Haaga-Helia
ammattikorkeakoulu Oy

Raspberry Pi robotin alustana

Matias Räisänen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2018

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä
Matias Räisänen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Raspberry Pi robotin alustana

Sivu- ja liitesivumäärä
30 + 8

Opinnäytetyön otsikko englanniksi
Raspberry Pi as a robot platform

Tämän opinnäytetyön tavoite on tutkia yhden piirilevyn tietokoneita, ja erityisesti Raspberry Pi:n käyttämistä robottiohjelmoinnin alustana, sekä lopuksi suunnitella, rakentaa ja ohjelmoida toimiva robotti.

Alkuun esitellään viisi suosittua yhden piirilevyn tietokonetta: Arduino, Asus Tinker Board, ODROID, LattePanda ja Raspberry Pi. Osio käsittelee laitteiden alkuperää ja teknisiä ominaisuuksia, sekä vertailee niitä keskenään.

Tämän jälkeen tutustutaan ehkä tunnetuimaan yhden piirilevyn tietokoneeseen, Raspberry Pi:hin. Luvussa käsitellään tarkemmin laitteen historiaa ja toimintaperiaatetta, sekä laitteen lisävarustelua erilaisilla, robotiikan kannalta oleellisilla lisälaitteilla.

Viimeiseksi opinnäytetyössä esitellään oman robotin suunnittelu, rakentaminen, ja ohjelmointi Python-ohjelmointikielellä.

Työn tuloksena syntyy katsaus yhden piirilevyn tietokoneisiin, niiden käyttöön robotiikassa, lyhyt tutustuminen Python-ohjelmointikieleen, sekä tietysti produktina itse robotti, joka voi toimia sekä täysin autonomisesti analysoimalla ympäristöään kaikuluotaamalla, tai vaihtoehtoisesti käyttäjän ohjaamana. Tuotokseen voi myös tutustua tarkemmin GitHub-projektissa osoitteessa https://github.com/matiasraisanen/tankbot, josta löytyy robotin lähdekoodi, sekä toimintaa havainnollistavia videoita.

Asiasanat
ohjelmointi, yhden piirilevyn tietokoneet, Raspberry Pi, Python, elektroniikkakomponentit

Sisällys

1 Johdanto 1
2 Yhden piirilevyn tietokoneet 2
2.1 Arduino 2
2.2 ASUS Tinker Board 3
2.3 ODROID 4
2.3.1 ODROID-C2 4
2.4 LattePanda 5
3 Raspberry Pi 7
3.1 Raspberry Pi:n versiot 8
3.2 GPIO-portti 9
3.3 Lisälaitteet 11
3.3.1 Ultraäänisensori 12
3.3.2 Moottoriohjainkortti 12
3.3.3 Kamera 13
3.3.4 Relekortti 14
4 Python 16
4.1 Robotin ohjaaminen Pythonilla 17
5 Robotin toteutus 19
5.1 Projektisuunnitelma 19
5.2 Robotin rakentaminen 19
5.3 Robotin ohjelmointi 22
5.4 Tuotos 24
6 Pohdinta 26
6.1 Jatkokehitys 27
Lähteet 28
Liitteet 31
Liite 1: Sonarbot.py – ultraäänisensorilla ajavan robotin koodi 31
Liite 2: Tankbot.py – näppäimistökomennoilla ajavan robotin koodi 34

1 Johdanto

Yhden piirilevyn tietokoneet, automatisaatio ja robotiikka tuntuu olevan kuuma puheenaihe tänä päivänä. Tämän opinnäytetyön tavoitteena on tutustua yhden piirilevyn tietokoneisiin, Python-ohjelmointikielen käyttöön robotin ohjelmoinnissa ja lopuksi toteuttaa robotti, käyttäen alustana Raspberry Pi:tä.

Tarkemmat tutkimuskysymyksen ovat seuraavat:

- Miten Raspberry Pi:n GPIO-porttia käytetään?
- Miten Raspberryllä pystyy käyttämään sähkömoottoreita?
- Miten muiden sensorien ja lisälaitteiden lisääminen onnistuu?
- Miten Python-kielellä toteutetaan robotti Raspberry-alustalle?

Mikä sitten oikeastaan on robotti? Encyclopedia Britannica määrittelee robotin seuraavasti:

> Robotti on mikä tahansa automaattisesti toimiva kone, joka korvaa ihmisen tekemän työn, vaikka se ei näytä ihmiseltä tai käyttäydy ihmisen tavoin. (Encyclopaedia Britannica Inc. 2006, 1629)

Encyclopedia Britannican määritelmä siis asettaa robotille oikeastaan kaksi vaatimusta: sen täytyy toimia automaattisesti, ja sen tarkoitus on korvata ihmisen tekemä työ. Käytännössä kuitenkin näkisin, että robotin ei tarvitse täyttää näitä vaatimuksia ollakseen robotti. Esimerkiksi poliisin käyttämä pomminpurkurobotti on mielestäni kyllä robotti, vaikka se ei toimikaan täysin autonomisesti, vaan sitä ohjaa ihminen. Ja toisaalta populaarikulttuurissa, varsinkin elokuvissa, robotti voi olla myös sellainen kone, joka ei varsinaisesti korvaa mitään ihmisen tekemää työtä, vaan on lähinnä autonomisesti toimiva kone.

Robottia on siis haastavaa yksiselitteisesti määritellä, vaikka jokaisella varmasti on selkeä kuva mikä on ja mikä ei ole robotti. Minun tavoitteeni on rakentaa ja ohjelmoida robotti, jota voin ohjata etänä, ja jolle pystyn itse ohjelmoimaan toiminnallisuuksia. Keskeisinä käsitteinä on loogisen ja fyysisen maailman yhdistäminen, sekä pienelektroniikkarakentaminen. Haluan toteuttaa ohjelmallisesti ohjattavan, todellisessa maailmassa toimivan laitteen.

2 Yhden piirilevyn tietokoneet

Techopedia (2018) määrittelee yhden piirilevyn tietokoneen seuraavasti:

> Yhden piirilevyn tietokone on sellainen, jossa yhdellä piirilevyllä on muisti, sisään- ja ulostulot, mikroprosessori ja muut tietokoneen toiminnan kannalta oleelliset ominaisuudet. Toisin kuin perinteinen tietokone, sen toiminta ei perustu ulkoisiin laajennusosiin. Yhden piirilevyn tietokone vähentää järjestelmän kokonaiskustannuksia, sillä piirilevyjen ja ohjainpiirien määrä on pienempi.

Arkikielessä käsite on kuitenkin hieman laajempi, sillä moni yhden piirilevyn tietokoneeksi kutsuttu laite vaatii myös jonkin pienen lisälaitteen toimiakseen. Esimerkiksi Raspberry Pi ei toimi ilman muistikorttia, jolta tietokoneen käyttöjärjestelmää ajetaan.

Yhden piirilevyn tietokoneiden kehitys on lähtenyt tarpeesta kehittää jokin helposti lähestyttävä ja edullinen tapa tutustuttaa ihmisiä, etenkin nuoria, ohjelmointiin.

Tässä luvussa esittelen muutamia suosittuja yhden piirilevyn tietokoneita.

2.1 Arduino

Arduino on yksinkertainen yhden piirilevyn tietokone. Se on saanut alkunsa Italiassa, Interaction Design Institute Ivrea -korkeakoulussa, vuoden 2005 tienoilla. Laitteen tarkoitus oli tarjota edullisia ja helppokäyttöisiä laitteita opiskelijoiden käyttöön projekteissaan. (Nussey 2013, 7-16)

Toisin kuin esimerkiksi Raspberry Pi ja perinteiset tietokoneet, Arduinoa ei ole tarkoitus käyttää käyttöjärjestelmän välityksellä. Arduino-koodit kirjoitetaan Arduino Softwarella, joka on Windows, Mac ja Linux -ympäristöissä toimiva kehitysympäristösovellus. Tuotettu koodi ladataan Arduinoon USB-portin kautta. Arduino on siis huomattavasti yksinkertaisempi laite, verrattuna Raspberry Pi:hin. (Nussey 2013, 34)

Nykyään Arduinosta löytyy montaa eri versiota, jokainen vähän omilla ominaisuuksillaan ja käyttötarkoituksillaan. Arduinon termeistä ei ole virallisia suomennoksia, joten suomennan ne tässä kuten parhaiten taidan. Arduino-tuotteet voidaan jakaa tyypeittäin piirikortteihin (boards), moduuleihin (modules), kilpiin (shields), sarjoihin (kits) ja lisälaitteisiin (accessories). Tällä hetkellä kortteja on 27, moduuleita 6, kilpiä 18, sarjoja 3 ja lisälaitteita 3 erilaista. (Arduino 2018)

Eri laitetyypit on myös jaoteltu ryhmiin käyttötarkoituksensa mukaan. Ryhmiä ovat aloittelijataso (entry level), erityisominaisuudet (enhanced features), asioiden internet (internet of things), opetuskäyttö (education) ja puettavat (wearables). (Arduino 2018)

2.2 ASUS Tinker Board

Tinker Board on Taiwanilaisen elektroniikkayritys AsusTek Computer Inc:in valmistama yhden piirilevyn tietokone. Muihin yhden piirilevyn tietokoneisiin se on verrattain uusi; ensimmäinen versio laitteesta, nimeltään Tinker Board, julkaistiin alkuvuodesta 2017.(Liliputing 2017)

Vuonna 2018 ASUS julkaisi laitteesta hieman parannellun version, Tinker Board S:n. Merkittävin uudistus on piirilevylle integroitu 16 gigatavun eMMC-muisti, joka on MicroSD-korttia nopeampi ja luotettavampi tallenusmedia. (LinuxLinks 2018)

Tinker Board on toiminnaltaan ja ulkomuodoltaan hyvin Raspberry Pi 3:sta muistuttava laite. Se on yhden piirilevyn tietokone, jolla on suunniteltu käytettävän Debian-pohjaista Linuxia. Käyttöjärjestelmä asennetaan Tinker Boardin tapauksessa MicroSD-kortille, mutta Tinker Board S tarjoaa mahdollisuuden myös eMMC-muistille asentamiseen. ASUS tarjoaa virallisena käyttöjärjestelmänä TinkerOS:n, mutta laitteella voi tiettävästi ajaa montaa muutakin ARM-arkkitehtuurin käyttöjärjestelmää, kuten mm. Android 6:tta. (Tinker Board Community 2018)

Tinker Boardin mitat ovat 85,6 mm x 56 mm, kun Raspberry Pi 3:n mitat ovat 85,60 mm × 56,5 mm. Tinker Boardin portit (4x USB, HDMI, Ethernet, 3,5mm Audio, GPIO ja virtaliitin) sijaitsevat myös vastaavissa kohdissa piirilevyä kuin Raspberry Pi 3+:ssa. ASUS lupaakin, että Tinker Board on yhteensopiva useimpien Raspberry Pi -koteloiden kanssa (AsusTeK Computer Inc 2017a; AsusTeK Computer Inc 2017b, 2).

Tinker Boardin merkittävimpiä etuja verrattuna Raspberry Pi 3+:aan ovat sen prosessoriteho ja keskusmuisti; Tinker Board S:n ytimessä on 1.8GHz neliydinprosessori, kun Raspberry Pi 3+:ssa on 1.4GHz neliydinprosessori. Tinker Board S:n keskusmuistina on 2GB DDR3-muistia, kun Raspberry Pi 3+:ssa on 1GB DDR2-muistia. Toisaalta Tinker Board S on huomattavasti kilpailijaansa kalliimpi; esimerkiksi Power.fi:n tämänhetkinen (18.9.2018) hinta laitteelle on 89,90 €, kun Raspberry Pi 3+ on samassa kaupassa 34,90 €. (AsusTek Computer Inc 2017a; Raspberry Pi Foundation 2018b)

Taulukko 1. Tinker Board S ominaisuudet

Laite	**Järjestel-mäpiiri (SoC)**	**Proses-soriteho**	**Keskus-muisti**	**Bluetooth**	**WiFi**	**Paino**	**USB**	**Hinta** (power.fi 18.9.2018)
Tinker Board S	Rockchip RK3288	1.8 GHz	2 GB DDR3	Kyllä	Kyllä	55g	4X USB 2.0	89,90 € *

2.3 ODROID

ODROIDit on sarja Etelä-Korealaisen Hardkernel Ltd:n yhden piirilevyn tietokoneita. Nimi on yhdistelmä sanoista Open ja Android. (Bommakanti & Roy 2015, 15)

Ensimmäinen ODROID-laite julkaistiin syksyllä 2009, ja se oli tarkoitettu Android-pelien kehitysalustaksi. Laitteessa oli peliohjaintyyppiset painikkeet ja näyttö integroituna. (ODROID 2018)

Sittemmin Hardkernel on julkaissut lukuisia eri laiteversioita, ja laitteet ovat muuttuneet näytöllisistä pelinkehitysalustoista paljon karsitumpaan suuntaan, missä mukana ei ole enää painikkeita tai näyttöä. Tällä hetkellä uusin ODROID-laite on keväällä 2016 julkaistu ODROID-C2, jonka esittelen tässä luvussa. (ODROID 2018)

2.3.1 ODROID-C2

Taulukko 2. ODROID-C2 ominaisuudet

Laite	**Järjestel-mäpiiri (SoC)**	**Proses-soriteho**	**Keskus-muisti**	**Bluetooth**	**WiFi**	**Paino**	**USB**	**Hinta** (odroid.co.uk 5.9.2018)
ODROID-C2	Amlogic S905	1.5 GHz	2 GB DDR3	Ei	Ei	70g	4X USB 2.0	59,99 €

ODROID-C2 on 64-bittinen yhden piirilevyn tietokone. Sen perustana on ARM-prosessori, ja käyttöjärjestelmänä pystyy käyttämään mm. Ubuntua, Androidia, ARCH Linuxia, sekä lukuisia avoimia käyttöjärjestelmiä. (Bommakanti & Roy 2015, 6)

Laitteessa on vakiona suuri jäähdytyssiili prosessorille, joka pitää huolen siitä, että prosessorin lämpö ei nouse käytössä liian korkeaksi. (Bommakanti & Roy 2015, 13)

Käyttöjärjestelmää käytetään joko erikseen asennettavilta MicroSD-kortilta, tai eMMC-moduulilta. eMMC-moduuli on tyypillisesti matkapuhelimissa käytetty tallennusmedia. (Bommakanti & Roy 2015, 7)

Toimiakseen laite tarvitsee 5V/2A DC -virtalähteen. Virtalähde liitetään piirilevyssä olevaan virtaliittimeen. Vaihtoehtoistesi laitteeseen voi syöttää virran sen micro-USB-portin kautta käyttäen 5V/2A USB-laturia. (Bommakanti & Roy 2015, 4-6)

Näyttöä varten laitteessa on yksi HDMI 2.0 -portti, jonka kautta voi näyttää jopa 4K-kuvaa. Lisälaitteita, kuten hiirtä ja näppäimistöä varten laitteessa on neljä kappaletta USB 2.0 -portteja. (Bommakanti & Roy 2015, 4-6)

Laitteessa on 40-pinninen GPIO-portti (General Purpose Input/Output), jota voi käyttää elektroniikan ja robotiikan kanssa kommunikointiin. (Bommakanti & Roy 2015, 11)

Internet-yhteys hoituu Ethernet-portin kautta, jolla päästään jopa 1 GBps nopeuteen. (Bommakanti & Roy 2015, 8)

Huomattava ero esimerkiksi Raspberry Pi:hin on langattomien yhteyksien puute; WiFi- ja Bluetooth-yhteyden eivät onnistu laitteella vakiona, vaan niitä varten tulee hankkia USB-väylän kautta käytettävä lisälaite. (Bommakanti & Roy 2015, 6-7)

2.4 LattePanda

LattePanda on alkujaan pienen kiinalaisen kehitystiimin Kickstarter-joukkorahoitusprojektina aloittanut yhden piirilevyn tietokone. Sen myyntivalttina oli olla yhden piirilevyn tietokone, jonka käyttöjärjestelmänä on täysi Windows 10. Projekti aloitettiin 2. joulukuuta 2015, ja se keräsi yhteensä 442 735 puntaa 4 060 rahoittajalta. Laitteen hinnaksi oli määritelty $69, mutta siitä oli Kickstarterin hengessä saatavilla monta erihintaista pakettia. Laitteessa on integroitu Arduino-yhteensopiva prosessori, ja siitä löytyy muille yhden piirilevyn tietokoneille ominaisesti myös GPIO-portti. (LattePanda 2018)

Ensimmäiset yksiköt postitettiin asiakkaille 31. maaliskuuta 2016, ja saman vuoden marraskuussa kehitystiimi kertoi saaneensa laitteelle virallisen toimittajan, solmittuaan sopimuksen RS Componentsin kanssa. (LattePanda 2018)

Kehitysaikanaan se kilpaili Raspberry pi 2:n kanssa, mutta Raspberry pi 3 julkaistiin noin kuukausi ennen kuin ensimmäiset LattePanda-yksiköt saatiin toimitettua.
(LattePanda 2018)

LattePandasta on sittemmin tullut uusia, kehittyneempiä versioita, ja tällä hetkellä uusin on LattePanda Alpha 864. Se on tehokas tietokone, mutta se on myös hintansa puolesta

aivan omassa luokassaan; DFRobot myy laitetta hintaan $358, tai aktivoidulla Windows 10 Pro:lla hintaan $398. Nykyään laitteessa on myös tuki Linux-käyttöjärjestelmälle.

Alpha 864:n erityisiä ominaisuuksia on sen tuki laitteen alapuolelle kiinnitettävälle M.2 portin SSD-tallennusmedialle. Kooltaan laite on kuitenkin huomattavasti esimerkiksi Raspberry Pi 3:sta suurempi: LattePanda on 130mm x 90mm, kun Raspberry Pi 3 on 85,60 mm × 56,5 mm. Alpha 864:stä löytyy niin ikään integroitu Arduino-prosessori, GPIO-portti, HDMI-ulostulo sekä WiFi ja Bluetooth yhteydet. Prosessori- ja muistitehoiltaan se on myös edeltäjiään tehokkaampi. (DFRobot 2018)

Taulukko 3. LattePanda ominaisuudet (DFRobot 2018)

Laite	**Järjestelmäpiiri (SoC)**	**Prosessoriteho**	**Keskusmuisti**	**Bluetooth**	**WiFi**	**Koko**	**USB**	**Hinta**
LattePanda Standard	Intel Cherry Trail	1.8 GHZ Quad Core	2 GB DDR3	Kyllä	Kyllä	89mm x 70mm	1x USB 3.0 2x USB 2.0	$69
LattePanda Alpha 864	Core m3-7y30	2.6 GHz Dual Core	8 GB DDR3	Kyllä	Kyllä	130mm x 90mm	3x USB 3.0 1x USB Type C	$358 / $398 Win 10 (dfrobot.com 8.10.2018)

3 Raspberry Pi

Raspberry Pi on Iso-Britannialaisen Raspberry Pi Foundationin valmistama yhden piirilevyn tietokone, jonka isänä voidaan pitää miestä nimeltä Eben Upton. Hän aloitti pienen ja halvan tietokoneen rakentelun vuonna 2006. Toimiessaan pitkään opettajana Cambridgen yliopistossa hän oli huomannut muutoksen uusien opiskelijoiden osaamisessa. Aikaisemmin, 90-luvulla, tietotekniikkaa opiskelemaan tulleilla nuorilla oli lähtötasoisesti jo osaamista ohjelmointikielistä, mutta siinä oli tapahtunut selvä muutos; nykyään opiskelijoiden kokemus tietokoneista oli pintapuolisempaa. Tästä seurasi se, että opintojensa alussa opiskelijoilla meni paljon aikaa ohjelmointikielien opetteluun, kun se oli aikaisemmin ollut jo valmiiksi hallussa ollut taito. (Upton & Halfacree 2016, 3)

Tätä muutosta silmällä pitäen Upton lähti minitietokonettaan kehittämään. Tarkoitus oli kehittää pieni ja halpa tietokone, jonka koulu voisi jakaa uusille opiskelijakandidaateille ilmaiseksi, ja jonka kanssa he voisivat harjoitella ja työskennellä pääsykoetta edeltävän kesän. Kun hakijat myöhemmin tulevat haastatteluun, heiltä kysyttäisiin mihin he ovat tietokonetta käyttäneet. Hakijat, joilla olisi mielenkiintoisimmat projektit, valittaisiin opiskeluohjelmaan. Tarkoituksena oli valmistaa laitteita sata, ja maksimissaan muutama tuhat kappaletta. (Upton & Halfacree 2016, 4)

Upton jätti työnsä yliopistolla, ja siirtyi piirilevyarkkitehdiksi Broadcomille. Siellä hän pääsi tutustumaan lähemmin matkapuhelimissa käytettyihin mikropiireihin, joista hän sai vaikutteita Raspberry Pihin. Sen prosessoriksi valikoituikin myöhemmin ARM-arkkitehtuurin mikroprosessori. Yhdessä viiden kollegansa kanssa Upton perusti Raspberry Pi säätiön, jonka nimissä he alkoivat kehittää projektia eteenpäin. Viidessä vuodessa he saivat aikaseksi ensimmäisen prototyypin tietokoneesta. Vuosien saatossa projektin maine oli levinnyt laajalle, ja tiimille oli selvää, että kysyntä oli paljon suurempi kuin mihin he olivat varautuneet. Laitteen ennakkotilaus oli tarkoitus avata yleisölle helmikuussa 2011. Tiimi oli varautunut toimittamaan 10 000 yksikköä ensimmäisen kuukauden aikana, mutta he saivat jo ensimmäisenä päivänä 100 000 tilausta. Toimitusten mahdollistamiseksi he ottivat yhteyttä paikallisiin elektoriikkavalmistajiin element14:ään ja RS Componentsiin, joiden kanssa he sopivat tietokoneen toimituksesta. Ensimmäisen vuoden aikana laitetta myytiin yli miljoona kappaletta, minkä johdosta Raspberry Pi:stä tuli nopeiten koskaan kasvanut tietokonevalmistaja. (Upton & Halfacree 2016, 4-9)

3.1 Raspberry Pi:n versiot

Raspberry Pi on käynyt elinkaarensa aikana läpi monta kehitysvaihetta, ja siitä on julkaistu eri versioita. Verrattuna toisiin tässä opinnäytetyössä esiteltyihin yhden piirilevyn tietokoneisiin, on Raspberry Pi:stä julkaistu kuitenkin verrattain vähäinen määrä erilaisia versioita.
Laitemalleja on tällä hetkellä seitsemän erilaista: Raspberry Pi 1 Model A+, Pi 1 Model B+, Pi 2 Model B, Pi 3 Model B, Pi Zero, Pi Zero W, ja uusimpana tulokkaana Pi 3 Model B+. (Raspberry Pi Foundation 2018a)

Zero ja Zero W ovat laitteen riisuttuja versioita, joissa on ominaisuuksia karsimalla saatu pienennettyä laitteen kustannusta ja kokoa huomattavasti. Zero W on muuten sama kuin Zero, mutta siinä on sekä Bluetooth- että WiFi-yhteys. (Raspberry Pi Foundation 2018a)

Laitemallin kirjoitusasusta nähdään montaa eri versioita, riippuen yleensä kirjoittajasta, mutta Raspberry Pi säätiön esimerkin mukaan virallinen kirjoitusmalli on seuraava, esimerkiksi uusimman mallin kohdalla: Raspberry Pi 3 Model B+. (Raspberry Pi Foundation 2018a)

Uusimman mallin parannuksina on 1.4 GHz neliydinprosessori, nopeampi verkkoyhteys, sekä mahdollisuus virransyöttöön verkkojohdon kautta PoE-laajannuskortilla.
(Raspberry Pi Foundation 2018a)

Taulukko 4. Raspberry Pi -mallien ominaisuudet

Laitemalli	Pi 1 Model A+	Pi 1 Model B+	Pi 2 Model B	Pi Zero	Pi Zero W	Pi 3 Model B	Pi 3 Model B+
Julkaisu	11/2014	7/2014	2/2015	11/2015	7/2017	2/2016	3/2018
Järjestelmä-piiri (SoC)	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2837	Broadcom BCM2837B0
Prosessori-teho	700 MHz Single Core	700 MHz Single Core	900 MHz Quad Core	1.0 GHz Single Core	1.0 GHz Single Core	1.2 GHz Quad Core	1.4 GHz Quad Core
Keskusmuisti	512 MB	512 MB	1 GB	512 MB	512 MB	1 GB	1GB LPDDR2
HDMI-ulostulo	Kyllä	Kyllä	Kyllä	Kyllä - Mini HDMI	Kyllä - Mini HDMI	Kyllä	Kyllä
Bluetooth	Ei	Ei	Ei	Ei	Kyllä	Kyllä	Kyllä
WiFi	Ei	Ei	Ei	Ei	Kyllä	Kyllä	Kyllä
Paino	23g	45g	45g	9g	9g	45g	45g
Hinta (ThePiHut.com 9.7.2018)	23,11 €	-	39,28 €	5,55 €	10,58 €	-	36,97 €

3.2 GPIO-portti

Raspberry Pi:n yhteys ympäristöönsä tapahtuu GPIO-portin kautta. Yhteydellä en siis viittaa verkkoyhteyteen, vaan laitteen suunniteltuun kykyyn havainnoida ja reagoida ympäristöönsä. GPIO tulee sanoista General Purpose Input Output, joka voidaan vapaasti suomentaa esimerkiksi yleiskäyttöinen sisään- ja ulostuloportti. Sen tarkoitus on lähettää ja vastaanottaa signaaleja. (Upton & Halfacree 2016, 15)

GPIO-portti on laitteen vasemmassa yläkulmassa. Siinä on kaksi 20 pinnin riviä 2,54mm mittaisia urospinnejä, yhteensä siis 40 pinniä. Suurin osa on vain yleiskäyttöisiä sisään- ja ulostulopinnejä, mutta osalla myös oma erityinen käyttötarkoituksensa. (Upton & Halfacree 2016, 207)

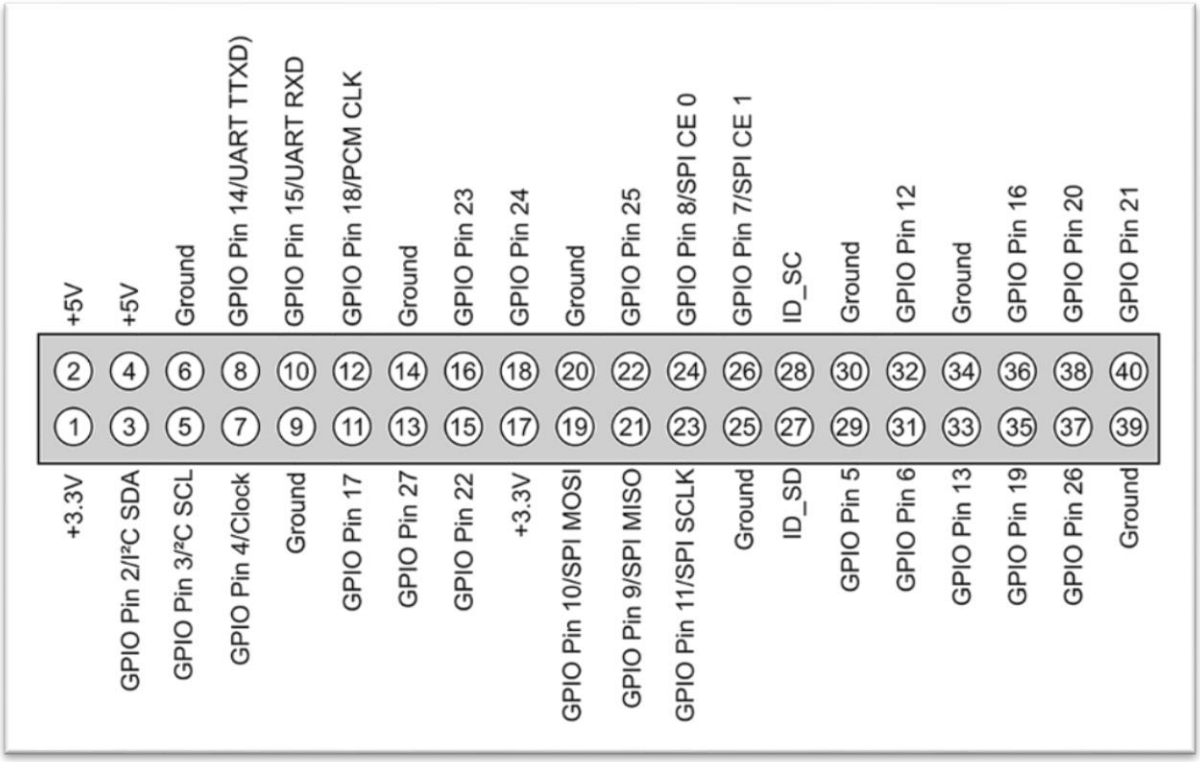
Pinnien numerointia voidaan lukea kahdella tapaa:

1. Pinnin fyysinen sijainti portissa yhdestä neljäänkymmeneen siten, että pinni numero yksi on portin vasemmassa yläkulmassa, ja pinni numero kaksi sen oikealla puolella, ja niin edelleen.

2. Pinnin Broadcom SOC-kanavan numero 0-27. Yli jääville 12 pinnille käytetään tällöin niiden toimintaan perustuvaa nimeä.

Pinneistä puhuttaessa on syytä tehdä erityisen selväksi kumpaa nimeämistapaa niistä käyttää, sillä pinnien vääränlainen yhdistäminen voi johtaa koko laitteen vaurioitumiseen. Yleinen tapa on, että puhuttaessa SOC-kanavanumerosta, käytetään pinninumeroa etuliitteellä GPIO, ja puhuttaessa fyysisestä sijainnista käytetään ainoastaan pinnin numeroa. Esimerkiksi pinni 10 on GPIO 15, ja niin edelleen. (Upton & Halfacree 2016, 208-210)

GPIO-portissa on virransyöttöön liittyen 12 pinniä: kaksi 3.3 V ulostuloa, kaksi 5 V ulostuloa ja kahdeksan maadoituspinniä. Näiden pinnien avulla voidaan syöttää virtaa lisälaitteille, mutta niiden kautta ei pysty kommunikoimaan. (Upton & Halfacree 2016, 209)



Kuva 1. Raspberry Pi GPIO-portti (Upton & Halfacree 2016, 208)

Virtapinnien lisäksi portissa on 28 yleiskäyttöpinniä. Pinnit voidaan asettaa ohjelmallisesti kolmeen eri tilaan: high tai low (ts. output), ja input. High-tilassa pinni lähettää 3.3 V jännitettä ja low-tilassa 0 V. Input-tilassa pinni on nimensä mukaisesti vastaanottavassa tilassa. High- ja low-tilat ovat myös loogisia tiloja, joissa high on tosi ja low on epätosi. (Upton & Halfacree 2016, 210)

Jotkin yleiskäyttöpinnit ovat lisäksi yhteydessä tiettyihin väyliin piirilevyllä. Pinnit numero 8 ja 9 (GPIO 14 ja GPIO 15) kommunikoivat UART sarjaportin kanssa, jota voi käyttää viestin välittämiseen Linux kernelistä. (Upton & Halfacree 2016, 211)

Pinnit numero 3 ja 5 (GPIO 2 ja GPIO 3) kommunikoivat I²C-väylän kanssa. Sitä käytetään, kun halutaan kommunikoida usean IC-komponentin välillä. Näitä ovat esimerkiksi erilaiset Raspberry Pi:n lisälevyt, kuten sähkömoottoriohjaimet. (Upton & Halfacree 2016, 211)

Pinnit numero 19, 21, 23, 24 ja 26 (GPIO 10, 9, 11, 8 ja 7) ovat yhteydessä SPI-väylään. Käyttötarkoitus on sama kuin I²C-väylällä, mutta SPI-väylä on nopeampi. (Upton & Halfacree 2016, 211)

GPIO-pinnit on suunniteltu käytettäväksi kahden ohjelmointikielen kanssa: Pythonin, tai erityisesti lapsille suunnatun visuaalisen ohjelmointikielen Scratchin kanssa. Mikäli ei ole koskaan ollut ohjelmointikielien kanssa tekemisissä, on Scratch varmasti helpommin lähestyttävä, mutta Pythonilla voi rakentaa huomattavasti monipuolisempia projekteja. (Upton & Halfacree 2016, 207-222)

3.3 Lisälaitteet

Raspberry Pihin on saatavilla lukematon määrä lisälaitteita. Yleisimmät lisälaitteet ovat GPIO-porttiin kiinnitettäviä lisälevyjä, jotka ovat siis käytännössä piirilevyjä, jotka on valmistettu tiettyyn funktioon. Lisälaitteita voi valmistaa kuka tahansa, jolla on tarpeeksi ymmärrystä elektroniikkarakentamisesta. Raspberry Pi -säätiö valmistaa myös itse lisälaitteita, mutta varsinaista yhtä virallista lisälaitevalmistajaa ei ole. Tunnettuja kolmannen osapuolen lisälaitevalmistajia ovat Adafruit ja SparkFun.

Raspberry Pi -säätiö on myös julkaissut Github-sivuillaan viralliset ohjeet lisälaitteiden valmistamiseen. Ohjeessa on kuusi kohtaa, joita noudattaen valmistettuja lisälaitteita voidaan kutsutaan nimellä HAT, joka tulee sanoista Hardware Attached on Top. Suomeksi niitä kutsutaan hatuiksi. Yksi hattujen ominaispiirteistä on, että ne ovat pinottavia. Yhteen Raspberry Pi:hin on siis mahdollista liittää monta lisälaitetta samanaikaisesti. (Raspberry Pi Foundation, Add-On Boards And HATs. 2018)

Lisälaitetyyppejä on vaikea luetella, sillä niitä on niin valtava määrä, ja niitä kehitetään jatkuvasti lisää. Tyypillisimmin ne liittyvät esimerkiksi grafiikan näyttämiseen, äänen tuottoon

tai nauhoittamiseen, ympäristön havainnointiin kuten lämpötilan tai kosteuden mittaamiseen, sähkömoottorien ohjaamiseen, ja niin edelleen. Lisälaitteena on myös saatavilla GPS-paikannusominaisuus, sekä 3G-modeemi, jolla laitteen saa matkapuhelinverkkoon. Lisälaitteista kiinnostuneen kannattaa selailla esimerkiksi Adafruitin (https://www.adafruit.com/categories) tai Sparkfunin (https://www.sparkfun.com/categories) verkkokauppojen sivuja, joilta löytyy tuhansia lisälaitteita.

3.3.1 Ultraäänisensori

Yksi lisälaite, jota aion projektissani käyttää, on HC-SR04 ultraäänisensori. Se mittaa etäisyyttä edessään oleviin esteisiin käyttämällä ultraääntä. Laitteessa on kaksi ultraäänilähetintä, vastaanotin ja ohjauspiiri.
Laitteelle annetaan ohjelmallisesti signaali, jonka seurauksena lähettimet lähettävät ultraäänen. Ultraääni kaikuu takaisin laitteelle osuessaan esteeseen, ja vastaanotin havaitse palaavan signaalin. Signaalin lähetyksen ja vastaanoton ajankohdan erotuksesta, tietäen äänennopeuden, voi laskea etäisyyden esteeseen, joka kimmokkeen aiheutti. (The Pi Hut 2018)

Kuva 2. HC-SR04 ultraäänisensori (The Pi Hut 2018)

Sensorin toimintakulma on 15°, ja se pystyy mittaamaan 2cm – 400cm etäisyyksiä jopa 3mm tarkkuudella. (Sparkfun 2018)

Sensori on myös hyvin edullinen lisälaite; esimerkiksi englantilainen ThePiHut.com myy sitä kirjoitushetkellä hintaan 2,29€, yhdysvaltalainen SparkFun.com 3,94$ ja suomalainen ihmevekotin.fi hintaan 5,90€.

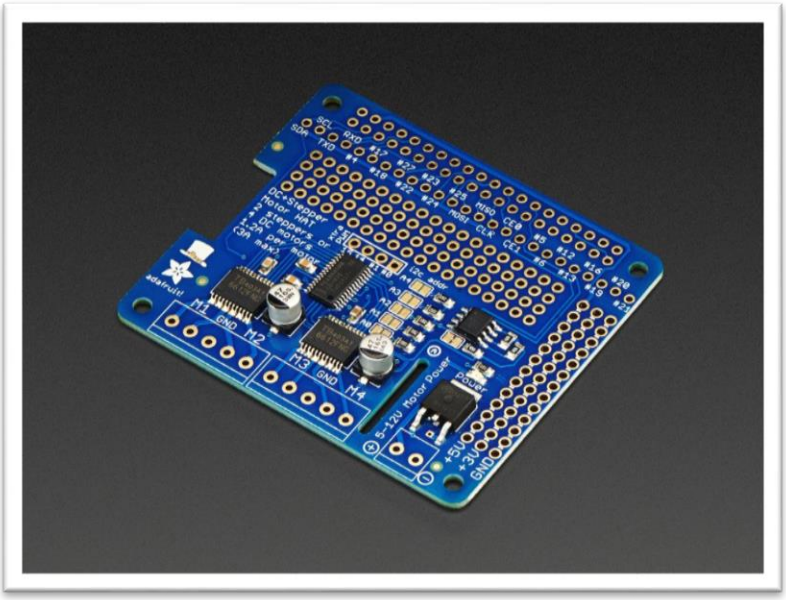
3.3.2 Moottoriohjainkortti

Raspberry Pi:tä voi käyttää myös sähkömoottorien ohjaamiseen, mutta siihen tarkoitukseen kannattaa käyttää erillistä ohjainkorttia, eli hattua. Sähkömoottoria ei ole turvallista liittää suoraan GPIO-pinneihin, sillä se voi aiheuttaa laitteeseen virtapiikin, joka vahingoittaa laitetta. (Molloy 2016, 407-409)

Sähkömoottoreita varten on olemassa monia erilaisia ohjainkortteja. Ohjainkortti asennetaan Raspberry Pi:n GPIO-porttiin joko juottamalla tai irrotettavalla liittimellä, ja sähkömoottorit kytketään ohjainkorttiin. Joissain ohjainkorteissa on lisäksi mahdollisuus ulkoiselle virtalähteelle, minkä käyttö onkin suositeltavaa sähkömoottorien korkean virrankulutuksen takia. (Molloy 2016, 410-413)

Esimerkkinä moottoriohjainkortista on Adafruitin valmistama DC & Stepper Motor HAT, jolla voi nimensä mukaisesti ohjata tasavirta- ja askelmoottoreita. Kortilla on tilaa kerrallaan neljälle tasavirtamoottorille, tai vaihtoehtoisesti kahdelle askelmoottorille. Kortti kytketään suoraan GPIO-porttiin erityisellä liittimellä. Kortin mukana toimitetaan liitin, jossa on 3mm pituiset pinnit. Mikäli kuitenkin haluaa hyödyntää mahdollisuutta pinota kortteja päällekkäin tai käyttää GPIO-porttia muillekin laitteille, kannattaa kortti yhdistää pidempipinnisellä, esimerkiksi 10mm pinneillä varustetulla liittimellä. Näin GPIO-portin pinnit jäävät edelleen niin pitkiksi, että niitä on helppo jatkokäyttää. Kortti käyttää kommunikointiin I²C-väylää, ja siten vain kahta GPIO-portin pinniä. Väylään voi samanaikaisesti kytkeä muitakin I²C-laitteita; Adafruitin mukaan esimerkiksi moottoriohjainkortteja voi pinota päällekkäin jopa 32 kappaletta, ja siten ohjata kerralla aina 128 tasavirtamoottoria. (Adafruit 2015)

Kun ohjainkortti on kytketty GPIO-porttiin, onnistuu sen käskyttäminen ohjelmallisesti esimerkiksi Pythonilla. Ohjainkorttien käytön tarkemmat ohjeet ja esimerkkikoodi on saatavissa kortin valmistajalta. (Adafruit 2015)



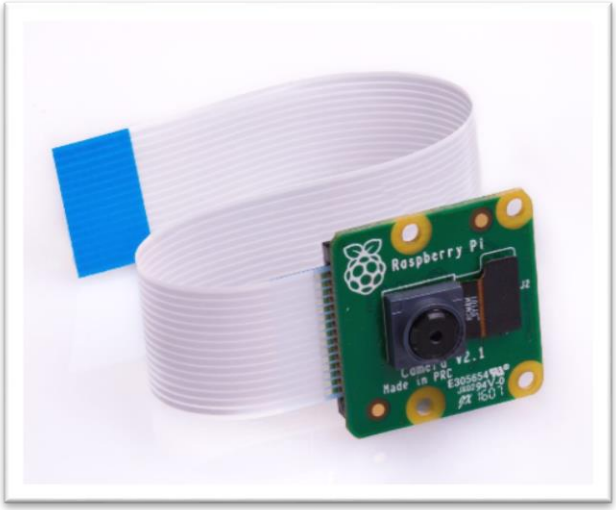
Kuva 3. Adafruit DC & Stepper Motor Hat (Adafruit 2018)

3.3.3 Kamera

Raspberry Pi:lle on tarjolla lukuisia eri kameramoduuleita, mutta Raspberry Pi -säätiö on julkaissut laitteelle myös viralliset kameramoduulit. Tällä hetkellä moduuleita on kaksi erilaista: Camera Module V2, ja pimeänäkökykyinen Pi Noir Camera V2. Kamerat liitetään lattakaapelilla piirilevyn kameraporttiin. (Raspberry Pi Foundation 2018a)

Kameramodulit ovat keskenään lähes identtiset, mutta pimeänäköversiossa kamerassa ei ole infrapunavalon suodatinta. Tällä tavalla, mikäli pimeä tila on valaistu ihmissilmälle näkymättömällä infrapunavalolla, pystyy kamera näkemään tilassa vaivatta. (Raspberry Pi Foundation 2018a)

Kameroissa on kahdeksan megapikselin sensori, joka pystyy ottamaan 3280 x 2464 pikselin valokuvia, sekä 1080p-tasoista videokuvaa. Kamerat ovat myös hyvin pieniä, kooltaan 25mm x 23mm x 9mm. (Raspberry Pi Foundation 2018a)



Kuva 4. Camera Module V2 (Raspberry Pi Foundation 2018a)

Kameroiden käyttöön säätiö on julkaissut nettisivuillaan helposti seurattavat ohjeet. Kameraa voi käyttää joko komentoriviltä raspicam- ja raspivid-komentorivityökaluilla, ja kameran käyttöön on myös erillinen Python-kirjasto. (Raspberry Pi Foundation 2018d)

Kameran käyttöön on myös kolmansien osapuolien julkaisemia työkaluja. Yksi mielenkiintoinen UV4L-niminen työkalu, joka luo verkkopalvelimen, johon se suoratoistaa kameran lähettämää kuvaa. Tämä on hyödyllistä esimerkiksi silloin, kun Raspberryyn ei halua yhdistää näyttöä, vaan näyttönä voi näin toimia mikä tahansa muu saman verkon laite. (Linux Projects 2018)

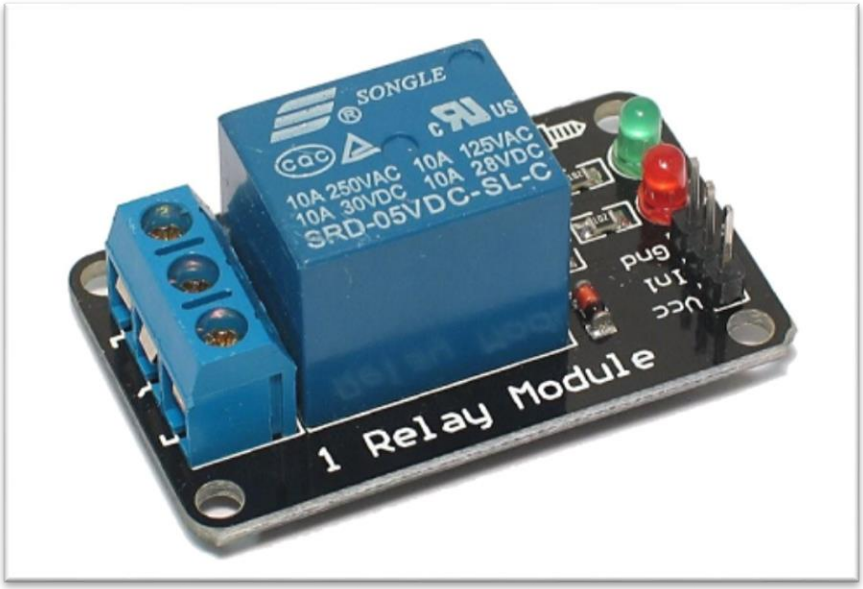
3.3.4 Relekortti

Raspberry Pi:llä on myös mahdollista hallita laitteita, jotka ovat virrantarpeensa puolesta niin vaativia, että niitä ei voi kytkeä suoraan Raspberryyn. Tällaisia ovat esimerkiksi verkkovirtaan kytkettävät laitteet, mutta käytännössä minkä vain laitteen on/off-toiminnallisuutta voi hallita kytkemällä Raspberryyn relekortin.

Relekortteja valmistetaan eri relelukumäärillä. Esimerkkinä käytän yksinkertaista, yhden releen relekorttia 5V DC käyttöjännitteellä. Kyseistä relekorttia myy useat elektroniikka-

komponenttikaupat. Yhden releen relekortissa on kuusi liitäntää, kolme kummallakin puolella korttia. Kortin toinen pää kytketään Raspberryyn, ja toinen pää hallittavaan laitteeseen. Ohjaavan laitteen pään virtapinnit kytketään Raspberryn 5V ulostuloon (pinnit numero 2 tai 4), maahan (pinnit numero 6, 9, 14, 20, 25, 30, 24 tai 39). Kolmas pinni on ohjaussignaalia varten, ja se kytketään Raspberryn vapaavalintaiseen GPIO-pinniin. (Aqib 2018)

Hallittava sähkölaite kytketään relekortin toiseen päähän. Hallittavan laitteen päässä on yleensä niin ikään kolme liitäntää. Nämä ovat Common (C), Normally open (NO) ja Normally closed (NC). Nimensä mukaisesti NO-tilassa virtapiiri on normaalitilassaan auki, mutta sulkeutuu kun sille antaa ohjaussignaalin, ja NC-tilassa toiminta on päinvastainen. Ulkoinen laite kytketään käyttötarkoituksen mukaiseen liitäntään. Tavanomaisin käyttötarkoitus lienee, että releellä kytketään jotain päälle, ja tällaisessa tapauksessa ulkoisen laitteen virransyöttö kytketään siis C- ja NO-liitäntöihin. Kun ohjauslaitteen pää antaa käskyn, virtapiiri sulkeutuu, ja ohjattava laite kytkeytyy päälle. (Aqib 2018)



Kuva 5. Relekortti (Partco 2018)

4 Python

Pythonista puhutaan usein yhtenä helpoiten opittavista ohjelmointikielistä. Sen kehitti vuonna 1990 hollantilainen mies nimeltä Guido van Rossum, joka on yhä mukana kielen kehityksessä. (Guttag, 2013, 8.)

Pythonista on nykyään käytössä kaksi eri versiota; Python 2.7 ja Python 3.xx (kirjoitushetkellä tuorein versio on 3.7.0.). Python 3 julkaistiin vuonna 2008, ja siinä korjattiin monia aikaisemman version epäjohdonmukaisuuksia. Versio ei kuitenkaan ole taaksepäin yhteensopiva, ja koska monia Python-kirjastoja ei ole vieläkään käännetty uusimpaan versioon, on kielestä yhä laajalla käytöllä kaksi eri versiota. On suositeltavaa käyttää kielen tuoreinta versiota jos vain mahdollista, mutta yhteensopivuusongelmien takia Python 2 on myös varteenotettava vaihtoehto. (Guttag, 2013, 8.)

Python on tulkattava kieli, tarkoittaen siis sitä, että tulkki kääntää ja ajaa koodin suoraan ja vapaasti, sen sijaan että koko ohjelma käännettäisiin ensin kokonaisuudessaan konekielelle (Guttag, 2013, 7).

Keskeinen asia kielessä on objektit, joilla kaikilla on tietty tyyppi. Tyyppi määrittelee, kuinka ohjelma käsittelee objekteja. Tyypit ovat joko skalaareja, tai non-skalaareja. Pythonissa on neljä skalaarista tyyppiä, jotka ovat kokonaisluvut int, reaaliluvut float, totuusarvomuuttuja boolean ja yksiarvoinen None-tyyppi. (Guttag, 2013, 9.)

Muita, non-skalaarisia, tietotyyppejä kielessä ovat listat (list), monikot (tuple), merkkijonot (string), sanakirjat (dictionary), sarjat (set) sekä tiedosto-objektit (file). Pythonissa listat eivät ole sidottu yhteen tiettyyn tietotyyppiin, vaan ne voivat sisältää sekoituksen mitä tahansa tietotyyppejä. Monikot taas muistuttavat hyvin paljon listaa, kuitenkin sillä erolla, että monikon arvoja ei voida enää muuttaa objektin luomisen jälkeen. Sanakirjat koostuvat avain-arvopareista, ne muistuttavat siis esimerkiksi Javascriptin objekteja. Sanakirjat on kätevä tapa kerätä yhteen tiettyjä ominaisuuksia, ja niiden arvoja. Sarjat ovat kokoelma objekteja, joilla ei ole sisäistä järjestystä. Ne ovat kuin sanakirja, missä on vain avaimia, ei arvoja. Tiedosto-objektit ovat nimensä mukaisesti tiedostoja. Python voi avata tiedostoja objekteiksi, ja lukea ja kirjoittaa niihin. (Ceder 2010, 19-25)

Objekteja voidaan nimetä muuttujilla. Muuttujat ovat ohjelmointikielille tyypillisesti merkkikokoriippuvaisia; esimerkiksi muuttujanimet koira ja Koira osoittavat eri muuttujiin. Muuttujien nimet voivat sisältää isoja ja pieniä kirjaimia, numeroita, ja alaviivan. Lisäksi kielessä

on tietty lista varattuja muuttujanimiä, joita ei tule käyttää muuhun tarkoitukseen. (Guttag, 2013, 9-11.)

Pythonille ominaista on koodin sisennysten merkitys; sen sijaan että kieli käyttäisi esimerkiksi hakasulkeita määrittelemään kontrollirakenteita, määritellään rakenteet sisentämällä ne tyypillisesti neljän välilyönnin verran (Guttag 2013, 15).

4.1 Robotin ohjaaminen Pythonilla

Robotin ohjaamisessa on pohjimmiltaan kyse käskyn antamisesta, ja sen välittämisestä jollekin instrumentille, joka suorittaa käskyn mukaisen toiminnon. Tästä syystä robotin ohjaamiseen on monia lähestymistapoja. Yksi asia on kuitenkin selvä: loogisen maailman ohjelmakoodin käskyt täytyy jotenkin välittää fyysiseen maailmaan elektronisiksi signaaleiksi.

Monimutkaisiin robottiprojekteihin on olemassa erityisiä työkaluja, jotka on nimenomaan tarkoitettu robotiikan kanssa työskentelyyn. Esimerkki tällaisesta työkalusta on esimerkiksi ROS, Robot Operating System. Se on ilmainen, avoimen lähdekoodin käyttöjärjestelmä, jonka etuja ovat kieliriippumattomuus, helppo testaus, skaalautuvuus, koodin uudelleenkäyttö, ja mahdollisuus jakaa toimintoja eri tietokoneille, vähentäen näin yksittäiselle koneelle aiheutuvaa kuormaa. Joseph Lentin perehtyy kirjassaan Learning Robotics Using Python syvemmin ROS:n ja Pythonin käyttämisestä robotiikassa. (Joseph 2015, 54-55)

Yksinkertaisen robotin ohjaamiseen riittää yksinkertaisemmat ratkaisut. Raspberry Pi:n tapauksessa yksinkertaisin ratkaisu on liittää ohjattava instrumentti piirilevyn GPIO-porttiin, ja käyttää RPi.GPIO-kirjastoa välittämään komentoja instrumentille portin kautta. RPi.GPIO on Ben Crostonin ylläpitämä Python-kirjasto, jonka avulla voi ohjelmallisesti käyttää Raspberry Pi:n GPIO-porttia.

Alla esimerkki koodista, joka lähettää signaalin pinniin numero 7 (GPIO4), odottaa sekunnin, ja katkaisee signaalin. Sitä voisi esimerkiksi käyttää sytyttämään LED-valon, käynnistämään sähkömoottori, tai tekemään jotain muuta, mikä tarvitsee toimiakseen vain yksinkertaisen signaalin.

```
1.  import RPi.GPIO as GPIO
2.  import time
3.
4.  GPIO.setmode (GPIO.BOARD)
5.  GPIO.setup(7, GPIO.OUT)
6.  GPIO.output(7, GPIO.HIGH)
7.  time.sleep(1)
8.  GPIO.OUTPUT(7, GPIO.LOW)
```

Kuva 6. RPi.GPIO-kirjaston käyttöesimerkki

Riveillä 1 ja 2 ohjelmaan tuodaan tarvittavat kirjastot.

Rivillä 4 GPIO-kirjasto asetetaan käsittelemään pinnejä perustuen niiden sijaintiin piirilevyllä. Toinen vaihtoehto olisi antaa komento GPIO.BCM, jolloin käytettäisiin pinnin SOC-kanavanumeroa. Tyypillisesti näkee käytettävän fyysistä paikkanumeroa, sillä se on helpompi hahmottaa, ja sen voi myös laskea itse piirilevystä ilman että tarvitsee muistaa, missä mikäkin pinni sijaitsee. Kumpikin tapa on tietysti yhtä pätevä, kunhan sitä käyttää oikein.

Rivillä 5 pinni numero 7 asetetaan OUT-tilaan, joka asettaa sen valmiustilaan signaalin lähetystä varten. Seuraavalla rivillä pinni asetetaan HIGH-tilaan, jolloin se alkaa lähettää signaalia. Rivillä 7 ohjelma asetetaan odottamaan yhden sekunnin, jonka jälkeen aikaisemmin HIGH-tilaan asetettu pinni asetetaan LOW-tilaan, joka katkaisee signaalin.

Lisää syvyyttä robotin ohjaukseen saa asentamalla Raspberryyn moottoriohjainkortin. Esimerkkinä sellaisesta on luvun 3.3.2. Adafruitin moottoriohjainkortti. Adafruit tarjoaa kortille esimerkkikirjaston, joka kommunikoi kortin kanssa Raspberryn I2C-väylän välityksellä. Kirjastoa hyödyntämällä kortille voi antaa käskyjä, jotka se välittää korttiin liitetyille laitteille. Moottoriohjainkortin käyttö vaatii taustalleen monimutkaisemman koodin, ja siihen voi tarkemmin tutustua Adafruitin ohjeessa, osoitteessa https://learn.adafruit.com/adafruit-dc-and-stepper-motor-hat-for-raspberry-pi/overview. Koodin yhdistäminen ja käyttö omassa ohjelmassa on kuitenkin verrattaen yksinkertaista: RPi-GPIO-kirjaston tapaan moottoriohjainkortin koodi tulee tuoda omaan ohjelmaan, jonka jälkeen korttiin kytketyille moottoreille on yksinkertaista antaa erilaisia käskyjä mm. pyörimissuunnasta ja -nopeudesta.

5 Robotin toteutus

5.1 Projektisuunnitelma

Lähdin lähestymään robottia enemmän populaarikulttuurin määritelmästä; halusin rakentaa itsestään liikkuvan robotin, jota pystyy toisaalta ohjaamaan, ja joka pystyy liikkumaan myös täysin autonomisesti. Mitään ihmisen tekemää työtä se ei tulisi korvaamaan, vaan minua kiehtoi enemmän saada aikaan tietokoneohjelman välityksellä ohjattava robotti.

Alkuun asetin robotille muutaman vaatimuksen:

1. Robottia täytyy pystyä ohjaamaan etänä
2. Robotin täytyy pystyä liikkumaan autonomisesti
3. Käyttäjän täytyy pystyä näkemään, mitä robottikin ”näkee”.

Ensimmäinen työvaihe oli tiedonhankinta. Raspberry Pi -aiheisia videoita on internetissä valtava tarjonta, ja koin YouTuben parhaimmaksi opetusmediaksi. Katsoin paljon eri kanavien videoita robottien rakentamisesta ja ohjelmoimisesta, ja niiden perusteelta sain neuvoa omien tavoitteitteni saavuttamiseen.

Toisena työvaiheena oli robotin fyysinen puoli, eli osien hankinta ja rakentaminen. Hankin osia eri verkkokaupoista ja suunnittelin niistä toimivan kokonaisuuden.

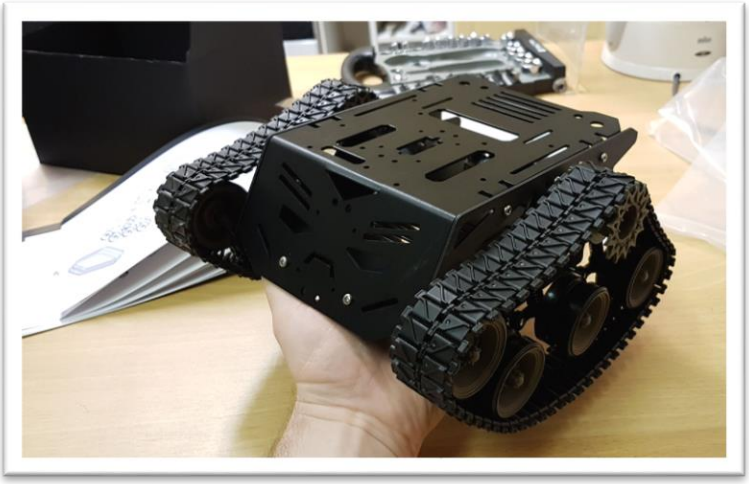
Viimeisenä vaiheena oli ohjelmoida robotti ottamaan vastaan käskyjä, ja reagoimaan ympäristöönsä haluamallani tavalla.

5.2 Robotin rakentaminen

Robotin ”aivoina” toimii Raspberry Pi 3. Kaikki ohjattava ja havaitseva elektroniikka siis kytkeytyy tähän tietokoneeseen, joka suorittaa robottia ohjaavaa ohjelmaa. Raspberryn valitsin siksi, että se tuntui helpoimmin lähestyttävältä ohjelmointialustalta. Kokemukseni perusteella sillä on tutustumistani yhden piirilevyn tietokoneista aktiivisin käyttäjäkunta, ja sen myötä oppimateriaalia on paljon tarjolla.

Robotin koriksi olin aluksi suunnitellut radio-ohjattavan auton koria, mutta tekemieni vertailujen perusteella tulin siihen tulokseen, että autojen korit ovat yleensä niin pieniä, että ne eivät ole optimaalisia robottirakentamiseen. Päädyin lopulta robottirakentelua varta vasten tarkoitettuun koriin, DFRobotin valmistamaan Devastator Tank -alustaan. Kyseessä on metallista valmistettu kehikko, joka kulkee telaketjuilla. Alustan mukana toimitetaan myös

kaksi tasavirtamoottoria, joilla telaketjuja ajetaan. Korissa on paljon valmiita reikiä ja asennuspaikkoja ruuveille, painikkeille, ja erilaisille lisälaitteille mitä robottiin tulisin liittämään, joten se oli modulaarisuutensa puolesta oivallinen alusta robotilleni.



Kuva 7. Robotin kori koottuna

Robotin autonomisesta liikkumisesta vastaamaan valitsin HC-SR04-ultraäänisensorin. Sensori liitetään Raspberry Pi:n GPIO-porttiin, ja sen avulla robotti pystyy laskemaan etäisyyden edessään oleviin esteisiin. Halusin saada sensorin kääntymään robotin asennosta riippumatta, ja tähän tarkoitukseen päädyin Adafruitin valmistamaan Mini Pan-Tilt Kit -moduuliin. Moduulissa on kaksi servomoottoria, jotka mahdollistavat sen kääntymisen vaaka- ja pystyasennossa.

Kuten aikaisemmissa kappaleissa on todettu, moottoreita ei ole hyvä kytkeä suoraan kiinni Raspberry Pi:hin. Moottoreita ohjaamaan tarvitsin siis kaksi eri ohjainkorttia, HAT-tua; yksi telaketjujen moottoreille, ja yksi pan-tilt-moduulin servomoottoreille.

Telaketjujen tasavirtamoottoreille valitsin Adafruit DV & Stepper Motor HATin. Kortilla voidaan ohjata neljää tasavirtamoottoria, tai kahta askelmoottoria. Se liitetään Raspberry Pi:n GPIO-porttiin mukana tulevalla liittimellä, joka tulee itse juottaa kiinni korttiin. Sähkömoottorien virransyöttö ei tule GPIO-portin kautta, vaan se tulee hoitaa ulkoisella virtalähteellä. Tätä varten kortissa on liitäntä.

Servomoottoreiden ohjaukseen valitsin Adafruit 16-Channel PWM / Servo HATin. Sillä voi nimensä mukaisesti ohjata yhteensä 16 servomoottoria kerrallaan. Toimintaperiaate on sama kuin edelliselläkin moottoriohjainkortilla, eli se liitetään mukana tulevalla liittimellä Raspberry Pi:n GPIO-porttiin. Servomoottorien virransyöttö tulee niin ikään hoitaa ulkoisella virtalähteellä.

Moottorien virtalähteenä toimi 5kpl AA-paristoja, joita varten korin mukana toimitettiin sopivasti paristokotelo. Raspberry Pi:n virtalähteeksi valitsin USB-virtapankin. USB-virtapankissa on kuitenkin huomioitava, että ulostuloportin täytyy pystyä 2.1 A virransyöttöön, sillä tyypillinen 1.0 A ei riitä Raspberrylle.

Robotin näköominaisuutta varten valitsin virallisen Raspberry Pi Camera Module V2:n. Kamera kytketään Raspberry Pi:n kameraporttiin mukana tulevalla lattakaapelilla.

Suurimman osan robotin osista sain tilattua Iso-Britanniassa toimivasta The Pi Hut -verkkokaupasta. Raspberryn, USB-virtapankin ja paristot ostin Suomesta Gigantista. Kaiken kaikkiaan osiin kului noin 300 euroa.

Osien haalimisen jälkeen oli rakennusvaihe. Adafruit tarjoaa valmistamiinsa osiin nettisivuillaan selkeät asennusohjeet, joita seuraamalla osat oli helppo saada asennettua. Hatut toimitettiin osissa, joten niiden käyttöönotto vaati hieman tarkkaa kolvausta. Sekä GPIO-liitin, että moottorien liittimet tuli itse juottaa kiinni piirilevyyn. Liittimien pinnit ovat kohtalaisen pieniä, joten tämä vaati erityistä tarkkuutta. Juottamisen jälkeen hatut asennettiin Raspberryyn liittämällä ne GPIO-porttiin. Hatut ladottiin päällekkäin, ja periaatteessa niitä voisi latoa päällekkäin lukemattoman määrän, sillä liittimet mahdollistavat aina uuden hatun kytkemisen edellisen päälle. Moottorien kytkeminen hattuihin oli yksinkertaista; piirilevyissä on selkeät merkinnät kullekin moottorille, ja mihin ne tulee kytkeä. Tasavirtamoottorien johdot ruuvattiin kiinni liittimiin, ja servomoottorit kytkettiin niiden omalla liittimellä piirilevyn vastakappaleeseen.

Pan-tilt -moduulin servoineen asensin robotin korin päälle. Korin päällä oli valmiita reikiä ruuveille, ja sain hyödynnettyä kaksi neljästä asennusreiästä. Moduulin sopivuus koriin ei siis ollut täydellinen, mutta kuitenkin tarpeeksi vakaa käyttötarkoitukseeni.

Kameran kiinnitin aluksi pan-tilt -moduuliin, jolloin käyttäjä pystyi kääntelemään kameraa ja katsomaan siten haluamaansa suuntaan. Tämä kuitenkin osoittautui huonoksi ideaksi, sillä muutaman käännöksen jälkeen kameran kaapeli oli jäänyt kiinni johonkin ja repeytynyt rikki. Hankin uuden kaapelin, ja päätin olla enää kokeilematta liikuteltavaa kameraa, sillä kaapeli oli selvästi liian hauras kyseiseen toimintaan. Päädyin asentamaan kameran kaksipuoleisella teipillä robotin korin keulaan, jolloin se kuvaa aina ajosuuntaan.

Ultraäänisensorin liitin jumper-kaapeleilla hattujen jatkeeksi GPIO-porttiin. Sensorissa on neljä pinniä: kaksi pinniä virralle, ja kaksi signaalien välitykseen. Sensori toimii 5V jännitteellä, joten virtapinnin kytkin Raspberryn 5V syöttöpinniin. Sensoriin maadoituspinnin kytkin niin ikään Raspberryn maapinniin. Signaalipinnit on merkitty tunnistein trigger ja echo. Näiden kautta sensori kommunikoi siihen kytketyn laitteen kanssa, ja ne kytkin vapaavalintaisiin GPIO-pinneihin Raspberryssä.

Koska pan-tilt -moduuli oli nyt vapaana, kiinnitin ultraäänisensorin siihen. Näin sensori voi tarvittaessa kääntyä riippumatta robotin suunnasta. Testeissä tämä osoittautui luotettavammaksi tavaksi havaita etäisyyttä esteisiin. Halusin vielä lisätä visuaalisen efektin havainnoimaan milloin sensori mittaa etäisyyttä. Tätä varten liitin sensorin kylkeen laser-diodin. Laser-diodista varoiteltiin, että sen virrankulutus on suuri ja sitä ei kannata kytkeä suoraan Raspberryyn, joten päätin sen sijaan tarjota sen tarvitseman virran moottorienkin käyttämistä AA-paristoista. Virransyötön johdin releen läpi, jonka taas kytkin Raspberryn GPIO-porttiin. Tällä tavoin saatoin ohjelmallisesti kytkeä diodin päälle ja pois siten, että diodin tarvitsema virta kuitenkin tulisi ulkoisesta virtalähteestä.

Hattujen asentamisen jälkeen Raspberry-kokonaisuus oli kasvanut sen verran suureksi, että se ei enää mahtunut robotin korin sisälle. Päätin siis kiinnittää sen nippusiteillä korin päälle, ja korin sisätilat jäivät näin virtalähteille.

5.3 Robotin ohjelmointi

Robotin ohjaamiseen tarvitsin ohjelman, joka tekee haluamani asiat. Pythonin henkeen kuuluen rakensin ohjelman yhdistelemällä eri kirjastoja ja niiden ominaisuuksia. Liikaa koodia ei kannata kirjoittaa itse, sillä moneen asiaan on jo valmiit kirjastot. Näin aikaa säästyy usein yksitoikkoisilta ja aikaa vieviltä osilta, ja voi keskittää aikansa oleelliseen, eli uuden luomiseen.

Oleellisin kirjasto ohjelmani kannalta on RPi.GPIO-kirjasto. Se on Ben Crostonin ylläpitämä Python-kirjasto, jolla mahdollistaa kommunikoinnin GPIO-portin kautta.

Valmistamiensa moottorikorttien ohjaamiseen Adafruit tarjoaa Python-kirjastot. Servo- ja tasavirtamoottorikorteille on kummallekin omat kirjastonsa, jotka mahdollistavat käskyjen antamisen korteille ohjelmallisesti.

Näppäimistökomentojen antamiseen robotille päätin hyödyntää PyGame-kirjastoa. Kirjasto on varsinaisesti tarkoitettu tietokonepelien ohjelmointia varten, mutta sen tuki näppäimistökomentojen käsittelyyn tekee siitä oivan valinnan myös robottini ohjaamiselle. Tarkemmin käytin pygame.keys-moduulia. Moduuli tarkkailee niin kutsuttuja tapahtumia (events), ja tunnistaa näppäimistöstä, milloin näppäin painetaan alas, ja milloin se vapautetaan. Moduulin avulla ohjelmoin robotilleni ohjausskeeman: telaketjuja ajetaan tietokonepeleistä tutulla WASD-ohjaustyylillä (W eteenpäin, S taaksepäin, A ja D kääntää), ja servomoottoreiden asentoa nuolinäppäimillä. Robottisovellus pitää näppäinten tilasta kir-

jaa kirjastossa (dictionary), ja kun tiettyjen näppäinten tila vaihtuu, toimii robotti sen mukaisesti. Robotin autonomisesta tilasta jätin kokonaan näppäinkomennot ohjelmoimatta. Looginen jatkokehitystavoite olisi ohjelmoida mahdollisuus vaihtaa robotin autonomisen ja käyttäjäohjatun tilan välillä jollain tietyllä näppäinkomennolla. PyGame-kirjaston huono puoli on, että koska se on suunniteltu käytettävän tietokonepelien tekemiseen, on se siten suunniteltu käytettävän sovellusikkunan kautta. Ihannetilanteessa robottiin voisi ottaa mahdollisimman kevyen, tekstipohjaisen SSH-yhteyden, mutta koska tekstipohjaisessa käyttöliittymässä ei voi avata sovellusikkunoita, on Raspberryyn otettava graafinen yhteys joko suoraan, tai käyttäen VNC-sovellusta. PyGame-kirjastoa voi käyttää myös ns. headless-tilassa ilman sovellusikkunaa asettamalla videoajuriksi dummy-ajurin, mutta tällöin eventit eivät toimi, ja sitä kautta myöskään key-moduuli ei toimi, joten sen käyttäminen ei ollut mahdollista robottini kanssa (PyGame 2018).

Ultraäänisensorille en käyttänyt valmista kirjastoa. Sen sijaan käytin ModMyPi.com-sivustolta löytämääni ohjetta apuna tutkan käyttämiseen (ModMyPi.com 2014). Tutkaa ohjelmoidessani huomasin laitteessa joitakin heikkouksia, jotka vaikeuttavat sen käyttöä luotettavaan navigointiin. Tutkan on vaikea havaita hyvin ohuita objekteja, kuten esimerkiksi pöydän- ja tuolinjalkoja. Myös tutkaan nähden viistossa olevia esteitä se ei havaitse kunnolla, mikä uskoakseni johtuu siitä, että näistä viistossa olevista esteistä ultraääni ei kimpoa suoraan takaisin tutkaan, vaan jatkaa matkaansa kulman mukaisesti. Kohtisuorassa tai loivassa kulmassa olevat esteet tutka taas havaitsi hyvin luotettavasti. Myös pehmeät pinnat aiheuttivat ongelmia, esimerkiksi suoraan edessä olevaa kangassohvaa tutka ei aina tahtonut havaita, minkä uskon johtuvan siitä, että pehmeä pinta absorboi ultraääntä itseensä, eikä näin aiheuta tarpeeksi vahvaa kimmoketta, jonka tutka havaitsisi. Ultraäänitutkaa luotettavampi laite navigointiin olisi niin kutsuttu LIDAR-tutka, joka käyttää laservaloa etäisyyden mittaamiseen. LIDAR-moduulit ovat vain huomattavasti kalliimpia kuin ultraäänitutka; käyttämäni HC-SR04 -ultraäänisensori maksoi 2,30€, ja laseretäisyysmittarit ovat yleensä noin sadasta eurosta jopa tuhansiin euroihin asti (Robotshop 2018). Vaihtoehtoisesti LIDAR-tutkia voi ottaa talteen esimerkiksi robottipölynimureista, jolloin voi selvitä halvemmalla.

Kameran käyttöön etsin eri ratkaisuja, ja päädyin lopulta käyttämään UV4L-työkalua, jonka lyhyesti esittelin luvussa 3.3.3. Robotin ohjaus tapahtui joko tekstipohjaisen SSH-yhteyden tai graafisen VNC-sovelluksen välityksellä, joista kumpikaan ei sovellu hyvin nopeasti päivittyvän livevideon katseluun. Tästä syystä päädyin UV4L:ään, jolla voin jakaa robotin näkemän kuvan verkkopalvelimen välityksellä. UV4L:lle annetaan komentorivillä komento tietyillä argumenteilla, ja sovellus käynnistää saamiensa ohjeiden mukaisen verkkopalvelimen. Esimerkiksi komento ”*sudo uv4l -nopreview --auto-video_nr --driver*

raspicam --encoding mjpeg --rotation 180 --width 320 --height 240 --framerate 20 --vstab yes --server-option '--port=9090' ” käynnistää palvelun portissa 9090, kääntää videokuvan 180 astetta, asettaa kuvakooksi 320x240 pikseliä ja kuvataajuudeksi 20 kuvaa sekunnissa. Kameran kuvaamaa kuvaa voi näin seurata livenä samasta verkosta, kirjoittamalla verkkoselaimen osoiteriville Raspberryn IP-osoitteen ja portin 9090. Tarkemmat ohjeet sovelluksen käytöstä löytyy projektin nettisivuilta. (Linux Projects 2018)

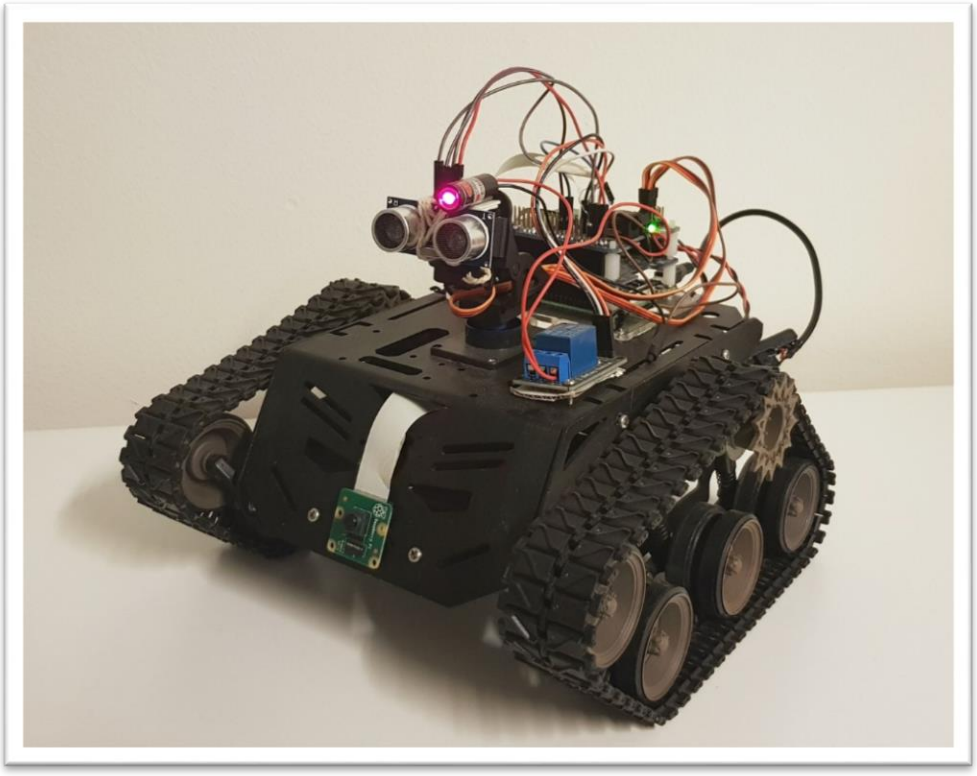
Uuden ohjelmointia tuskin kovin moni tekee ilman esimerkkejä, ja minullakin suurena apuna toimi stackoverflow.com-sivusto, sekä Google-haut yleensäkin; jos kohtasin ongelman tai mielessäni oli toiminnallisuus, kirjoitin kuvauksen Google-hakuun, ja löysin paljon neuvoja ja esimerkkejä muilta ihmisiltä, kuinka he ovat ongelmat ratkaisseet. Varsinaisessa ohjelmassa käytin hyväkseni muiden esimerkeistä saamiani oppeja, valmiita kirjastoja, ja kirjoitin itse näiden väliseen kommunikointiin tarvittavan koodin.

5.4 Tuotos

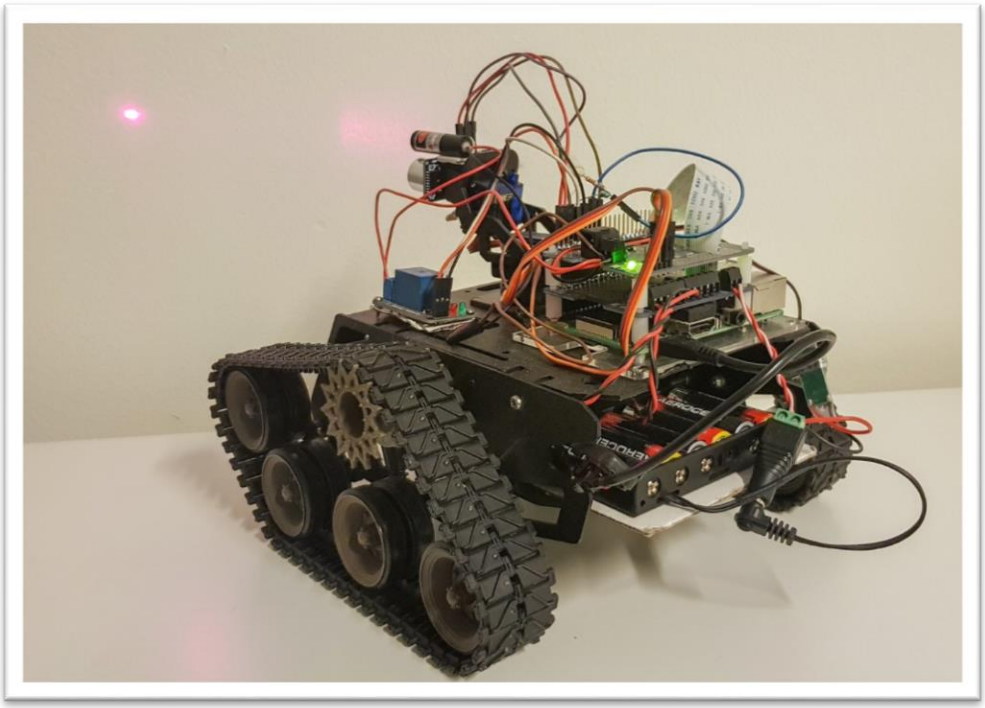
Robotin rakennus- ja ohjelmointivaiheet kulkivat tosiasiassa samanaikaisesti, sillä rakentaminen sisälsi paljon kokeilua, erehtymistä ja soveltamista. Jos yksi asia ei toiminut, kokeilin toisenlaista lähestymistapaa, kokoonpanoa, osaa, tai ohjelmallista toteutusta.

Projektin tuotoksena syntyi yhdistelmä fyysisen ja loogisen maailman elementtejä, jotka yhdessä muodostavat autonomisesti toimivan robotin. Robotin, jota käyttäjä voi halutessaan ohjata, mutta joka kykenee myös itsenäiseen toimintaan. Kuvat 8 ja 9 esittelevät valmiin robotin. Kuvista voi havaita, että en panostanut esteettiseen kaapelointiin, ja johtoviidakko saattaa näyttää hieman sekavalta.

Ohjelmakoodia syntyi 463 riviä; näppäimistöohjatun robotin koodi on 260 riviä (liite 2) ja kaikuluotaava robotti 203 riviä (liite 1). Ohjelmakoodi on myös kokonaisuudessaan katseltavissa GitHub-projektissa osoitteessa https://github.com/matiasraisanen/tankbot. Osoitteesta löytyy myös robotin toimintaa havainnollistavia videoita, sekä luettelo robotin osista.



Kuva 8. Valmis robotti edestä



Kuva 9. Valmis robotti takaa

6 Pohdinta

Tämän opinnäytetyön tavoitteena on ollut tutustua yhden piirilevyn tietokoneisiin, sekä erityisesti Raspberry Pi:n käyttämiseen robotin alustana.

Ensimmäinen tutkimuskysymys koski GPIO-portin käyttämistä. Portin käyttäminen tuli projektin yhteydessä selväksi; portissa on 40 pinniä, joilla on eri funktioita. Portteja voi hyödyntää ohjelmallisesti tiettyjen kirjastojen avulla, riippuen käytettävästä kielestä. Portteihin voi liittää lisälaitteita, joilla laite voi havainnoida ympäristöään, tai antaa käskyjä siihen kytketyille laitteille.

Seuraava kysymys koski sähkömoottorien käyttöä laitteella. Tutkimuksessa selvisi, että laitteeseen voi kytkeä moottorin suoraan ja ohjata sitä ohjelmallisesti, joskaan se ei ole suotavaa, sillä moottorit saattavat aiheuttaa virtapiikin, joka vahingoittaa laitetta. Suoraan kytkemisen sijaan on suositeltavaa käyttää erillistä moottoriohjainkorttia, jonka välityksellä käskyt moottoreille välitetään.

Kolmantena oli kysymys muiden sensorien ja lisälaitteiden käyttämisestä. Käytin projektissani erilaisia, eri tavalla liitettäviä lisälaitteita; moottoriohjainkortit toimivat I2C-väylän välityksellä, kameran liitin lattakaapelilla kameraporttiin, ja ultraäänisensori sekä laser-osoitinta ohjaava rele toimivat vapaavalintaisten GPIO-pinnien kautta. Näin tulin kokeilleeksi erilaisia tapoja yhdistää lisälaitteita.

Viimeiseksi halusin selvittää, kuinka Python-kielellä ohjelmoidaan robotti käyttäen alustana Raspberry Pi -tietokonetta. Selvisi, että lähestymistapoja on lukuisia, ja tyypillisintä on käyttää hyväksi jo valmiita kirjastoja, ja yhdistellä niiden toiminnallisuuksia omaksi, uudeksi kokonaisuudekseen. Python välittää GPIO-portin kautta käskyt ohjainkortille, joka taas kommunikoi ne eteenpäin päätelaitteen ymmärtämäksi signaaliksi, ja toisin päin. Python on parhaimmillaan yksinkertaista, ja kieli on helppo oppia ja omaksua. Kielellä ja kirjastoilla voi lukea laitteen eri signaaleja, ja haluamallaan tavalla tulkita signaalit toiminnallisuuksiksi. Aloittaessani projektin Python oli minulle täysin uusi kieli, mutta projektin edetessä opin kielestä paljon.

Projektin aikana tulin siis vastanneeksi kaikkiin asettamiini projektikysymyksiin. Opin uusista aihealueista valtavasti. Erityisen tutuksi minulle tuli ohjelmistokehityksen ongelmaratkaisu; internetin keskustelufoorumien ja erityisesti StackOverflow.com:in merkitys ongelmanratkaisussa oli merkittävä.

6.1 Jatkokehitys

Projektin tuloksena syntyi jo toimiva kokonaisuus, mutta se jätti myös paljon varaa jatkokehitykselle. Looginen kehitysidea olisi yhdistää käyttäjäohjattu tila ja autonominen tila yhdeksi ohjelmaksi, jossa tilojen välillä voisi vaihtaa napin painalluksella. Robotin ohjelmoinnissa mielenkiintoista on, että rajana on oikeastaan vain oma mielikuvitus, ja modulaarisuutensa ansiosta voin lisätä robottialustaani monenlaisia lisäosia.

Tämän projektin valmistumisen jälkeen robotin kehitys varmasti vielä jatkuu. Erityisesti minua kiinnostaisi lähteä kehittämään robottiin jonkinlaista tartuntakouraa, jolla voisi tarttua erilaisiin esineisiin ja siirrellä niitä. Automaattinen robottikoura on käsitykseni mukaan todella vaikea ohjelmoida, varsinkin jos haluaa pitää kustannukset kohtalaisen matalalla, mutta ihmisohjatun tartuntakouran ohjelmoinnista olisi hyvä aloittaa. Useat nettikaupat myyvät tartuntakourien rakennussarjoja, ja hinnat vaihtelevat muutamasta kymmenestä eurosta satoihin euroihin.

Robotin ohjaamista helpottamaan voisi myös ohjelmoida graafisen käyttöliittymän. Käyttöliittymiä voi ohjelmoida käyttäen apuna esimerkiksi nimenomaan käyttöliittymien rakentamiseen tarkoitettua Tkinter-kirjastoa, mutta se voisi onnistua myös projektissa jo käytössä olevalla PyGame-kirjastollakin. Tkinter-kirjaston käytön opiskeluun mainio apu on John E. Graysonin kirja Python and Tkinter Programming.

Projektin aikana opin käyttämään Pythonia niin hyvin, että olen pystynyt hyödyntämään sitä töissäkin, ja se on taito, jota aion pitää yllä erilaisilla pienillä projekteilla.

Lähteet

Adafruit 2015. Adafruit DC and Stepper Motor HAT for Raspberry Pi. Luettavissa: https://learn.adafruit.com/adafruit-dc-and-stepper-motor-hat-for-raspberry-pi. Luettu: 6.9.2018

Adafruit 2018. Adafruit DC & Stepper Motor HAT. Luettavissa: https://www.adafruit.com/product/2348. Luettu: 30.10.2018

Aqib, M. 2018. Controlling AC Devices with Raspberry Pi | Raspberry Pi Relay Control. Luettavissa: https://electronicshobbyists.com/controlling-ac-devices-with-raspberry-pi-raspberry-pi-relay-control/. Luettu: 31.10.2018

Arduino 2018. Arduino products. Luettavissa: https://www.arduino.cc/en/Main/Products. Luettu: 29.5.2018

AsusTek Computer Inc 2017a. Tinker Board S User's Manual. Luettavissa: https://www.asus.com/ae-en/Single-Board-Computer/Tinker-Board-S/HelpDesk_Manual/. Luettu: 18.9.2018

AsusTeK Computer Inc 2017b. Tinker Board Frequently Asked Questions. Luettavissa: http://dlcdnet.asus.com/pub/ASUS/mb/Linux/Tinker_Board_2GB/FAQ-Tinkerboard_20170425.pdf. Luettu: 18.9.2018

Bommakanti, V. & Roy, R. User Manual Odroid-C2. 2015. Hard Kernel, Ltd. Anyang. Luettavissa: https://magazine.odroid.com/wp-content/uploads/odroid-c2-user-manual.pdf. Luettu: 5.9.2018

Ceder, N. R. 2010. The Quick Python Book. Manning Publications Co. Greenwich.

DFRobot 2018. LattePanda Alpha 864 – Tiny Ultimate Windows / Linux Device. Luettavissa: https://www.dfrobot.com/product-1728.html. Luettu: 8.10.2018

Encyclopedia Britannica, Inc. 2006. Britannica Concise Encyclopedia. Encyclopedia Britannica, Inc.

Guttag, J. V. 2013. Introduction to Computation and Programming Using Python. The MIT Press. Cambridge, Massachusetts, London.

Joseph, L. 2015. Learning Robotics Using Python. Packt Publishing Ltd. Birmingham.

LattePanda 2018. LattePanda - A Windows 10 Computer For Everything. Luettavissa: https://www.kickstarter.com/projects/139108638/lattepanda-a-45-win10-computer-for-everything/description. Luettu: 8.10.2018

Liliputing 2017. Asus Tinker Board mini PC now available in US for $60. Luettavissa: https://liliputing.com/2017/04/asus-tinker-board-mini-pc-now-available-us-60.html. Luettu: 18.9.2018

LinuxLinks 2018. Review: Asus Tinker Board S – Single-Board Computer. Luettavissa: https://www.linuxlinks.com/review-asus-tinker-board-s-single-board-computer/. Luettu: 18.9.2018

Linux Projects 2018. UV4L. Luettavissa: https://www.linux-projects.org/uv4l/. Luettu: 29.10.2018

MacDonald, G. Using Relays and Relay Boards with the Raspberry Pi. Katsottavissa: https://www.youtube.com/watch?v=b6ZagKRnRdM

ModMyPi 2014. HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi. Luettavissa: https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi. Luettu: 26.10.2018

Molloy, D. 2016. Exploring Raspberry Pi. John Wiley & Sons, Inc. Indianapolis.

Nussey, J. 2013. Arduino For Dummies. John Wiley & Sons, Ltd. West Sussex.

Techopedia 2018. Single Board Computer (SBC). Luettavissa: https://www.techopedia.com/definition/9266/single-board-computer-sbc. Luettu: 29.5.2018

ODROID 2018. ODROID Wiki. Luettavissa: https://wiki.odroid.com/start. Luettu: 5.9.2018.

Partco 2018. Yhden releen relekortti 5V DC. Luettavissa: https://www.partco.fi/fi/saehkoemekaniikka/releet/relekortit/17387-relaymod-1.html. Luettu: 31.10.2018

PyGame 2018. HeadlessNoWindowsNeeded – wiki. Luettavissa: https://www.pygame.org/wiki/HeadlessNoWindowsNeeded. Luettu: 14.11.2018

Raspberry Pi Foundation 2018a. Products. Luettavissa: https://www.raspberrypi.org/products/. Luettu: 9.7.2018

Raspberry Pi Foundation 2018b. Raspberry Pi 3 Model B+. Luettavissa https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. Luettu: 18.9.2018.

Raspberry Pi Foundation 2018c. Add-On Boards And HATs. Luettavissa: https://github.com/raspberrypi/hats. Luettu: 10.7.2018

Raspberry Pi Foundation 2018d. Camera module. Luettavissa: https://www.raspberrypi.org/documentation/usage/camera/. Luettu: 29.10.2018

Robotshop.com 2018. LIDAR, Laser Scanners & Rangefinders. Luettavissa: https://www.robotshop.com/en/lidar.html. Luettu: 14.11.2018

Sparkfun 2018. Ultrasonic Sensor - HC-SR04. Luettavissa: https://www.sparkfun.com/products/13959. Luettu: 26.10.2018

Techopedia 2018. Single Board Computer (SBC). Luettavissa: https://www.techopedia.com/definition/9266/single-board-computer-sbc. Luettu: 29.5.2018

The Pi Hut 2018. Ultrasonic Distance Sensor (HC-SR04). Luettavissa: https://thepihut.com/products/ultrasonic-distance-sensor-hcsr04. Luettu: 26.10.2018

Tinker Board Community 2018. Tinker Board Wiki. Luettavissa: https://tinkerboarding.co.uk/wiki/index.php/Software#Operating_Systems. Luettu: 18.9.2018.

Upton, E. & Halfacree, G. 2016. Raspberry Pi User Guide. Wiley. West Sussex.

Liitteet

Liite 1: Sonarbot.py – ultraäänisensorilla ajavan robotin koodi

```
1.  import RPi.GPIO as GPIO
2.  import time
3.  from Adafruit_PWM_Servo_Driver import PWM
4.  import Robot
5.  GPIO.setmode(GPIO.BOARD)
6.  GPIO.setwarnings(False)
7.
8.  #Sonar based on: https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-
    raspberry-pi
9.
10. '''''
11. Bot functions as follows:
12. Execute a distance scan every 500ms.
13. If distance to object is greater than 25cm, drive forward.
14. If distance is less than 25cm, stop and scan the environment at 45 and 135 degree angles.
15. Turn the bot to the direction where the distance to object was greater. Turn until at least 5
    0cm clearance.
16. '''
17.
18. TRIG = 18
19. ECHO = 16
20.
21. GPIO.setup(TRIG,GPIO.OUT)
22. GPIO.setup(ECHO,GPIO.IN)
23. GPIO.output(TRIG, False)
24.
25. pwm = PWM(0x40)
26. panCenter = 365 #Pan servo, center position
27. tiltCenter = 420
28. panMAX = 610  #Leftmost position
29. panMIN = 150  #Rightmost position
30.
31.
32. pwm.setPWMFreq(60)
33. pwm.setPWM(3, 0, 450) #Start tilt servo at a slight downward angle.
34. pwm.setPWM(2, 0, panCenter)  #Start pan servo at center
35.
36. robot = Robot.Robot(left_trim=0, right_trim=0)
37.
38. def laserOn():
39.     GPIO.setup(7, GPIO.OUT)
40.
41. def laserOff():
42.     GPIO.setup(7, GPIO.IN)
43.
44.
45. def measureDist():
46.     GPIO.output(TRIG, True)
47.     time.sleep(0.00001)
48.     GPIO.output(TRIG, False)
49.
50.     while GPIO.input(ECHO)==0:
51.       pulse_start = time.time()
52.
53.     while GPIO.input(ECHO)==1:
54.       pulse_end = time.time()
55.
56.     pulse_duration = pulse_end - pulse_start
57.     distance = pulse_duration * 17150
58.     distance = round(distance, 2)
```

```
59.
60.     return distance
61.
62. def sonarSweep():
63.     '''''Make a 180 degree sweep, and measure distance at every 45 degrees'''
64.     sonarAngle = 150                        #Start on the right, or 0 degrees
65.     pwm.setPWM(2, 0, sonarAngle)
66.     time.sleep(.2)
67.     laserOn()
68.     distance = measureDist()                #Measure distance to object
69.     #print "First measure: ", distance
70.     time.sleep(.2)
71.
72.     #Next we want to find out a direction to turn to
73.     if distance < 25:                        #If object is closer than 25cm, turn the sonar
74.         sonarAngle = 110                    #Change angle value to 110. Servo angle values ar
    e not directly proportional to their values in degrees, hence the manual change here.
75.         while distance < 25:
76.             #print "Too close: ", distance
77.             sonarAngle = sonarAngle + 125   #Increment the angle by 125, through 150, 235, 36
    5, 485 and 610
78.
79.             if sonarAngle > 610:         #If we exceed maximum angle of 610, turn the whole r
    obot on the spot
80.                 #print("No room! Turn around")
81.                 pwm.setPWM(2, 0, panCenter)
82.                 break
83.
84.             pwm.setPWM(2, 0, sonarAngle)    #Turn the sonar to the new angle
85.             #print "Let's try a new angle:", sonarAngle
86.             time.sleep(.2)
87.             distance = measureDist()        #Scan for objects
88.             #print "New measure: ", distance
89.             time.sleep(.1)
90.     pwm.setPWM(2, 0, panCenter)
91.     time.sleep(.5)
92.
93.     #Next we start turning into the direction, until we have at least 50cm room.
94.
95.     if sonarAngle < 360: #Turn right
96.         distance = measureDist()
97.         time.sleep(.2)
98.         while distance < 50:
99.             robot.RT_backward(150)
100.                robot.LT_forward(150)
101.                distance = measureDist()
102.                time.sleep(.5)
103.            laserOff()
104.            robot.stop()
105.            return "dist below 30"
106.
107.        elif sonarAngle > 360:   #Turn left
108.            distance = measureDist()
109.            time.sleep(.2)
110.            while distance < 50:
111.                robot.RT_forward(150)
112.                robot.LT_backward(150)
113.                distance = measureDist()
114.                time.sleep(.5)
115.            laserOff()
116.            robot.stop()
117.            return "dist above 30"
118.        else:
119.            laserOff()
120.            return "Something Went Wrong"
121.
122.    def sonarSweep2():
```

```
123.         '''''Measure distance at 45 and 135 degrees, compare them, and turn in the directio
    n of the greater value'''
124.
125.         sonarAngle = 235                        #Measure on the right, at 45 degrees
126.         pwm.setPWM(2, 0, sonarAngle)
127.         time.sleep(.2)
128.         laserOn()
129.         distanceRight = measureDist()
130.         print "Distance on right: ", distanceRight
131.         time.sleep(.2)
132.
133.         sonarAngle = 485                        #Measure on the left at 135 degrees
134.         pwm.setPWM(2, 0, sonarAngle)
135.         time.sleep(.2)
136.         distanceLeft = measureDist()
137.         print "Distance on left: ", distanceLeft
138.         time.sleep(.2)
139.
140.         sonarAngle = panCenter                  #Return the sonar to center
141.         pwm.setPWM(2, 0, sonarAngle)
142.         time.sleep(.2)
143.
144.         #Next we want to find out a direction to turn to
145.         if distanceRight > distanceLeft: #Compare values, and choose the direction of turn

146.             print "Right has more room. Turning right."
147.             distance = measureDist()
148.             time.sleep(.2)
149.             while distance < 45: #Turn right until clearance is at least 45cm.
150.                 print "Clearance of 45cm reached on right. Turning."
151.                 robot.RT_backward(150)
152.                 robot.LT_forward(150)
153.                 distance = measureDist()
154.                 time.sleep(.2)
155.             laserOff()
156.             robot.stop()
157.             return "Right turn complete"
158.
159.         else:                       #Turn left until clearance of at least 45cm.
160.             print "Left has more room."
161.             distance = measureDist()
162.             time.sleep(.2)
163.             while distance < 45:
164.                 print "Clearance of 45cm reacehd on left. Turning."
165.                 robot.RT_forward(150)
166.                 robot.LT_backward(150)
167.                 distance = measureDist()
168.                 time.sleep(.2)
169.             laserOff()
170.             robot.stop()
171.             return "Left turn complete"
172.
173.
174.     def moveBot():
175.
176.         running = True
177.         botSpeed = 200
178.
179.         while running:
180.             distance = measureDist()
181.
182.             if distance > 40:
183.                 print "Forward. Distance: ", distance
184.                 robot.RT_forward(botSpeed)
185.                 robot.LT_forward(botSpeed)
186.             else:
187.                 print "Stop. Distance: ", distance
188.                 runCount = 0
```

```
                    robot.stop()
                    result = sonarSweep2()
                    print result

                time.sleep(.5)

            print("Stop")
            robot.stop()

        def main():
            moveBot()
            GPIO.cleanup()

        if __name__ == "__main__":
            main()
```

Liite 2: Tankbot.py – näppäimistökomennoilla ajavan robotin koodi

```
from Adafruit_PWM_Servo_Driver import PWM
import pygame
import os, sys
import Robot
import threading
import RPi.GPIO as GPIO
import time
import sonicBot

GPIO.setmode(GPIO.BOARD)

#Servo setup:
#Connect PAN servo to slot 2 and TILT servo to slot 3

pwm = PWM(0x40)
panServo = 365  #Pan servo, center position
tiltServo = 420 #Tilt servo, center position

panMAX = 610  #Leftmost position
panMIN = 150  #Rightmost position

tiltMAX = 530 #Downmost position
tiltMIN = 220 #Upmost position

servoSpeed = 5 #Turning speed

pwm.setPWMFreq(60)
pwm.setPWM(3, 0, tiltServo) #Start servos centered
pwm.setPWM(2, 0, panServo)  #Start servos centered

#END servo setup


speed = 255 #Speed multiplier, values 0-255
reducedSpeed = int(speed*0.50)  #Alternate track speed for turning. The lower the percentage,
    the lower the turning radius and forward momentum.
robot = Robot.Robot(left_trim=0, right_trim=0)

#os.environ['SDL_VIDEODRIVER'] = 'dummy'
pygame.init()
screen = pygame.display.set_mode((100, 100))

running = True

#Dictionary for command values.
```

```
#The use of a dictionary helps us to read multiple values at the same time,
#and to specify an action related to their value.

moveCommandValues = { 'W':0,
                      'A':0,
                      'S':0,
                      'D':0, }

cameraCommandValues = { 'K_UP':0,
                        'K_DOWN':0,
                        'K_LEFT':0,
                        'K_RIGHT':0,
                         }

functionCommandValues = { 'K_SPACE':0, }


print ("********\n" + "*READY!*\n" + "********")
print("Press ESCAPE to QUIT")

while running:
    for event in pygame.event.get():
      #When a key is pressed down
      if event.type == pygame.KEYDOWN:

        #Change values for Tank Movement
        if event.key == pygame.K_w:
          #print("forward!")
          moveCommandValues['W'] = 1
        if event.key == pygame.K_s:
          #print("reverse")
          moveCommandValues['S'] = 1
        if event.key == pygame.K_a:
          #print("turn left")
          moveCommandValues['A']= 1
        if event.key == pygame.K_d:
          #print("turn right")
          moveCommandValues['D'] = 1
        if event.key == pygame.K_SPACE:
          #print("stop")
          robot.stop()

        #Change values for Camera Movement
        if event.key == pygame.K_UP:
          #print("Camera up")
          cameraCommandValues['K_UP'] = 1
        if event.key == pygame.K_DOWN:
          #print("Camera down")
          cameraCommandValues['K_DOWN'] = 1
        if event.key == pygame.K_LEFT:
          #print("Camera left")
          cameraCommandValues['K_LEFT'] = 1
        if event.key == pygame.K_RIGHT:
          #print("Camera right")
          cameraCommandValues['K_RIGHT'] = 1
            if event.key == pygame.K_SPACE:
              functionCommandValues['K_SPACE'] = 1

            if event.key == pygame.K_RCTRL: #Press right CTRL to center PanTilt-kit
              panServo = 365
              tiltServo = 420
              pwm.setPWM(2, 0, panServo)
              pwm.setPWM(3, 0, tiltServo)

            #Press ESCAPE to QUIT
            if event.key == pygame.K_ESCAPE:
              print(moveCommandValues)
              print(cameraCommandValues)
```

```
                running = False

            #When a key is released
            if event.type == pygame.KEYUP:

              #Change values for Tank Movement
              if event.key == pygame.K_w:
                #print("stopFORWARD")
                moveCommandValues['W'] = 0
              if event.key == pygame.K_s:
                #print("stopREVERSE")
                moveCommandValues['S'] = 0
              if event.key == pygame.K_a:
                #print("stopLEFT")
                moveCommandValues['A'] = 0
              if event.key == pygame.K_d:
                #print("stopRIGHT")
                moveCommandValues['D'] = 0

              #Change values for Camera Movement
              if event.key == pygame.K_UP:
                #print("Camera up STOP")
                cameraCommandValues['K_UP'] = 0
              if event.key == pygame.K_DOWN:
                #print("Camera down STOP")
                cameraCommandValues['K_DOWN'] = 0
              if event.key == pygame.K_LEFT:
                #print("Camera left STOP")
                cameraCommandValues['K_LEFT'] = 0
              if event.key == pygame.K_RIGHT:
                #print("Camera right STOP")
                cameraCommandValues['K_RIGHT'] = 0

              if event.key == pygame.K_SPACE:
                functionCommandValues['K_SPACE'] = 0

          #Translate values into movement commands

          #Only forward
          if  moveCommandValues['W'] == 1 \
          and moveCommandValues['A'] == 0 \
          and moveCommandValues['S'] == 0 \
          and moveCommandValues['D'] == 0:
            robot.RT_forward(speed)
            robot.LT_forward(speed)

          #Left turn while moving forward
          if  moveCommandValues['W'] == 1 \
          and moveCommandValues['A'] == 1 \
          and moveCommandValues['S'] == 0 \
          and moveCommandValues['D'] == 0:
            robot.RT_forward(speed)
            robot.LT_forward(reducedSpeed)

          #Right turn while moving forward
          if  moveCommandValues['W'] == 1 \
          and moveCommandValues['A'] == 0 \
          and moveCommandValues['S'] == 0 \
          and moveCommandValues['D'] == 1:
            robot.RT_forward(reducedSpeed)
            robot.LT_forward(speed)

          #Only reverse
          if  moveCommandValues['W'] == 0 \
          and moveCommandValues['A'] == 0 \
          and moveCommandValues['S'] == 1 \
          and moveCommandValues['D'] == 0:
            robot.RT_backward(speed)
```

```
            robot.LT_backward(speed)

          #Left turn while reversing
          if  moveCommandValues['W'] == 0 \
          and moveCommandValues['A'] == 1 \
          and moveCommandValues['S'] == 1 \
          and moveCommandValues['D'] == 0:
            robot.RT_backward(reducedSpeed)
            robot.LT_backward(speed)

          #Right turn while reversing
          if  moveCommandValues['W'] == 0 \
          and moveCommandValues['A'] == 0 \
          and moveCommandValues['S'] == 1 \
          and moveCommandValues['D'] == 1:
            robot.RT_backward(speed)
            robot.LT_backward(reducedSpeed)

          #Only left turn
          if  moveCommandValues['W'] == 0 \
          and moveCommandValues['A'] == 1 \
          and moveCommandValues['S'] == 0 \
          and moveCommandValues['D'] == 0:
            robot.RT_forward(speed)
            robot.LT_backward(speed)

          #Only right turn
          if  moveCommandValues['W'] == 0 \
          and moveCommandValues['A'] == 0 \
          and moveCommandValues['S'] == 0 \
          and moveCommandValues['D'] == 1:
            robot.RT_backward(speed)
            robot.LT_forward(speed)

          #Stop when no keys pressed
          if moveCommandValues['W'] == 0 \
          and moveCommandValues['A'] == 0 \
          and moveCommandValues['S'] == 0 \
          and moveCommandValues['D'] == 0:
            robot.stop()


          #Translate values into camera movement
          if cameraCommandValues['K_UP'] == 1:  #If the key is pressed down
            #print("UP")
            tiltServo -
    = servoSpeed     #Change the servo position by the value of servoSpeed.
            if tiltServo < tiltMIN:     #If positional value exceeds the set limit,
              tiltServo = tiltMIN       #set it back to its limit value.
            pwm.setPWM(3, 0, tiltServo) #Move the servo to the specified location

          if cameraCommandValues['K_DOWN'] == 1:
            #print("DOWN")
            tiltServo += servoSpeed
            if tiltServo > tiltMAX:
              tiltServo = tiltMAX
            pwm.setPWM(3, 0, tiltServo)

          if cameraCommandValues['K_LEFT'] == 1:
            #print("LEFT")
            panServo += servoSpeed
            if panServo > panMAX:
              panServo = panMAX
            pwm.setPWM(2, 0, panServo)


          if cameraCommandValues['K_RIGHT'] == 1:
            #print("RIGHT")
```

```
            panServo -= servoSpeed
            if panServo < panMIN:
              panServo = panMIN
            pwm.setPWM(2, 0, panServo)

          if functionCommandValues['K_SPACE'] == 1: #Laser on
            GPIO.setup(7, GPIO.OUT)
          if functionCommandValues['K_SPACE'] == 0: #Laser off
            GPIO.setup(7, GPIO.IN)


    pygame.quit()
    quit()
```