocol). RTP-protokollalla välitettyä puhetta voidaan ottaa vastaan lähes kaikilla mediasoittimilla. Toiseen suuntaan puhe liikkuu HTTP-protokolla (Hypertext Transfer Protocol) käyttäen. Kohteessa äänen siirtämiseen robotille liittyvät operaatiot voidaan hoitaa lähes kaikilla mediasoittimilla. Kuvassa 6.3.2.1 on vasemmalla periaatekuva äänen siirtämisestä kohteesta robotille ja oikealla periaatekuva äänen siirtämisestä robotilta kohteeseen. Testaamisessa käytettiin VLCPlayer mediasoitinta, joka on avoimen lähdekoodin ohjelma. [33]



Kuva 6.3.2.1 vasemmalla periaatekuva äänen siirtämisestä päätelaitteesta robotille ja oikealla periaatekuva äänen siirtämisestä robotilta päätelaitteeseen.

6.4.3 Hälytyksen hakuohjelma

Hälytyksen hakuohjelman tehtävä on jatkuvasti lukea sitä sähköpostilaatikkoa, mihin hälytyspalvelin lähettää tiedot mahdollisista hälytyksistä ja etsiä sähköpostista hälytyksen paikkatieto ja välittää ne tiedot robotille. Ohjelma on toteutettu Python-ohjelmointikielellä käyttäen hyväksi kielen valmiita kirjastoja. Valmiita kirjastoja käytetään sähköpostien hakemiseen postipalvelimilta ja sähköpostiviestien jäsentämiseen.

7 OHJELMISTOJEN TESTAUS JA KOE-KÄYTTÖ HOITOKODEISSA

Tässä luvussa luodaan katsaus siihen, miten Naolle tehtyjä ohjelmistoja testattiin ja koekäytettiin. Ohjelmistojen koekäytyö tapahtui kolmessa eteläpohjanmaalaisessa hoitokodissa, joiden henkilökunta oli jo mukana päättämässä, minkälaisia ohjelmia Naolle tullaan toteuttamaan. Hoitokodit ovat hyviä testiympäristöjä, koska Naon ohjelmat oli suunniteltu hoitokotien tarpeita varten.

7.1 Ohjelmistotestaus

Ohjelmistotestaus tarkoittaa töitä, jonka tehtävä on varmistaa, että toteutettu ohjelmisto on toivotun kaltainen, ja että kaikki sen ominaisuudet toimivat niin kuin oli suunniteltu. Testaustyötä voi pitää jatkuvana vertailutehtävänä: testaustyön on tarkoitus havaita ne kohdat, jossa toteutettu ohjelmisto poikkeaa suunnitellusta ohjelmistosta.[35]

Testaus prosessille määritellään yleisesti kolme testauksen päävaihetta, jotka ovat: yksikkötestaus, integrointitestaus ja järjestelmätestaus. Näitä kolmea vaihetta voidaan kutsua testaustasoiksi, koska niissä testaus tehdään eri tasoilla. Seuraavissa kolmessa aliluvussa perehdytään tarkemmin jokaiseen testauksen tasoon. [35]

7.1.1 Yksikkötestaus

Yksikkötestaus on testausprosessin alin taso. Se on samalla yleisin testaustyön vaiheista ja samalla se on ainut testausmenetelmä, jota käytetään kaikissa ohjelmistoja tekevissä organisaatioissa. Yksikkötestauksessa testataan yhden yksittäisen moduulin, funktion tai olion toimintaa. Se tehdään yleensä välittömästi toteutuksen yhteydessä, ja yleensä sen tekee itse ohjelmoijan.[35]

Yksikkötestauksen tehtävä on varmistaa, että toteutettu toiminto tai muutos olemassa olevaan järjestelmään toimii ainakin periaatteessa. Siinä tutkitaan, että tehdyt muutokset menevät käännöksestä läpi virheittä ja että ne toteuttavat funktion toiminnalliset ominaisuudet toimivat oikein, eli komponentti toimii oikein syötteisiin joita sille annetaan. Esimerkkisi yksikkötestiä varten voidaan rakentaa joukko funktio- tai oliokutsuja, joiden tekemiin

kutsuihin komponentin pitää osata vastata oikein tai toteuttaa oikeanlainen toiminto kutsuttaessa. Yksikkötestin epäonnistuessa ohjelmoija tietää, että komponentissa on vikaa ja se on helppo korjata, ennen kuin se ehti käydä osana laajempaa ohjelmaa.[35]

7.1.2 Integrointitestaus

Integrointitestaus on testausprosessin toiseksi alin taso. Se tehdään yksikkötestauksen jälkeen. Integrointitestauksessa järjestelmän eri osia aletaan sovittamaan yhteen tarkoituksena saada lopulta koko järjestelmä toimimaan yhtenä kokonaisuutena. Siinä uusi ohjelmistokomponentti liitetään osaksi järjestelmää, jonka muut osat ovat aiemmin läpäisseet testauksen. Jos järjestelmään tuotavan komponentin kaikkia kytkentöjä muihin järjestelmän osiin ei ole valmiina, niin voidaan korvata rakentamalla testausta varten sijaiskomponentteja, joiden avulla järjestelmä saadaan käynnistymään testausta varten vaikka osa toiminallisuuksista puuttuisikin.[35]

Integrointitestauksen tärkein tavoite on varmistaa, että järjestelmän osat toimivat myös yhdessä. Integrointitestausta varten tehdään testitapauksia, jotka ovat laajempia kokonaisuuksia kuin yksikkötestauksessa, mutta ei vielä täysin koko järjestelmän käyttöä koskevia testejä. Tämän tyylisiä testitapauksia voivat olla mm. moduulien välinen viestivaihto tai samaa tietokantaa käyttävien moduulien yhteistoiminnan varmistaminen.[35]

7.1.3 Järjestelmätestaus

Järjestelmätestaus on testaus prosessin ylin taso, johon siirrytään, kun komponentit ovat läpäisseet testausprosessin kaksi alempaa tasoa, jotka ovat aliluvuissa 7.1.1 ja 7.12 esitellyt yksikkötestaus ja integrointitestaus. Järjestelmätestaus on nimensä mukaisesti sitä testaustyötä, joka tehdään kokonaisella järjestelmällä. Järjestelmätestaus ei tarkoita mitään yhtä testaustapaa vaan se on yleisnimike kaikelle testaukselle, joka tehdään kokonaisille järjestelmälle, jossa ei ole mukana integrointitestauksen aikana tehtyjä sijaiskomponentteja. [35]

Järjestelmätestauksen tavoitteena on varmistaa, että järjestelmä toteuttaa kaikki sille asetetut tavoitteet ja toimii kokonaisuutena. Järjestelmätestauksen aikana testausta suoritetaan testiympäristössä, ei varsinaisessa lopullisessa kohdeympäristössä.[35]

7.2 Tuolijumppaohjelman testaus

Tuolijumppaohjelman testauksen voi jakaa kolmeen eri työvaiheeseen, jotka ovat testaus prosessin kolme tasoa: yksikkötestaus, integrointitestaus ja järjestelmätestaus. Työvaiheet suoritettiin testausprosessin tasojen mukaisessa järjestyksessä, ensimmäisenä suoritettiin alin taso ja sitä sitten edettiin järjestyksessä kohti ylintä tasoa.

Tuolijumppaohjelma oli helppo jakaa erilaisiin komponentteihin, jotka voitin helposti yksikkötestata Choregraphe-nimistä ohjelmaa käyttäen. Choregraphe-ohjelmalla voidaan kääntää Pythonilla tehty komponentti tavukoodiksi, jos komponentti kääntyi ilman virheilmoituksia ja varoituksia, niin voitiin siirtyä yksikkötestaamisen seuraavaan vaiheeseen, joka on koodin ajaminen Choregraphen debuggaus-tilassa. Jos komponentti teki halutut asiat debuggaus-tilassa ajettaessa, voitiin siirtyä komponentin yksikkötestaamisen viimeiseen vaiheeseen, jossa komponentti ajettiin robotissa. Jos komponentti toimi halutusti robotissa, niin komponentti läpäisi yksikkötestin. Nämä kolme vaihetta tehtiin jokaisille ohjelman komponentille.

Tuolijumppaohjelman integrointitestaus tehtiin samaan tyylin kuin yksikkötestaus. Yksikkötestin läpäisseet komponentit lisättiin yksitellen järjestelmän osaksi ja testattiin toimiko komponenttien välinen kommunikaatio halutulla tavalla. Kun kaikki komponentit olivat läpäisseet integrointitestausvaiheen, voitiin siirtyä tuolijumppaohjelman järjestelmätestaukseen.

Tuolijumppaohjelman järjestelmätestaus tehtiin samaan tyylin kuin integrointi- ja yksikkötestaus. Kun komponentit olivat läpäisseet yksikkötestauksen, voitiin siirtyä ohjelman järjestelmätestaukseen. Jos ohjelmisto läpäisi järjestelmätestauksen, ohjelmisto oli valmis koekäyttöön hoivakotoihin. Lisää tuolijumppaohjelmiston koekäytöstä aliluvussa 7.4.

7.3 Valvontasovelluksen testaus

Valvontasovelluksen testausta hankaloitti, että se koostuu hyvin erilaisista osista kuten aliluvusta 6.4 voi havaita. Se pitää sisällään komponentteja ja Linuxin apuohjelmien konfiguroinnista perinteisiin ohjelmistokomponentteihin. Valvontasovelluksen testauksen voi jakaa kolmeen eri työvaiheeseen, jotka ovat testausprosessin kolme tasoa: yksikkötestaus, integrointitestaus ja järjestelmätestaus. Testaukseen toi lisähaastetta se, että valvontasovelluksen testaamiseen ei voinut käyttää Choregraphe ohjelman debuggaus tilaa vaan ohjelmat piti testata suoraan robotissa, koska kaikuluotainten käyttöä ei voitu testata Choregraphe-ohjelmassa. Työvaiheet suoritettiin testausprosessin tasojen mukaisessa järjestyksessä, ensimmäisenä suoritettiin alin taso ja siitä sitten edettiin järjestyksessä kohti ylintä tasoa.

Valvontasovelluksen jakaminen erilaisiin komponentteihin tapahtui lähes automaattisesti johtuen se rakenteesta, kuten aliluvusta 6.4 voidaan nähdä. Komponentit voitiin helposti yksikkötestata. Ohjelmistokomponenttien testaaminen tehtiin ajamalla niitä etäyhteyden yli robotissa, mikä toi aika paljon lisävaikeusastetta testaamiseen. Ohjelmissa mahdollisesti olevat ohjelmointivirheet saattoivat aiheuttaa tilanteita, jossa robotti oli vaarassa rikkoitua. Muiden kuin ohjelmistokomponenttien testaaminen voitiin suorittaa täysin turvallisessa ympäristössä. Sisätilanavigontiohjelmakomponentin yksikkötestausta varten piti rakentaa toimistoon testirata, jossa sen toimivuutta voitiin testata. Samaa testirataa käytettiin integrointi- ja järjestelmätestauksessa.

Valvontasovelluksen integrointitestaus tehtiin samaan tyylin kuin yksikkötestaus. Yksikkötestin läpäisseet komponentit lisättiin yksitellen järjestelmän osaksi ja testattiin toimiko komponenttien välinen kommunikaatio halutulla tavalla Kun kaikki komponentit olivat läpäisseet integrointitestausvaiheen, voitiin siirtyä valvontasovelluksen järjestelmä testaukseen.

Valvontasovelluksen järjestelmätestaus tehtiin samaan tyylin kuin integrointi- ja yksikkötestaus. Jos ohjelmisto läpäisi järjestelmätestauksen, niin sitten ohjelmisto oli valmis koekäyttöön hoivakotoihin. Lisää valvontasovelluksen koekäytöstä aliluvussa 7.4

7.4 Yleistä ohjelmistojen koekäytöstä hoitokodeissa

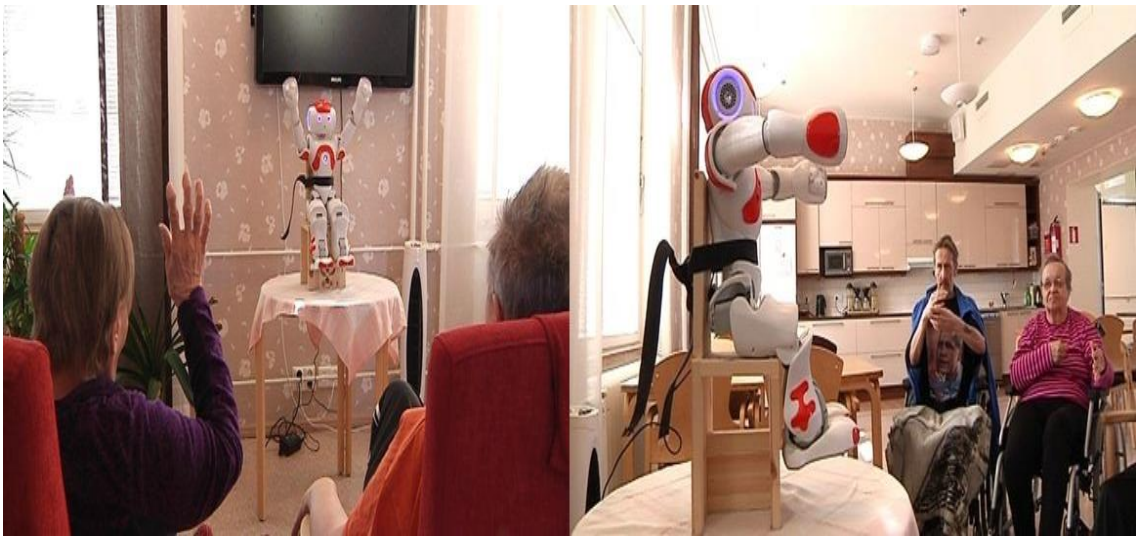
Ohjelmistoja koekäytettiin oikeassa käyttöympäristössä kolmessa hankkeessa mukana olleessa eteläpohjanmaalaisessa hoitokodissa. Hoitokodeissa testattiin tuolijumppaohjelmaa ja valvontasovellusohjelmistoa. Vierailut hoitokoteihin tehtiin 15.11.2011, 13.1.2012 ja 21.5.2012. Kahdella ensimmäisellä käyntikerralla testattiin tuolijumppaohjelmaa ja viimeisellä käyntikerralla testattiin valvontasovellusohjelmistoa ja tuolijumppaohjelmaa. Taulukosta 7.1 näkyy kuinka paljon hoitokodeissa on asukkaita.

Taulukko 7.1 Hoitokotien asukasmäärät [33]

	Hoitokoti A	Hoitokoti B	Hoitokoti C
Asukasmäärä	12	10	22

7.4.1 Tuolijumppaohjelman koekäyttö hoitokodeissa

Tuolijumppaohjelmaa koekäytettiin hoitokodeissa kolmella eri testauskerralla. Kaikissa kolmessa kohteessa hoitokodin asukkaat kokoontuivat yleensä ruokailu- tai oleskelutilaan, jossa robottijumppa järjestettiin (kuva 7.4.1). Hoitokodin hoitajia oli paikalla selittämässä jumpan ideaa asukkaille ja avustamassa heitä jumpassa. Hoitokotien asukkaista osa lähti hyvin mukaan jumppaan, kun taas osa tyytyi vain seuraamaan sitä. Jumppaan aktiivisesti suhtautuvat asukkaat olivat selvästi innostuneita robotista ja pitivät sen käyttöä hyvänä ajatuksena. Robotti herätti mielenkiintoa asukkaissa, ja yleensä siihen suhtauduttiin positiivisesti. Hoitokodeissa oli kuitenkin muutamia asukkaita joiden asenne tuntui olevan torjuva. Henkilökunnan mukaan nämä asukkaat olivat kuitenkin yleensä sellaisia, jotka eivät muutenkaan osallistuneet yhteisiin jumppa- tai muihin virkistystuokioihin. Hoitokotien henkilökunnan palautteella oli suuri merkitys jumppaohjelman kehittämisessä.



Kuva 7.4.1 Robottijumppaa hoitokodissa

7.4.2 Valvontasovelluksen koekäyttö hoitokodeissa

Koekäyttö suoritettiin kahdessa osassa. Ensin navigointia ja puheen sekä kuvan siirtoa kokeiltiin toimistoympäristössä, johon järjestettiin hoitolaitosympäristöä vastaavia tilanteita. Robotti pystyi suunnistamaan huoneisiin, joista hälytys oli annettu, ja avaamaan kuva- ja ääniyhteyden. Toisessa vaiheessa kokeiltiin valvontasovellusta hoitolaitoksissa. Kokeilujen tarkoitus oli selvittää, miten robotti hakeutuu asukkaan huoneeseen hoitokodissa, ja miten äänen ja kuvan siirto onnistuu hoitokodin toimistoon tai muuhun tilaan

sijoitettuun kannettavaan tietokoneeseen. Kokeilut osoittivat, että robotin tiedonsiirto onnistuu hoitokotiympäristössä. Samoin robotin kaiuttimien äänen laatu on riittävän hyvä, jotta vanhukset voivat kuulla hoitajan äänen kaiuttimien välittämänä.

8 TULOKSET, TULEVAISUUDEN KEHITYS-IDEAT JA KATSAUS MUIHIN NAO-TUOTEPER-HEEN HUMANOIDIROBOTTEIHIN

Tässä luvussa luodaan katsaus siihen, minkälaisia tuloksia diplomityössä saavutettiin ja minkälaisia kehitysideoita Naon parempaan hyödyntämiseen vanhustenhuollossa syntyi työtä tehdessä ja luodaan katsaus siihen, mitä muita Nao-tuoteperheen humanoidirobotteja on tuotu markkinoille tai ollaan tuomassa markkinoille

8.1 Tulokset

Työn tekemisen aikana tehtyjen kokeilujen ja testien perusteella voidaan todeta, että robotti soveltuu hyvin erilaisiin aktivointi-, virkistys- ja kuntoutustehtäviin vanhustenhoidossa. Työssä toteutettu tuolijumppaohjelma osoittautui hyväksi tavaksi pilotoida robottia vanhusten kuntoutuksessa. Ohjelmaa on mahdollista muokata niin, että robotti voi toteuttaa muitakin kuntouksen ohjaustehtäviä. Robotin havaittiin sopivan mainiosti myös osaksi hoitokotien valvontajärjestelmää työssä kehitetyn valvontasovellutuksen ansiosta.

Työtä tehdessä havaittiin useita haasteita ja kehittämiskohteita robotin vanhusten huoltosovellukselle. Yhtenä haasteena on robotin sisätilanavigoinnin edelleen kehittäminen tarkemmaksi ja robustimmaksi. Sisätilanavigoinnin laatua voitaisiin parantaa käyttämällä WLAN-paikannusta sekä robotin sisäistä instrumentointia kehittämällä. Haasteena nähtiin Naon pieni koko, joka muodostaa ongelmia erityisesti ovien avaamisen ja liikkumisen suhteen. Kokoa suurempi haaste on kuitenkin robotin melko hidas etenemisnopeus, jonka vuoksi siirtymisaika hoivakotien melko pitkillä käytävillä voi muodostua liian hitaaksi esimerkiksi hälytyksiin vastaamisessa. Haasteena ovat myös Naon melko lyhyt akun kesto ja robotin käyttöliittymä, joka on vielä aivan liian monimutkainen hoitokotien henkilökunnan käytettäväksi.

8.2 Tulevaisuuden kehitysideat

Nao robotin käyttöä vanhustenhuollossa voisi tulevaisuudessa kehittää monella tavalla, jos sen pari perustavaa laatua olevaa ongelmaa saataisiin korjattua. Nämä perustavaa laatua olevat ongelmat ovat heikko akun kesto, hidas kävelynopeus, huono suoraankävelykyky ja robotin käyttöliittymä on vielä liian hankala maallikoiden käytettäväksi.

8.2.1 Naon ominaisuuksien kehittäminen

Heikkoa akun kestoa on vaikea parantaa ilman robotin valmistajan toimia, mutta onneksi valmistajalla on ollut ainakin suunnitelmia asian parantamiseksi. Yksi askel parempaan suuntaan on robotin valmistajan tekemä latausasema, johon robotti osaa automaattisesti navigoida, kun robotin akun lataus taso on laskenut riittävän alas. Ohjelmallisesti akun käyttöä voisi vähentää tekemällä ohjelmat sellaisiksi, etteivät ne pitäisi sähköjä päällä sellaisissa komponenteissa, joita ei tarvita kyseisen ohjelman suorittamiseen. Vapautettaisiin robotin moottori aina, kun siitä ei olisi pakko pitää päällä. Pitäisi tehdä ohjelma joka mahdollistaa yksittäisen moottorin sähköttömäksi kytkemisen.

Naon hidasta kävelynopeutta on vaikea parantaa ilman robotin valmistajan toimia, mutta sitä voisi ehkä parantaa kirjoittamalla Naon kävelyyn liittyvät ohjelmat itse uudestaan. Yksi vaihtoehto hitaan kävelynopeuden aiheuttamien ongelmien ratkaisemiseksi voisi olla jonkinlainen liikkuva alusta, jonka päällä Nao seisoisi tai istuisi, kun sen olisi tarvetta suorittaa pitempiä siirtymiä.

8.2.2 Toteutettujen ohjelmien kehittäminen

Robotin huonoa suoraankävelykykyä olisi sen sijaan helpompi parantaa ohjelmallisesti. Aliluvussa 6.2.2 on esitelty yksi yksinkertainen ratkaisu tähän ongelmaan, mutta sen huonona puolena on, että se hidastaa huomattavasti robotin etenemistä. Tulevaisuudessa suoraankävelykykyä voitaisiin parantaa jollain robotin kameroita hyväksikäyttävällä ohjelmistosovelluksella. Robotin kameroiden hyödyntämiseen suoraankävelykyvyn parantamiseksi on kaksi vaihtoehtoa, jotka ovat kameroiden käyttäminen gyroskooppeina tai mittaamalla kameroiden kuvasta robotin etäisyys kuljettavan käytävän seinistä ja sen avulla määritellä, onko robotti kävellyt suoraan. Jälkimmäisessä vaihtoehdossa pitää olettaa, että kuljettavat käytävät ovat suoria. Mutta molemmista kameroita hyödyntävistä vaihtoehdoista olisi varmasti hyötyä robotin etenemisnopeuden parantamisessa nykyiseen vaihtoehtoon verrattuna.

8.2.3 Käyttöliittymän kehittäminen

Suurin tulevaisuuden haaste Naon käyttämiseksi vanhustenhuollon apuvälineenä on tehdä sen käyttöliittymästä riittävän yksinkertainen ja selkeä, jotta mahdollisesti hoidossa olevat vanhuksetkin voisivat sitä itse käyttää apuvälineenä omissa kodeissaan. Myös hoitajien pitäisi pystyä käyttämään robottia työssään apuvälineenä. Käyttöliittymä pitäisi olla sellainen, että robottia pystyisi käyttämään apuvälineenä ilman pitkää perehdytystä. Robotin käyttöliittymän tulisi olla nettiselaimessa toimiva sovellus, koska silloin sitä olisi helppo muokata toimiva versio erilaisiin päätelaitteisiin, kuten esimerkiksi puhelimeen, tablettiin sekä tietokoneessa toimivaksi versioksi.

8.3 Muut Aldebaran Roboticsin humanoidirobotit

Aldebaran Robotics on kehittänyt Nao-humanoidirobotin ympärille kokonaisen humanoidirobotti tuoteperheen, johon kuuluvat Naon isoveljet Romea ja Pepper. aliluvuissa 8.3.1 ja 8.3.2 esitellään tarkemmin Romeon ja Pepperin ominaisuuksia, kehityshistoriaa ja minkälaiseen käyttöön robotit ovat suunniteltu.

8.3.1 Romeo

Romeo on Aldebaran Roboticsin kehittämä humanoidirobotit, jonka kehittäminen aloitettuun tammikuussa 2009 ja kehitystyön ensimmäinen vaihe valmistui joulukuussa 2012. Ensimmäisen vaiheen tavoitteet olivat: luoda interaktiivinen, avoin ja modulaarinen mekatroniikka- ja ohjelmistoalusta, luoda robotista henkilökohtainen alustaja, jonka avulla voi seurata robotin ja ihmisen välistä vuorovaikutusta, kehittää robusti tutkimusalusta ja perustaa teollinen robotiikkaklusteri. Kehitystyön toinen vaihe alkoi joulukuussa 2012 ja sen on tarkoitus päättyä vuonna 2016. Toisen vaiheen tavoitteet ovat: kehittää tietoturvallinen ja fyysisesti turvallinen robotti, kehittää robotille ohjelmia, joiden avulla robottia voitaisiin käyttää henkilökohtaisena avustajana ja luoda robotille kyky oppia käyttäjän tottumuksia ja tarpeita, jotta se olisi parempi henkilökohtainen avustaja käyttäjälleen. Romeo kuuluu samaan humanoidirobottiperheeseen Naon kanssa. Se on 1,40 m pitkä humanoidirobotti, jonka ohjelmointiin voidaan käyttää samoja ohjelmointityökaluja kuin Naon ohjelmointiin.[36]

8.3.2 Pepper

Pepper on Aldebaran Roboticsin ja SoftBank Mobilen yhteistyön hedelmänä syntynyt humanoidirobotti. SoftBank Mobile on japanilainen teleoperaattori. Se kuuluu samaan humanoidirobottiperheeseen Naon kanssa. Pepper on 1,20 m pitkä humanoidirobotti, jonka ohjelmointiin voidaan käyttää samoja ohjelmointityökaluja kuin Naon ohjelmointiin. Pepperin suurimmat erot Naoon nähden ovat, että se liikkuu pyörillä ja sen rinnassa on 10,1 tuuman suuruinen kosketusnäyttö. Pepper ei ole suunniteltu kotitöitä tekeväksi robotiksi vaan helpottamaan ihmisten kanssakäymistä toisten ihmisten kanssa ja helpottamaan ihmisten yhteyksiä ulkomaailmaan.[37]

9 YHTEENVETO

Diplomityössä tehty työ osoittaa, että robotiikka tarjoaa hoitohenkilökunnalle apuvälineen vanhustenhoidossa. Toisaalta on selvää, että robotti ei tule syrjäyttämään ihmistä hoitotyössä, vaan se toimii lähinnä avustajana ja tarjoaa työkalun turvallisuuden lisäämiseen valvontaan ja kulunseurantaan liittyvissä tehtävissä. Robotti voi olla arvokas apu henkilökunnalle varsinkin silloin, kun henkilöstöä on vähän paikalla (esimerkiksi yöaikaan). Samalla se tarjoaa asukkaille virkistystä ja mielekästä tekemistä silloin, kun henkilökunta ei sitä ehdi tehdä. Hoivakodeissa robotti on saanut positiivisen vastaanoton sekä asukkailta että henkilökunnalta. Vastoin monia ennakkokäsityksiä hoivakodeissa asuvat vanhukset suhtautuivat positiivisesti ja avoimesti robottiin, eikä pelkoja tai ennakkoluuloja juurikaan esiintynyt. Tämä havainto on tärkeä siinä mielessä, että asukkaiden positiivinen asenne tekee robotin käytön helpommaksi ja mielekkäämmäksi hoivakotiympäristössä. Tässä diplomityössä toimintaympäristönä oli vanhustenhoito, johon robotiikan todettiin soveltuvan hyvin. On kuitenkin tärkeää muistaa, että robottisovellusta voidaan käyttää myös monilla muilla sovellusalueilla kuten lasten ja nuorten kuntoutuksessa, mielenterveystyössä, työhyvinvoinnin lisäämisessä jne. Tästä hankkeesta saadut kokemukset rohkaisevat osaltaan robotiikan soveltamista myös näille uusille alueille.

LÄHTEET

[1] Kaisa Backman, Kotona asuvien ikääntyvien huolenpito [WWW]. 2001 [viitattu 20.5.2012]. Saatavissa: http://herkules.oulu.fi/isbn9514259033/isbn9514259033.pdf

[2] Sosiaali- ja terveysministeriö, Väestön ikääntyminen Suomessa [WWW]. 4.2.2001 [viitattu 20.5.2012]. Saatavissa: http://pre20031103.stm.fi/suomi/tao/julkaisut/romppanen/vaesto.htm.

[3] Heli Karppinen, Kuntien vanhustenhuoltopalveluiden kustannukset ja toiminnan tehokkuus [WWW]. 11.2007 [viitattu 20.5.2012]. Saatavissa: http://tutkielmat.uta.fi/pdf/gradu02163.pdf

[4] Vanhusten avohoidon vaikuttavuus, Lääketieteellinen Aikakauskirja Duodecim 1993;109(7):567, Kaisu Pitkälä, Kirsi Kinnunen, Reijo Tilvis viitattu 20.5.2012

[5] Sosiaali- ja terveysministeriö, Kotihoito tukee kotona selviytymistä [WWW]. 10.5.2012 [viitattu 20.5.2012]. Saatavissa: http://www.stm.fi/sosiaali_ja_terveyspalvelut/sosiaalipalvelut/kotipalvelut

[6] Sosiaali- ja terveysministeriö, Terveydenhuolto Suomessa [WWW]. 2004 [viitattu 20.5.2012]. Saatavissa: http://www.stm.fi/c/document_library/get_file?folderId=28707&name=DLFE-3479.pdf&title=Terveydenhuolto_Suomessa_fi.pdf viitattu 20.5.2012

[7] Terveyden ja hyvinvoinnin laitos, Terveydenhuollon menot ja rahoitus 2010 [WWW]. 21.3.2012 [viitattu 20.5.2012]. Saatavissa: http://www.thl.fi/tilastoliite/tilastoraportit/2012/Tr05_12.pdf

[8] [WWW]. [viitattu 19.10.2011]. Saatavissa: http://www.aldebaran-robotics.com/

[9] [WWW] [viitattu 16.4.2015] http://ark.intel.com/products/35460

[10] Aldebaran-robotics, Programming Guide [WWW]. [viitattu: 16.2.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/dev/index.html

[11] Open Source Initiative, Licenses by Name [WWW]. [viitattu 1.2.2012]. Saatavissa: http://www.opensource.org/licenses/alphabetical

[12] Python community, About Python [WWW]. [viitattu 1.2.2012]. Saatavissa: http://www.python.org/about/

[13] Python Software Foundation, Application Domains [WWW]. [viitattu 1.2.2012]. Saatavissa: http://www.python.org/about/apps/

[14] Nokia, Nokia Open Source wiki [WWW]. [viitattu 1.2.2012]. Saatavissa: http://www.developer.nokia.com/Community/Wiki/Nokia_Open_Source

[15] Guido van Rossum, A Brief Timeline of Python [WWW]. 20.1.2009 [viitattu 1.2.2012]. Saatavissa: http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html

[16] Guido van Rossum, PEP 3000 -- Python 3000 [WWW]. 5.4.2006 [viitattu 1.2.2012]. Saatavissa: http://www.python.org/dev/peps/pep-3000/

[17] Python Software Foundation, Python v2.7.3 documentation -- General Python FAQ [WWW]. [viitattu 1.2.2012]. Saatavissa: http://docs.python.org/faq/general#why-is-it-called-python

[18] Bjarne Stroustrup, C++ -ohjelmointi, Suom. Veli-Pekka Ketola. Jyväskylä, Teknolit 2000.

[19] Bjarne Stroustrup,C++ Applications [WWW]. 7.2.2012 [viitattu 16.2.2012]. Saatavissa: http://www.stroustrup.com/applications.html

[20] TIOBE Software, TIOBE Programming Community Index for July 2012 [WWW]. 7.2012 [viitattu 2.8.2012]. Saatavissa: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[21] DedaSys LLC,Programming Language Popularity [WWW]. 13.4.2011 [viitattu 16.2.2012]. Saatavissa: http://www.langpop.com/

[22] Herbert Schildt, C++ The Complete Reference Third Edition, Osborne McGraw-Hill 1998,

[23] Aldebaran-robotics, NAO Software 1.12.5 documentation - NAOqi Framework [WWW]. [viitattu 16.2.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/dev/naoqi/index.html

[24] Kai Koskimies ja Tommi Mikkonen: Ohjelmisto-arkkitehtuurit Talentum ISBN 952-14-0862-6

[25] Aldebaran-robotics, NAO Software 1.12.5 documentation - NAOqi modules APIs [WWW]. [viitattu 16.2.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/naoqi/index.html

[26] Aldebaran-robotics, NAO Software 1.12.5 documentation - Choregraphe User Guide [WWW]. [viitattu 14.3.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/software/choregraphe/index.html

[27] Aldebaran-robotics, NAO Software 1.12.5 documentation - Monitor [WWW]. [viitattu 14.3.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/software/monitor/index.html

[28] Aldebaran-robotics, NAO Software 1.12.5 documentation - NAOsim [WWW]. [viitattu 14.3.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/software/naosim/index.html

[29] Kallio, J., Bäck, I., Salmenaho, A., Perälä, S., Mäkelä, K. A-Robot-Assisted Exercise Program for the Rehabilitation of Older Adults. The 17th Finnish National Conference on Telemedicine and eHealth 2012. Ms Viking Mariella Helsinki- Tukholma-Helsinki.

[30] I. Bäck, J. Kallio, K. Mäkelä, Enhanced Map-based Indoor Navigation System of a Humanoid Robot Using Ultrasound Measurements, Intelligent Control and Automation, Vol. 3, pp. 111-116, 2012.

[31] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, Vol. 1, No. 1, 1959, pp. 269-271. doi:10.1007/BF01386390

[32] Aldebaran-robotics, NAO Software 1.12.5 documentation - Inertial unit [WWW]. [viitattu 15.6.2012]. Saatavissa: http://www.aldebaran-robotics.com/documentation/nao/hardware/inertial.html

[33] I. Bäck, J. Kallio, S. Perälä, K. Mäkelä, Remote Monitoring System for Nursing Home Residents by using a Humanoid Robot, Journal of Telemedicine and Telecare

[34] I. Bäck, J. Kallio, S. Perälä, K. Mäkelä, An assistive humanoid robot for Elderly Care, Submifted to /SG"/SA RC 2012 Conference, Eindhofen, The Netherlands 26.-29.6.2012

[35] Jussi Pekka Kasurinen: Ohjelmistotestauksen käsikirja Docendo ISBN 978-952-5912-99-9

[36] Projet Romeo, Background [WWW]. [viitattu 2.11.2015]. Saatavissa: http://projet-romeo.com/en/background

[37] Aldebaran-robotics, More about Pepper [WWW]. [viitattu 29.10.2015]. Saatavissa: https://www.aldebaran.com/en/a-robots/pepper/more-about-pepper