Robotiikka

Tapio Hansson

Sisältö

Johdanto

Tämä materiaali on suunniteltu Hyvinkään Sveitsin lukion koulukohtaisen syventävän kurssin materiaaliksi. Kurssilla on tarkoitus tutustua robotiikkaan käytännönläheisesti ja tavoitteena on rakentaa robotti ja oppia ohjelmoimaan sitä. Kurssilla käytetään Raspberry Pi-minitietokoneen ympärille rakennettavaa CamJam edukit 3-robottia, joka on edullisuutensa ja hyvän oppaistonsa ansiosta erinomainen laite robotiikkaan tutustumista varten.

1 Mitä ovat robotit?

Wikipedia määrittelee robotin seuraavasti:

> Robotti (tšek. robota 'pakkotyö') tarkoittaa useimmiten mekaanista laitetta tai konetta, joka osaa jollain tavoin toimia fyysisessä maailmassa.

Robotin perusedellytys on siis toiminta fyysisessä maailmassa. Erilaisia älykkäitä järjestelmiä, jotka toimivat tietokoneella tai internetissä kutsutaan boteiksi. Robotti on käsitteenä paljon vanhempi kuin tietokone, minkä vuoksi aikaiset robotit ovat olleet hyvin mekaanisia laitteita jotka ovat suorittaneet jotain hyvin määriteltyä tehtävää. Modernimmat ohjelmoitavat tietokonepohjaiset robotit ovat yleistyneet varsin pitkälti samaa tahtia tietokoneiden kanssa, mutta viime vuosina tietokoneiden kehityttyä yhä pienemmiksi, vähävirtaisemmiksi ja halvemmiksi voi nyt jo kuka tahansa rakentaa hyvinkin monimutkaisia robottijärjestelmiä hyvin edulliseen hintaan. Ohjelmoitavasta robotista on vielä matkaa älykkääseen robottiin. Älykkäiden robottien kehitys tuleekin olemaan luultavasti hyvin nopeaa tekoälyn kehittyessä kaiken aikaa paremmaksi ja paremmaksi.

Robotin määritelmä on siis varsin laaja, eikä sillä välttämättä tarkoiteta ihmisen kaltaista androidia. Monimutkainen robotti tarkkailee ympäristöään ja suorittaa annettuja tehtäviä sen mukaan mitä tietoa se saa sensoreiltaan. Se voi esimerkiksi sammuttaa valot, kun se ei havaitse liikettä, lukita ovia tai säätää lämmitystä. Toisaalta äärimmilleen viritetty robotti osaa liikuttaa potilasta tämän hengityksen tahdissa, jotta sädehoito saadaan kohdistettua täydellisesti kohteeseen. Yksinkertainen robotti saattaa puolestaan vain liikauttaa jotain osaa napin painalluksesta.

Robotiikka tarkoittaa insinööritieteen haaraa joka keskittyy robotteihin. Koska robotit ovat hyvin monimuotoisa, on robotiikkakin varsin monialainen tieteen haara ja sisältää konetekniikkaa, ohjelmointia ja materiaalitieteitä. Robotiikka on peräisin tieteiskirjailija Isaac Asimovin tieteiskirjoista. Asimov määritteleekin kirjoissaan robotiikan lakeja, joita on tälläkin kurssilla syytä noudattaa poikkeuksitta. Kolme robotiikan peruslakia ovat

1. Robotti ei saa vahingoittaa ihmistä eikä toiminnasta pidättäytymällä saattaa tätä vahingoittumaan.

2. Robotin täytyy totella ihmisten sille antamia määräyksiä, paitsi silloin kun ne ovat ristiriidassa ensimmäisen pääsäännön kanssa.

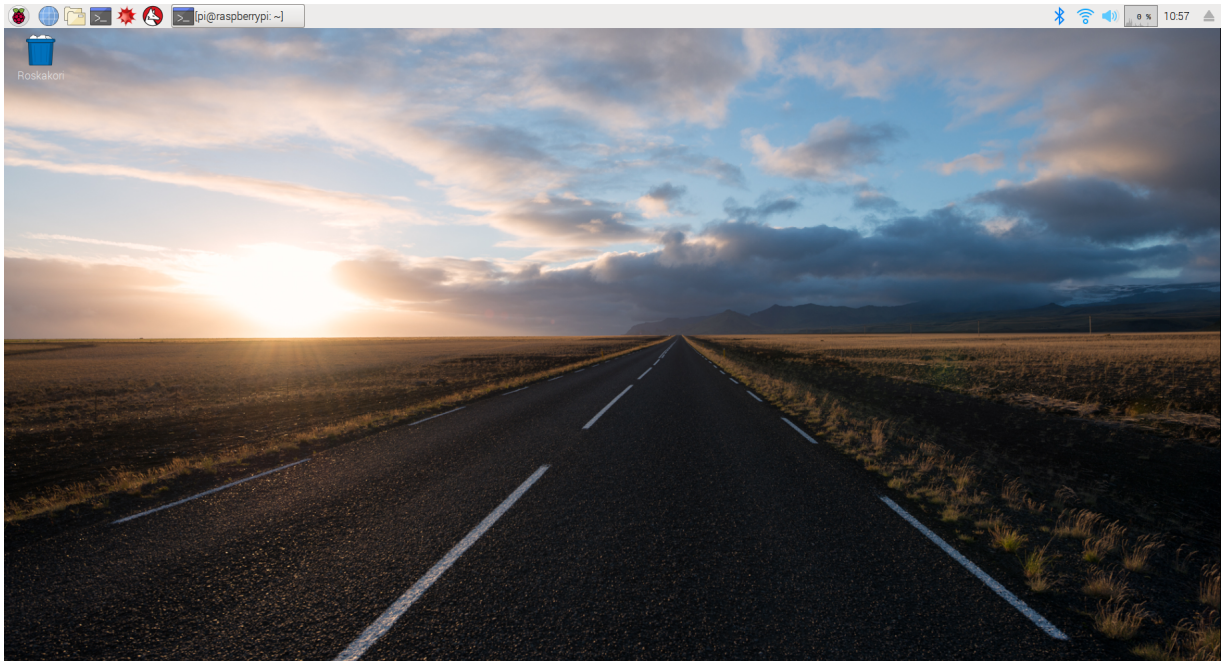
3. Robotin täytyy varjella omaa olemassaoloaan sikäli kuin se ei ole ristiriidassa ensimmäisen tai toisen pääsäännön kanssa.

2 Raspberry Pi ja Raspbian

Tällä kurssilla robotit rakennetaan Raspberry Pi-korttitietokoneen ympärille. Raspberrylle (raspi) on tarjolla useita käyttöjärjestelmiä, joista tässä käytetään ehkä yleisintä niistä, eli Raspbiania. Raspbian pohjautuu Debian-linuxiin ja on ilmainen ja vapaa ohjelmisto.

Ohjelmisto asennetaan toista tietokonetta käyttäen sd-kortille, josta raspi käynnistää itsensä. Ohjeet asennukseen löytyvät esimerkiksi tästä: https://www.raspberrypi.org/documentation/installation/noobs.md

Kun järjestelmä on valmiiksi asennettu, pitäisi siitä päätyä työpöydälle, joka näyttää kuvan 1 mukaiselta.



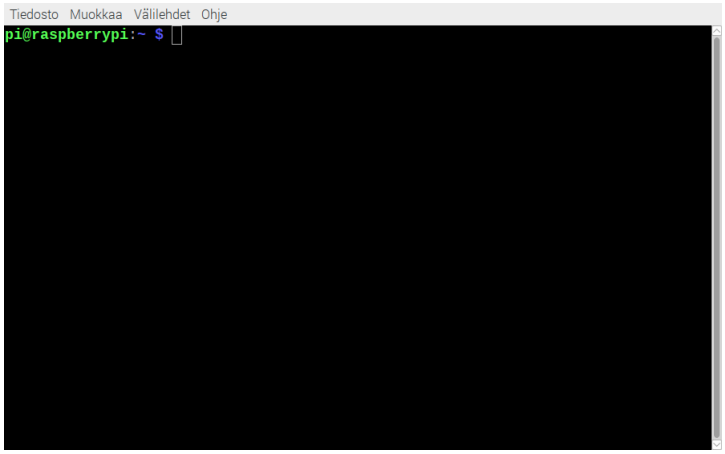
Kuva 1: Raspbian-työpöytä

Yläreunan vasemmasta laidasta löytyvästä vadelmakuvakkeesta pääsee valikkoon, josta löytyvät kaikki käyttöjärjestelmän olennaiset osiot, eli lähinnä asetukset ja ohjelmat. Osa oletuksena käytössä olevista ohjelmistoista on varsin yksinkertaisia, sillä vaikka korttikoneet ovatkin kehittyneet varsin paljon viime vuosina, on niillä edelleen käytössä varsin rajalliset resurssit, joten useimmiten käyttöön kannattaa valita mahdollisimman keveitä ohjelmstoja.

Valitsemalla valikosta "Apuohjelmat - LXTerminal"pitäisi näytölle aueta musta ruutu, jota kutsutaan päätteeksi, tuttavallisemmin komentoriviksi tai terminaaliksi. Ammoisina aikoina, ennen graafisia käyttöliittymiä tietokoneet koostuivat vain yhdestä komentorivistä, eikä niillä siten voinut tehdä useampia asioita yhtä aikaa. Komentorivi on varsin tehokas työväline kun sitä tottuu käyttämään, vaikka se saattaa tuntua graafisiin ohjelmiin tottuneille hieman vieraalta. Tällä kurssilla sitä kuitenkin käytetään varsin paljon.

Koska robotti tarvitsee älyn, täytyy sitä varten kirjoittaa ohjelmia. Ohjelmat kirjoitetaan tekstieditorilla. Koska Raspbianin mukana tuleva tekstieditori on hieman rajoittunut, kannattaa tilalle asentaa toinen. Hyvä vaihtoehto on varsin kevyt mutta monipuolinen Pluma, jonka voi asentaa kirjoittamalla juuri avaamaasi terminaali-ikkunaan

```
sudo apt install pluma
```



Kuva 2: Komentorivi, eli pääte

ja painamalla enteriä kun ohjelma kysyy haluatko varmasti jatkaa. Kun asennus on valmis, löytyy Pluma valikosta apuohjelmien sisältä. Pluman asetuksista kannattaa säätää ohjelmointia helpottavia asetuksia, kuten rivinumerointi ja sulkuparin korostus.

3 Johdatus ohjelmoinnin perusteiden alkeisiin

Kurssilla käytetään Python-ohjelmointikieltä. Python soveltuu yksinkertaisuutensa vuoksi varsin hyvin ensimmäiseksi ohjelmointikieleksi, mutta taipuu tarvittaessa hyvin monipuoliseen käyttöön.

Python ohjelmat kirjoitetaan .py-päätteisiin tekstitiedostoihin, joita voi ajaa komentoriviltä käsin. Ennen kuin paneudutaan pythonin saloihin tarkemmin kirjoitetaan ensimmäinen ohjelmaja testataan ympäristöä.

Vaikka kurssilla käytetään Raspberry Pitä ja siinä olevia ohjelmia, voi koko tämän luvun sisältöä ja tehtäviä tehdä omalla tietokoneella asentamalla Python-ympäristön itselleen. Windows-tietokoneessa tämä onnistuu lataamalla Python-asennusohjelma osoitteesta: https://www.python.org/ftp/python/3.6.5/python-3.6.5.exe ja suorittamalla asennus. Tämän jälkeen voit kirjoittaa koodia .py-tiedostoihin ja ajaa niitä käyttämällä komentokehotetta (command prompt) kuten tämän ohjeen muissa osissa tehdään. Hyvä tekstieditori Windowsille on Notepad++, jonka voi ladata osoitteesta https://notepad-plus-plus.org/download/v7.5.6.html.

3.1 Hello world!

Historiallisista syistä yleensä ensimmäinen ohjelma joka ohjelmointia aloittaessa tehdään on nimeltään Hello world. Sen tarkoituksena on ainoastaan tulostaa kyseinen teksti näytölle. Avaa nyt asentamasi tekstieditori Pluma ja kirjoita sinne rivi

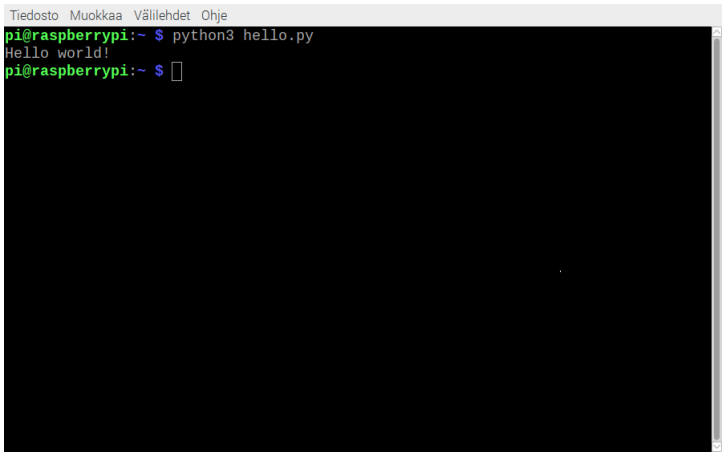
```
print("Hello world!")
```

Tallenna tiedosto kotikansioosi nimellä "hello.py". Linux-pohjaisissa käyttöjärjestelmissä jokaisella käyttäjällä on oma kotikansio, johon hänen henkilökohtaiset tiedostonsa on tarkoitus tallentaa. Raspbianissa on oletuksena yksi käyttäjä nimeltään pi, joten tallenna kaikki tiedostosi kansioon /home/pi. Tähän kansioon tullaan tässä oppaassa viittaamaan kotikansiona.

Ohjelmat ajetaan siis komentoriviltä. Pythonista on tällä hetkellä käytössä vielä kahta eri versiota Python 2 ja Python 3. Tämä ohje käyttää 3. versiota, mutta Raspbianin (raspbian 9-versiossa) oletusversiona on vielä Python 2. Tämän vuoksi aina ohelmia ajettaessa tulee käyttää komentoa python3. Kirjoita siis nyt komentoriville

```
python3 hello.py
```

ja lopputulkosen pitäisi näyttää hyvin pitkälit kuvan 3 mukaiselta. Jos et ole aiemmin ohjelmoinut, niin onneksi olkoon! Teit juuri ensimmäisen ohjelmasi.



Kuva 3: Hello world ohjelma

3.2 Komentorivin käyttöä

Tietokonetta voi periaatteessa käyttää vain komentorivin kautta. Tekstipohjaiset ohjelmat saattavat alkuun tuntua hieman vierailta, mutta osa niistä on erittäin näppäriä työkaluja, ja etenkin ohjelmoinnin kanssa komentoriviä joutuu käyttämään hieman enemmän. Kun avaat päätteen olet oletuksena kotikansiossasi. Komentorivisi on siis aina jonkin kansion sisällä, jossa komennot suoritetaan. Kun ohelmia alkaa kertyä enemmän on ne hyvä järjestää kansioihin, joten komentorivillä on syytä osata liikkua kansiosta toiseen. Seuraavassa lista muutamista peruskomennoista joita komentoriviä käyttäessä usein tarvitsee.

- **ls**-komento listaa kaikki kansiossa olevat asiat. Sen avulla löytää tiedostoja ja muita kansioita.
- **cd**-komennolla (change directory) voit liikkua kansiorakenteessa. Voit kotikansiossa ollessasi siirtyä esimerkiksi Lataukset-kansioon komentamalla

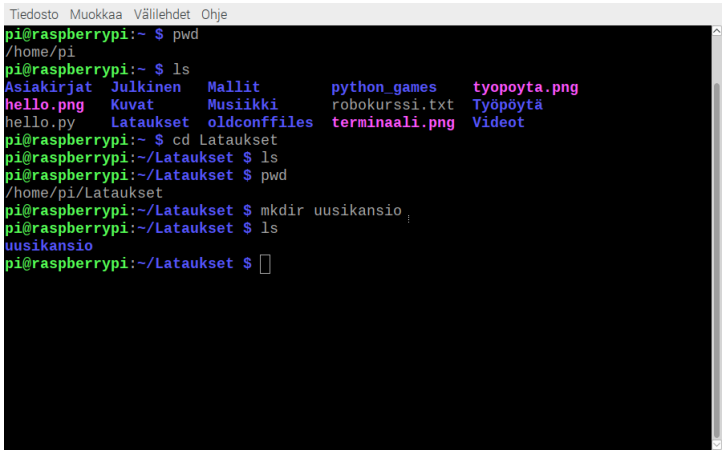
  ```
  cd Lataukset
  ```

  Takaisin päin kansiorakenteessa pääsee komentamalla

  ```
  cd ..
  ```

- **pwd**-komento tulostaa tarkan sijainnin jossa olet. Tämä voi olla hyödyllistä, jos eksyt kansiorakenteessa.
- **mkdir**-komennolla (make directory) voit luoda uuden kansion.

Esimerkkejä näet vielä kuvassa 4



Kuva 4: Tärkeimpiä komentoja

3.3 Pythonin perusteet

Hello world ei ole mitenkään kovin mielenkiintoinen ohjelma sillä se ei tee mitään kovin merkittävää. Perusohjelmointi koostuu oleellisesti vain muutamasta tärkeimmästä rakenteesta ja niitä ovat muuttujat, ehtolauseet ja toistorakenteet. Näihin kun lisätään esimerkiksi tiedostojen käsittelyä tai syötteen lukua käyttäjältä saadaan jo hyvin monipuolisia ohjelmia aikaan. Tämän kurssin tavoitteena on saada ohjelmoitua robottia, joten keskitymme tarkemmin niihin osioihin, jotka auttavat tätä tavoitetta silmällä pitäen.

3.3.1 Muuttujat

Muutujat ovat ohjelmoinnin tärkeimpiä työkaluja. Niihin voidaan tallentaa lukuarvoja tai tekstiä ja niitä voidaan tarpeen mukaan muuttaa myöhemmin. Tietokone on varsin hyvä laskin, joten Pythonista löytyy suoraan yhteen, vähennyt ja kerto ja jakolasku joita voi käyttää lukujen välillä. Muutetaan Hello world-ohjelmaa siten että lasketaan kaksi lukua yhteen. Avaa uusi tiedosto tekstieditoriin ja kirjoita sinne rivi

```
print(1+2)
```

Tallenna tiedosto esimerkiksi nimellä yhteenlasku.py. Ajettuasi ohjelman huomaat että ohjelma tulostaa tekstin 1+2 sijaan luvun 3.

Sama tehtävä voidaan tehdä käyttämällä muuttujia. Muuttuja määrätään yhtäsuuruusmerkin avulla esimerkiksi

```
luku1 = 1
```

Muuttujien avulla voidaan myös laskea suoraan. Muuta seuraavaksi yhteenlasku.py -tiedosto niin että se näyttää tältä:

```
1 luku1 = 1
2 luku2 = 2
3 summa = luku1+luku2
4 print(summa)
```

Lopputulos näyttää samalta, mutta nyt lukuja voi käyttää useammassa paikassa. Seuraavalla esimerkkikoodilla voit laskea myös erotuksen ja tulon.

```
luku1 = 1
luku2 = 2
summa = luku1+luku2
erotus = luku1-luku2
tulo = luku1*luku2
print("lukujen summa: " + str(summa))
print("lukujen erotus: " + str(erotus))
print("lukujen tulo: " + str(tulo))
```

Tulostuskomennossa esiintyvä str()-funktio muuntaa kokonaisluvusta merkkijonon, jotta se voidaan yhdistää tekstin kanssa samaan tulostuskomentoon.

3.3.2 Ehtolauseet

Ehtolauseella ohjelma voi päättää mitä se tekee riippuen jostakin ehdosta, kuten muuttujan arvosta. Logiikka jota yleensä käytetään on "jos ...niin". Pythonin if-lause toimii seuraavan esimerkin mukaisesti:

```
luku = 3
if luku < 5:
    print("Luku on alle 5!")
else:
    print("Luku on 5 tai enemmän!")
```

if-komentoa seuraa ehto, jonka jälkeen tulee kaksoispiste. Tärkeä huomio on, että printkomennot on sisennetty. Pythonissa koodilohkot erotellaan sisentämällä. Print komento on sisennettynä if-ehtoon verrattuna ja tarkoittaa siten että se kuuluu if-lauseeseen ja toteutetaan ainoastaan jos ehto toteutuu. Else-lauseen avulla voidaan määrätä mitä tehdään siinä tapauksessa, että annettu ehto ei täyty. Ehtoja voi asettaa myös useampia ja jaottelua voi jatkaa else if-rakenteella (elif), kuten seuraava esimerkki havainnollistaa:

```
luku = 7
if luku < 5:
    print("Luku on alle 5!")
elif luku < 10 :
    print("Luku on yli 5 mutta alle 10!")
else:
    print("Luku on 10 tai enemmän!")
```

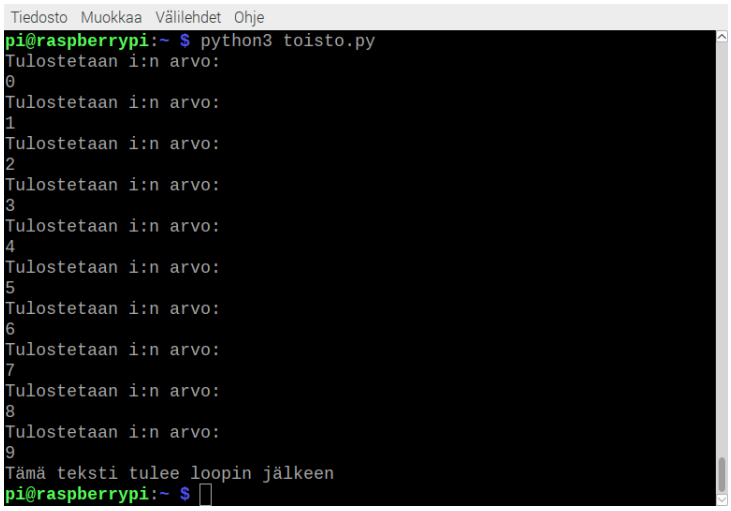
3.3.3 Toistorakenteet

Tietokone soveltuu myös erinomaisen hyvin asioiden toistamiseen useita kertoja. Tällaisia ongelmia varten on toistorakenteet, eli loopit, joista useimmin käytettyjä ovat *for* ja *while*. For-loopit käyvät läpi jonkin tietyn listan tai joukon kun taas while-looppi toistuu niin kauan kuin ennalta määrätty ehto on voimassa. Tässä kohdassa esittelemme tarkemmin while-rakenteen, ja palaamme for-rakenteeseen myöhemmin, sikäli mikäli se osoittautuu tarpeelliseksi. Useimmat ongelmat ovat kuitenkin ratkaistavissa kummalla tavalla tahansa.

While-ehto toimii seuraavan esimerkin osoittamalla tavalla. Kirjoita koodi itsellesi ja kokeile ajaa se. Muista laittaa sisennykset oikein, jotta loopin sisällä tehdään kaikki mikä kuuluukin.

```
1 i = 0
2 while(i<10):
3         print("Tulostetaan i:n arvo:")
4         print(i)
5         i=i+1
6 print("Tämä teksti tulee loopin jälkeen")
```

Edellisen esimerkin tulosteen pitäisi olla kuten kuvassa 5. Ehto, jonka tulee toteutua



Kuva 5: Esimerkin tuloste

kirjoitetaan sulkeisiin while-sanan jälkeen. Tässä tapauksessa muuttujan i arvon tulee olla alle 10. Looppia jatketaan niin kauan kunnes ehto ei ole enää voimassa, eli kun i:n arvoksi muuttuu kymmenennen kierroksen lopussa 10, ei seuraava kierros enää ala. Sisennyksillä määrätään jälleen mitkä asiat kuuluvat loopin sisälle.

3.3.4 Esimerkki: Fibonaccin lukujono

Eräs hyvä esimerkki muuttujien ja toistorakenteen käytöstä on rekursiivisten lukujonojen laskeminen. Seuraava ohjelma laskee tunnetun Fibonaccin lukujonon jäseniä. Fibonaccin lukujonossa seuraava jäsen saadaan aina kahden ensimmäisen summana, ja kaksi ensimmäistä jäsentä ovat ykkösiä.

```
1  i=1
2  luku1 = 1
3  luku2 = 1
4
5  print("Fibonaccin lukujono")
6  print(luku1)
7  print(luku2)
8  while(i<100):
9          luku3 = luku1 + luku2
10         luku1 = luku2
```

```
11            luku2 = luku3
12            i=i+1
13    print(luku3)
14    print(len(str(luku3)))
```

Ensiksi määrätään ensimmäiset jäsenet, sekä indeksi i, joka toimii ehtona ja määrää samalla montako jäsentä lasketaan. Ennen varsinaista toistoa tulostetaan kaksi ensimmäistä jäsentä, ja loput tulostetaan loopissa. Kolmanteen lukuun tallennetaan seuraava jäsen, eli kahden edellisen summa. Tämän jälkeen täytyy luvut 1 ja 2 muuttaa seuraaviksi, jotta siirrytään lukujonossa eteenpäin. Lopuksi tulostetaan uusi jäsen ja kasvatetaan ehtona käytettyä muuttujaa i yhdellä.

4 Robotin rakentaminen

Tämä osio on suomennettu ja pienin muokkauksin suoraan kopioitu CamJamin omasta ohjeesta numero 2, joka on kaikkien muiden ohjeiden ohella, saatavilla osoitteessa https://github.com/CamJam-EduKit/EduKit3/tree/master/CamJam%20Edukit%203%20-%20GPIO%20Zero.

Tarvikkeet joita tässä vaiheessa tarvitaan ovat:

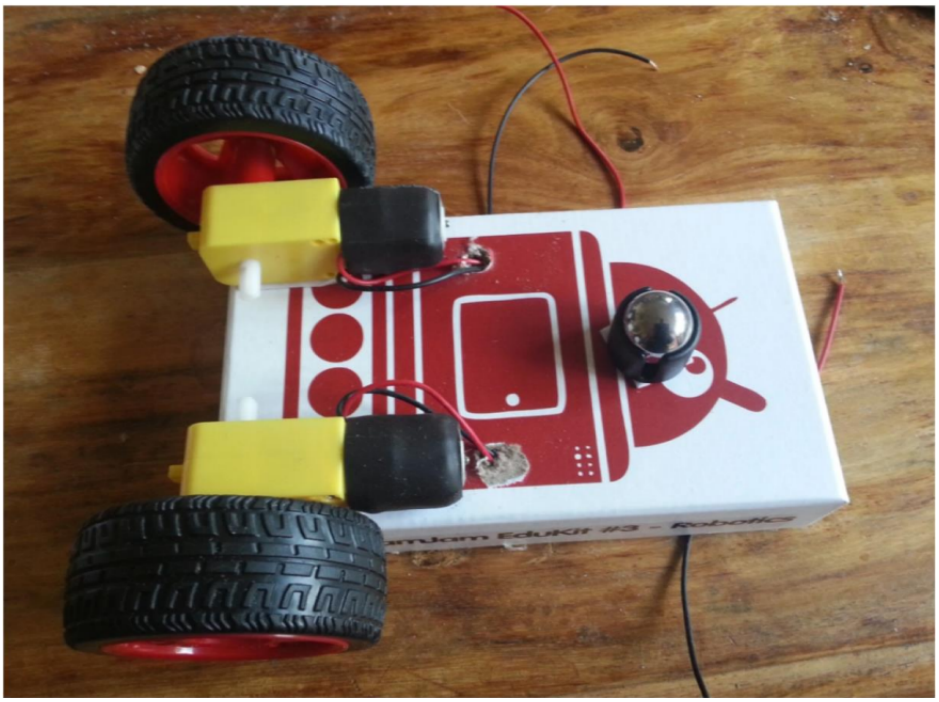
- Raspberry Pi sisältäen Raspbianin.
- Moottorin ohjausalusta (Motor Controller Board)
- Kaksi moottoria
- Pyörät
- Paristokotelo
- Kuula
- Kaksipuolista teippiä osien kiinnittämiseen.
- Pieni ruuvimeisseli johtimien kiinnittämiseen.
- CamJam-edukit-laatikko (tai vastaava rungoksi kelpaava laatikko), johon robotti asennetaan.

Robotin voi periaatteessa rakentaa varsin vapaamuotoisesti. Paketin mukana tulevat pyörät ovat varsin kovaa kumia, joten suurin osa Robotin massasta (virtalähde, paristokotelo ja Raspberry Pi) kannattaa pyrkiä kiinnittämään mahdollisimman hyvin pyörien päälle, jotta kitka riittää liikuttamaan robottia kunnolla.

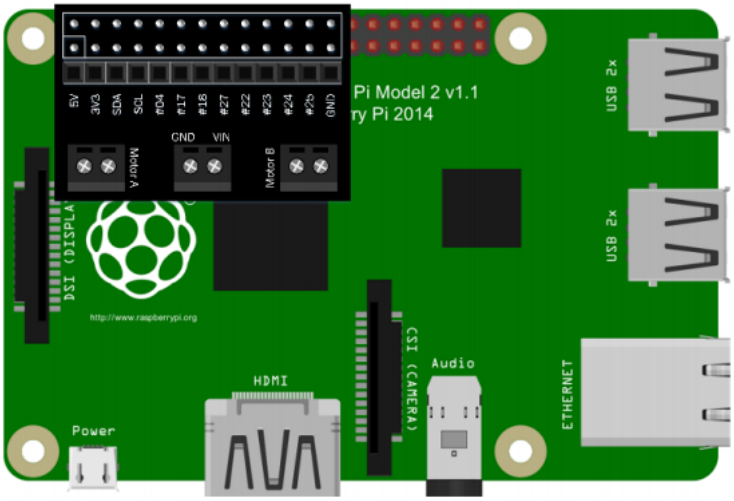
Kiinnitä moottorit mahdollisimman suoraan laatikon toiseen päähän ja kuula hahloineen toiseen päähän käyttäen kaksipuolista teippiä. Moottorien viereen kannattaa tehdä pienet reiät, joista johtimet voi viedä läpi. Lopputulos voi näyttää esimerkiksi kuvan 6 kaltaiselta.

Moottorin ohjauslauta kytketään raspin GPIO (General purpose input/output) pinneihin, jotka ovat piirilevystä pystyssä olevia teräviä tikkuja. Lauta kytketään reunimmaisiin pinneihin kuvan 7 osoittamalla tavalla siten, että lauta jää piirilevyn päälle.

Kun olet testannut miten lauta kiinnitetään levyyn, kannattaa se vielä irrottaa ja kytkeä moottorien johtimet siihen ennen kuin sen asettaa lopullisesti paikoilleen. Moottorit ja



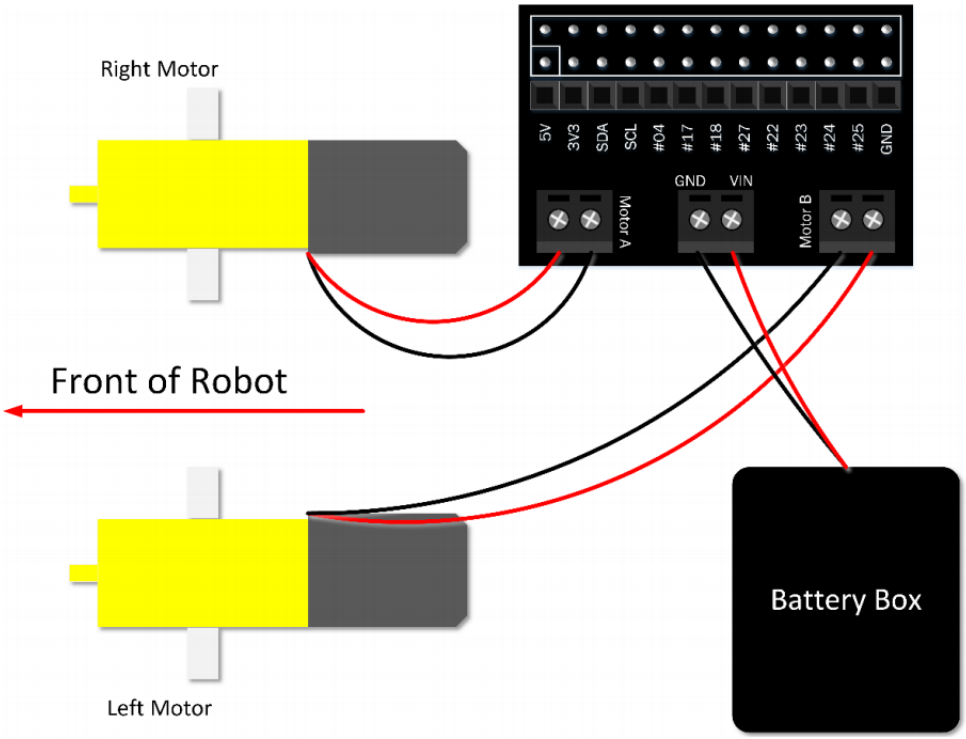
Kuva 6: Robotin osat kiinnitettynä



Kuva 7: Moottorien ohjauslaudan kytkeminen

paristokotelo kytketään lautaan kuvan 8 mukaisesti. Erityisesti paristokotelon kytkennässä tulee olla tarkkana, sillä piirilevy voi vahingoittua jos johtimet kytkee väärin päin. Varmista siis kahdesti, että musta johto menee liittimeen, joka on merkitty GND, ennen kuin kytket mitään päälle. Liittimissä on pienet ruuvit, joiden avaamiseen ja kiinnittämiseen tarvitset pientä ristipäistä ruuvimeisseliä. Tällä ohjeella rakennettavan robotin etuosassa on kaksi pyörää, ja kääntymisen mahdollistava kuula on takaosassa. Voit kuitenkin periaatteessa rakentaa robotin kumminpäin haluat. *Varoitus:* Moottorien johtimien väri voi vaihdella, joten ne saattavat aluksi pyöriä vääriin suuntiin. Tämä korjataan myöhemmin.

Kun olet kiinnittänyt ohjauslevyn takaisin Raspberry Pihin, robotti on valmis. Paristokotelo antaa virtaa ainoastaan moottoreille, joten mikäli haluat että robotti pystyy liikkumaan muutenkin kuin virtajohtonsa perässä, tarvitset vielä virtalähteen raspille.



Kuva 8: Moottorien ja paristokotelon kytkennät.

Kätevimmin tämä käy tavallisella matkapuhelimille tarkoitetulla varavirtalähteellä, josta löytyy micro-usb-liitäntä. Myös tämä virtalähde kannattaa pyrkiä kiinnittämään mahdollisimman hyvin pyörien päälle.

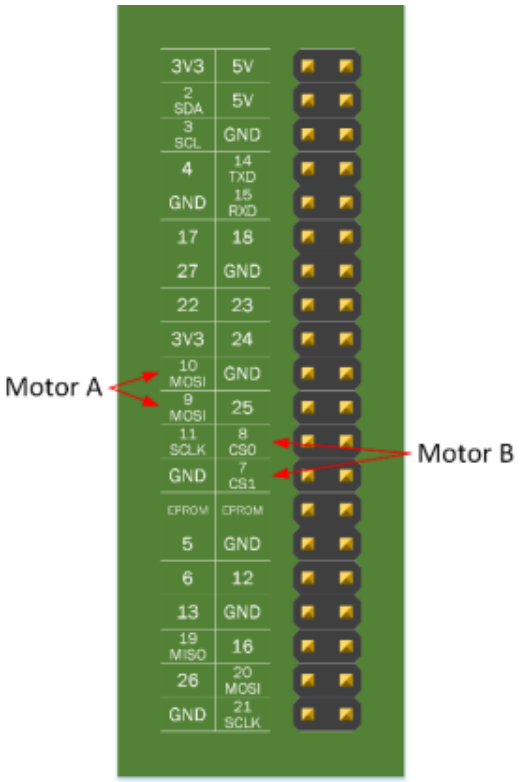
5 Robotin liikuttaminen

Ensimmäinen tehtävä on saada robotti hieman liikkeelle ja varmistaa, että molemmat pyörät pyörivät samaan suuntaan.

Moottorit on kytketty kytkentäalustan kautta suoraan GPIO-pinneihin. Pinnit ovat varsin yksinkertaisia, ja ainoa mitä niille voidaan tehdä on kytkeä päälle tai pois. Raspberry Pin:n GPIO-pinneihin voi siis ohjelmallisesti kytkeä jännitteen, joka kertoo moottorille, että sen pitää pyöriä. Moottorien kytkentäalusta tekee fyysisestä kytkennästä helppoa, mutta meidän täytyy silti tietää mitä pinnejä kytketään päälle ja mitä pois, jotta moottori saadaan pyörimään. Kummallekin moottorille on varattu kaksi pinniä. Oikean puoleista moottoria (Moottori A) ohjataan gpio-pinneillä 9 ja 10 ja vasemman puoleista moottoria (Moottori B) ohjataan pinneillä 7 ja 8. Kytkennät näkee kuvasta 9.

Toinen moottorille kytketystä pinnistä kääntää sitä eteenpäin ja toinen taaksepäin. Mikäli molemmat ovat päällä tai pois samaan aikaan, moottori ei liiku lainkaan. Tavoitteena on, että oikean puoleinen moottori kääntyy pinnillä 10 eteenpäin ja pinnillä 9 taaksepäin. Vastaavasti vasemman moottorin tulisi kääntyä eteenpäin kun pinniin 8 on kytketty jännite, ja taaksepäin, kun pinniin 7 on kytketty jännite.

Pythonissa on tapana, että kaikki ominaisuudet eivät ole oletuksena mukana, vaan niitä otetaan käyttöön niin sanottujen kirjastojen avulla. Kirjastoissa on valmiita funktioita, joiden avulla voi tehdä monenlaisia asioita. GPIO-pinnien ohjaamiseen on olemassa kirjasto nimeltä RPi.GPIO ja se otetaan käyttöön seuraavassa esimerkkikoodissa. Kirjoita



Kuva 9: Moottorien kytkennät GPIO-pinneihin

koodi tiedostoon, ja aja se.

Aseta robotti jonkin telineen päälle, siten että sen pyörät eivät kosketa pöytää tai lattiaa, jottei se lähde liikkeelle kun ajat koodin. Se voi olla yllättävän nopea!

```
1  import RPi.GPIO as GPIO # GPIO-pinnien käyttöön tarkoitettu
       kirjasto
2  import time # Aikaa vaativien tehtävien käyttöön tarkoitettu
       kirjasto
3
4  # Asetetaan GPIO-tila
5  GPIO.setmode(GPIO.BCM)
6  GPIO.setwarnings(False)
7
8  # Asetetaan käyttöön otettavat pinnit
9  GPIO.setup(7, GPIO.OUT)
10 GPIO.setup(8, GPIO.OUT)
11 GPIO.setup(9, GPIO.OUT)
12 GPIO.setup(10, GPIO.OUT)
13
14 # Asetetaan molemmat moottorit ensin pois päältä
15 GPIO.output(7, 0)
16 GPIO.output(8, 0)
17 GPIO.output(9, 0)
18 GPIO.output(10, 0)
19
```

```
20 # Käännetään oikeaa moottoria eteenpäin
21 GPIO.output(9, 0)
22 GPIO.output(10, 1)
23
24 # Käännetään vasenta moottoria eteenpäin
25 GPIO.output(7, 0)
26 GPIO.output(8, 1)
27
28 # odotetaan 1 sekunti, jotta pyörät ehtivät pyöriä
29 time.sleep(1)
30
31 # Alustetaan GPIO-pinnit ja samalla sammutetaan moottorit
32 GPIO.cleanup()
```

Kaikki #-merkillä varustetut rivit ovat kommentteja, ja niiden mukaan ottaminen on valinnaista, mutta ohjelmakoodeja kannattaa aina kommentoida varsin huolellisesti, jotta tiedät myöhemminkin, mitä olet tehnyt.

Kun ajat koodin, pitäisi molempien renkaiden pyöriä sekunnin ajan. Muuttamalla time.sleep()-funktion sisällä olevaa aikaa, voit säätää kuinka pitkään renkaat pyörivät. Varmista tässä vaiheessa, että molemmat renkaat pyörivät nimenomaan eteenpäin. Jos näin ei ole, vaihda renkaiden kytkentä toisinpäin kytkentälaudalta.

6 Kääntyminen ja ajaminen

Monimutkaisempien ohejlmien kirjoittaminen vaatii, että koodin rakenne on järkevä. Funktiot mahdollistavat komentojen kokoamisen yhden komennon alle, jolloin samaa koodipätkää voi käyttää monta kertaa ilman että sitä tarvitsee joka kerta kirjoittaa kokonaan uudelleen. Usein funktioille annetaan *argumentteja*, jolloin saman koodin voi ajaa useammalla eri arvolla. Esimerkkikoodi havainnollistaa tätä.

```
1 def yhteen(luku1, luku2):
2         summa = luku1 + luku2
3         print(summa)
4
5 yhteen(1,2)
6 yhteen(32,10)
```

Robotin liikuttamisen kannalta funktiot ovat oleellisia, sillä niiden avulla voidaan asettaa robotti liikkumaan eteenpäin tai taaksepäin ilman edellisen luvun pitkiä koodipätkiä. Seuraavassa koodissa määritellään funktiot, joiden avulla robottia voi liikuttaa eteenpäin, taaksepäin ja pysäyttää moottorit kokonaan. Eteen ja taaksepäin liikkuminen on jo varsin mukavaa, mutta robotin tulisi mielellään osata kääntyäkin. Vasemmalle kääntymistä varten oikean puoleisen moottorin pitäisi pyöriä eteenpäin ja vasemmanpuoleisen taaksepäin. Oikealle kääntymistä varten tietysti päinvastoin. Lisätään funktiomäärittelyjen perään funktiot kääntymistä varten.

```
1 import RPi.GPIO as GPIO # GPIO-pinnien käyttöön tarkoitettu
      kirjasto
2 import time # Aikaa vaativien tehtävien käyttöön tarkoitettu
      kirjasto
```

```
3
4  # Asetetaan GPIO-tila
5  GPIO.setmode(GPIO.BCM)
6  GPIO.setwarnings(False)
7
8  #Tallennetaan muuttujat pinneille
9  #A tarkoittaa oikeanpuoleista moottoria ja B vasenta
10 pinAEteen = 10
11 pinATaakse = 9
12 pinBEteen = 8
13 pinBTaakse = 7
14
15 #Asetetaan jokainen pinni
16 GPIO.setup(pinAEteen, GPIO.OUT)
17 GPIO.setup(pinATaakse, GPIO.OUT)
18 GPIO.setup(pinBEteen, GPIO.OUT)
19 GPIO.setup(pinBTaakse, GPIO.OUT)
20
21 #Funktio, jolla sammutetaan moottorit
22 def stop():
23         GPIO.output(pinAEteen, 0)
24         GPIO.output(pinATaakse, 0)
25         GPIO.output(pinBEteen, 0)
26         GPIO.output(pinBTaakse, 0)
27
28 #Funktio jolla robotti kulkee eteenpäin aika-muuttujan kertoman
      ajan
29 def eteen(aika):
30         GPIO.output(pinAEteen, 1)
31         GPIO.output(pinATaakse, 0)
32         GPIO.output(pinBEteen, 1)
33         GPIO.output(pinBTaakse, 0)
34         time.sleep(aika)
35         stop()
36
37 #Funktio jolla robotti kulkee taaksepäin aika-muuttujan kertoman
      ajan
38 def taakse(aika):
39         GPIO.output(pinAEteen, 0)
40         GPIO.output(pinATaakse, 1)
41         GPIO.output(pinBEteen, 0)
42         GPIO.output(pinBTaakse, 1)
43         time.sleep(aika)
44         stop()
45
46 def vasemmalle(aika):
47         GPIO.output(pinAEteen, 1)
48         GPIO.output(pinATaakse, 0)
```

```
49          GPIO.output(pinBEteen, 0)
50          GPIO.output(pinBTaakse, 1)
51          time.sleep(aika)
52          stop()
53
54  def oikealle(aika):
55          GPIO.output(pinAEteen, 0)
56          GPIO.output(pinATaakse, 1)
57          GPIO.output(pinBEteen, 1)
58          GPIO.output(pinBTaakse, 0)
59          time.sleep(aika)
60          stop()
61
62  #Kutsutaan funktioita. Aika-argumentti annetaan sekunteina.
63  eteen(3)
64  vasemmalle(2)
65  taakse(2)
66
67  #Tämä komento kannattaa ajaa jokaisen ohjelman päätteeksi
68  GPIO.cleanup()
```

Näillä funktioilla voitkin jo ohjelmoida robotin kulkemaan sellaisen reitin kuin haluat.

7 Värin tunnistus

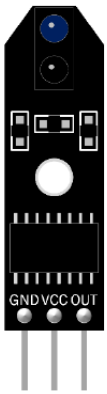
Camjam-robotin mukana tulee kaksi sensoria, joista toisella voidaan erottaa musta ja valkoinen väri toisistaan ja toisella mitata etäisyyksiä kohteesta. Näiden sensorien avulla voidaan rakentaa esimerkiksi robotti, joka osaa seurata mustaa viivaa valkealla alustalla. Aloitetaannkin siitä, että saadaan robotti erottamaan musta ja valkoinen väri toisistaan.

Värin tunnistava sensori näyttää kuvan 10 mukaiselta. Värin tunnistus on tosin hieman liioiteltua, sillä sensori kykenee todellisuudessa erottamaan vain mustan ja valkoisen valon toisistaan. Se koostuu periaatteessa kahdesta osasta, eli lampusta joka lähettää valoa tietyllä taajuudella ja valosensorista joka mittaa pinnasta heijastuneen valon määrää tällä kyseisellä taajuudella. Sensorissa on kolme pinniä, joista kahta käytetään virran kytkemiseksi, ja kolmas ilmaisee syötteellä minkä värisellä pinnalla sensori on. Mikäli heijastuvan valon määrä on vähäinen, eli pinta on musta, ei sensori syötä virtaa pinnille. Jos taas valoa heijastuu runsaasti, on pinta valkoinen ja tällöin sensori kytkee jännitteen pinnille. Laite luonnollisesti jakaa kaikki muutkin värit joko mustaksi tai valkoiseksi riippuen pinnan heijastusominaisuudesta.

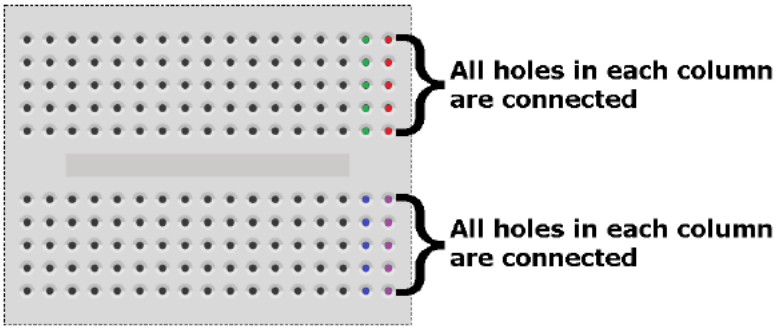
Sensorin kytkennässä käytetään mukana tullutta kytkentäalustaa (breadboard) sekä johtimia. Kytkentäalusta mahdollistaa komponenttien kytkemisen kohtalaisen kätevästi ilman, että niitä tarvitsee juottaa kiinni toisiinsa. Niitä käytetäänkin yleensä piirien suunnittelussa ja prototyyppien valmistuksessa, jotta laitteiden testaaminen on helpompaa. Pakkauksen mukana tulevassa kytkentäalustassa on kaksi osaa, joiden pystysarakkeet on kytketty toisiinsa kuvan 11 mukaisella tavalla.

Huom! Ole erittäin huolellinen kytkiessäsi sensoria. Väärin tehty kytkentä saattaa pahimmassa tapauksessa rikkoa sensorin.

Sensori kytketään kuvan 12 osoittamalla tavalla. Sensorin maa-pinni (GND, ground)



Kuva 10: Mustan ja valkoisen erottava sensori

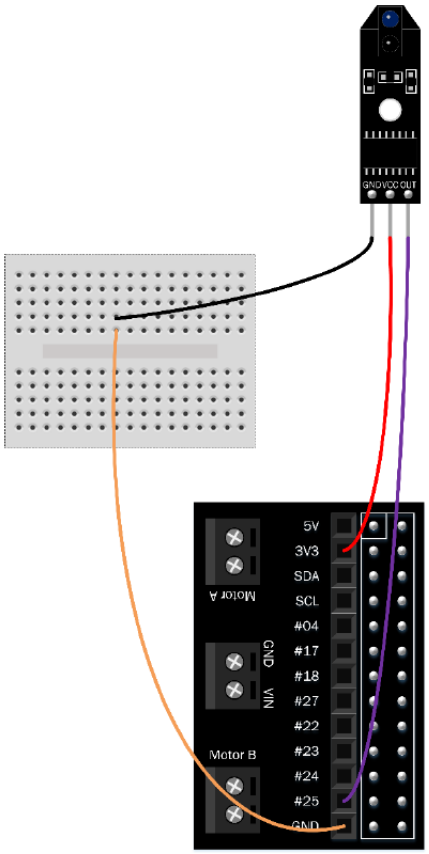


Kuva 11: Kytkentäalusta

kytketään kytkentäalustan kautta moottorin ohjauslaudan maa-napaan. Kytkentäalustaa käytetään, koska samaa liitäntää tarvitaan myös seuraavan anturin liittämiseen. Pinni joka on merkitty VCC antaa sensorille käyttöjännitteen. Se kytketään ohjauslaudan liittimeen, joka on merkitty 3V3, mikä tarkoittaa, että liitäntä antaa 3,3 voltin jännitteen. Kolma pinni, joka on merkitty OUT, antaa signaalin raspille sen mukaan, tulkitseeko sensori näkemänsä värin mustaksi vai valkoiseksi. Se kytketään ohjauslaudan kautta raspin pinniin numero 25. Lukemalla tämän pinnin tilaa, voidaan sensorin antamaa tietoa käyttää ohjelmakoodissa.

Seuraavaksi kopioi seuraava ohjelmakoodi uuteen tiedostoon, tallenna ja aja. Ohjelma lukee sensorin tilan kahden sekunnin välein ja kertoo käyttäjälle näkeekö se mustan vai valkoisen alustan. Vaihtele alustaa sensorin alla ja varmista, että se toimii kuten pitääkin. Ohjelman voi lopettaa painamalla näppäimistöltä Ctrl+c.

```
1  # Viivsensorin testausohjelma
2
3  import RPi.GPIO as GPIO
4  import time
5
6  # Alustetaan GPIO-pinnit
7  GPIO.setmode(GPIO.BCM)
8  GPIO.setwarnings(False)
9
10 # Tallennetaan pinnin numero muuttujaan
11 viivasensori = 25
```



Kuva 12: Sensorin kytkeminen moottorien ohjauslautaan.

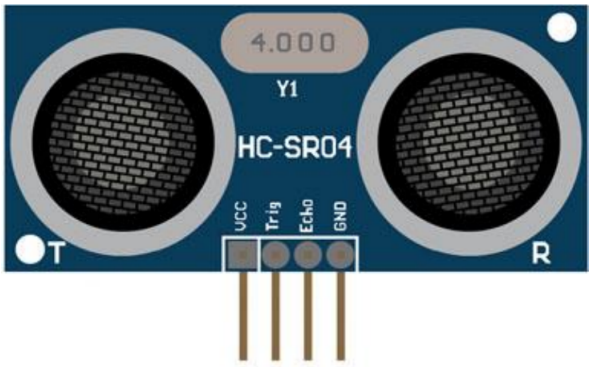
```
12
13 # Tehdään tästä pinnistä input-pinni, jotta sen tila voidaan
       lukea
14 GPIO.setup(viivasensori, GPIO.IN)
15
16 # Try-except-rakenteella voidaan tehdä jotain, kunnes tulee
       poikkeus
17
18 try:
19     # Tämä tekee loopin, jota toistetaan ikuisesti
20     while True:
21         # Jos sensori palauttaa nollan, eli on mustalla
               alustalla
22         if GPIO.input(viivasensori)==0:
23             print('Sensorin mielestä se on mustalla alustalla')
24             print('Poistu painamalla ctrl+c')
25         # Muussa tapauksessa sensori palautta 1, eli on
               valkealla alustalla
26         else:
27             print('Sensorin mielestä se on valkoisella alustalla
                   ')
28             print('Poistu painamalla ctrl+c')
29         # Odotetaan 2 sekuntia, ja toistetaan sama uudelleen
```

```
30            time.sleep(2)
31
32  # KeyboardInterrupt-virhe tulee kun käyttäjä painaa ctrl+c
33  # Tällöin ohjelman ajo lopetetaan, mutta ennen sitä tehdään vielä exceptin sisältö
34  except KeyboardInterrupt:
35      GPIO.cleanup()
36      print('Poistuit painamalla ctrl+c')
```

Ohjelmassa hyödynnetään Try-except-rakennetta, joka mahdollistaa jonkin asian tekemisen niin kauan kunnes tapahtuu jokin virhe, tässä esimerkissä käyttäjä painaa näppäimistöltä Ctrl+c, minkä vuoksi ohjelma kokee niin sanotun KeyboardInterrupt-virheen ja ohjelman ajo lopetetaan. Rakennetta käytetään, jotta pinnien tila nollataan aina ohjelmakoodin lopuksi GPIO.cleanup()-komennolla. Rakenteen toiminnasta ei tässä vaiheessa tarvitse tietää sen enempää. Ennen sensorin lopullista kiinnittämistä, kannattaa kokeilla miltä etäisyydeltä se lukee alustan värin mahdollisimman luotettavasti.

8 Ultraäänianturi etäisyyden mittaamiseen

Seuraavana tavoitteena on pystyä mittaamaan etäisyyksiä. Sen avulla robotti voi tunnistaa eteensä tulevia esteitä ja välttää niitä. Edukit 3:n mukana tulee HR-SC04-ultraäänisensori. Joka näyttää kuvan 13 mukaiselta.



Kuva 13: Etäisyyttä mittaava ultraäänisensori

Sensori kytketään neljän pinnin avulla. Trig-pinniin tuleva signaali saa aikaan sen, että kaiutin, joka on merkitty T-kirjaimella (transmitter) lähettää ultraäänipulssin jonka kesto on n. 10 mikrosekuntia. Vastaavasti kun R-kirjaimella merkitty mikrofoni vastaanottaa kaiun, lähettää sensori Echo-pinniin 5 voltin jännitteen.

Mittaamalla lähetetyn ja vastaanotetun signaalin välinen aikaero voidaan laskea kuinka kaukana kohde on. Kuljettu matka s on

$$s = tv,$$

missä t on matkaan käytetty aika ja v kulkunopeus. Ääni kulkee ilmassa n. 343,26 m/s, joten käyttämällä senttimetrejä saadaan äänen kulkemaksi matkaksi

$$s = t \cdot 34326\,\mathrm{cm/s}.$$

. Lisäksi heijastunut ääni on kulkenut matkan kahdesti, joten etäisyydeksi d saadaan

$$d = \frac{t \cdot 34326\,\mathrm{cm/s}}{2}.$$

Toimiakseen sensori vaatii 5 voltin jännitteen, joka saadaan moottorien ohjauslevyn kautta. Sensori myös lähettää 5 voltin jännitteen, mutta koska Raspberry Pi:n pinneihin voi kytkeä korkeintaan 3,3 voltin jännitteen tulee signaalijännitettä hieman laskea. Tämä tehdään jakamalla jännite input-pinnin ja maa-navan välille käyttämällä vastuksia.

Pakkauksessa tulee mukana useita vastuksia ja ne saattavat olla peräisin eri valmisteeristä minkä vuoksi niissä saatetaan käyttää eri merkintöjä. Vastuksien merkinnöissä käytetään neliviivaista ja viisiviivaista värikoodia. Sensorin kytkemiseksi tarvitaan 330 Ω ja 470 Ω vastukset.

- Neliviivaisella merkinnällä
  - 330 Ω vastus on oranssi, oranssi, ruskea, kulta
  - 470 Ω vastus on keltainen, violetti, ruskea, kulta
- Viisiviivaisella merkinnällä
  - 330 Ω vastus on oranssi, oranssi, musta, musta ruskea
  - 470 Ω vastus on keltainen, violetti, musta, musta ja ruskea

Sensorin kytkentä tehdään kuvan 14 osoittamalla tavalla. Kuvaa katsomalla lienee sanomattakin selvää, että kytkennä tulee tehdä jälleen erittäin huolellisesti, ettei osia hajoa.

Työnnä sensori kiinni kytkentäalustaan siten, että sensorin GND-pinni menee samalle riville, jolta lähtee johdin Pi:n GND-liitäntään. Taivuta vastusten jalkoja ja kytke ne siten, että 330 Ω vastus lähtee riviltä jossa on sensorin Echo-pinni ja päätyy johonkin tyhjään kytkentäalustan riviin. Kytke vastaavasti 470 Ω vastus saman rivin ja GND-pinnin välille. Kytke sitten valitsemasi rivi ohjauslevyn liitäntään numero 18. Katso kuvaa huolellisesti!

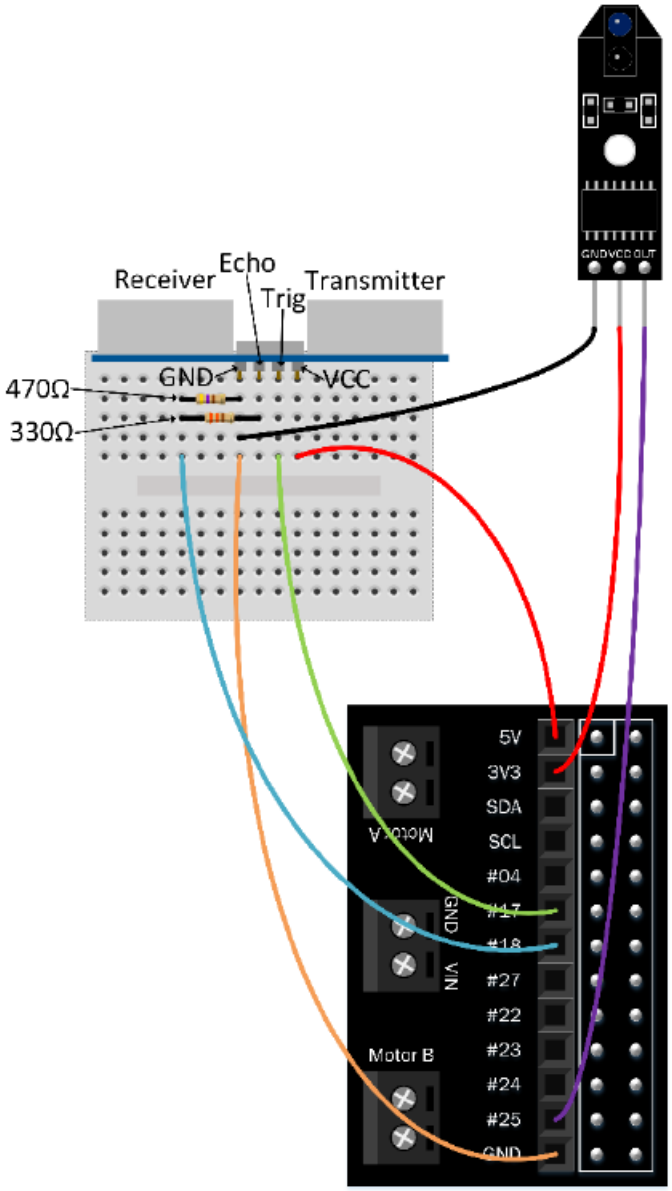
Kytke sitten rivi, jolla on sensorin VCC-pinni johtimella ohjauslevyn päässä olevaan liitäntään jossa lukee 5V. Tästä pinnistä sensori saa käyttöjännitteensä.

Yhdistä johtimella rivi, jolla on sensorin Trig-pinni ohjauslevyn liittimeen numero 17.

Tällä kytkennällä saadaan siis laskettua sensorin antama 5 volttia Raspberry Pi:lle sopivaan 3,3 volttiin ja sensorin lähettämä signaali voidaan lukea.

Kirjoita seuraava ohjelmakoodi tiedostoon ja kokeile ajaa. Ohjelma saattaa sisältää elementtejä, joihin et ole vielä tutustunut, mutta älä anna sen häiritä.

```
1  #Etäisyysanturin käyttö
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setwarnings(False)
8
9  #Asetetaan pinnien numerot muuttujiin
10 pinLahetys = 17 # Pinni jolla pulssi lähetetään
```



Kuva 14: Ultraäänianturin kytkentä

```
11 pinVastaanotto = 18 # Pinni joka lukee kaiun
12
13 print("Etäisyyden mittaus ultraäänellä. Lopeta painamalla ctrl+c
       ")
14
15 #Asetetaan pinnit lähettämään ja vastaanottamaan dataa
16 GPIO.setup(pinLahetys, GPIO.OUT) # Lähettävä pinni
17 GPIO.setup(pinVastaanotto, GPIO.IN) #Vastaanottava pinni
18
19 try:
20     while True:
21         GPIO.output(pinLahetys, False)
22         # Sensorin valmistautumiseen menee hetki, odotetaan sen
```

```
            aikaa
23          time.sleep(0.5)
24          # Lähetetään 10 mikrosekunnin pulssi
25          GPIO.output(pinLahetys, True)
26          time.sleep(0.00001)
27          GPIO.output(pinLahetys, False)
28
29          #Käynnistetään ajastin
30          LahtoAika = time.time()
31
32          # Lähtöaika nollataan, kunnes vastaanottimelta saadaan
                signaali
33          while GPIO.input(pinVastaanotto)==0:
34              LahtoAika = time.time()
35
36          # Lopetetaan kun signaali häviää
37          while GPIO.input(pinVastaanotto)==1:
38              LoppuAika = time.time()
39              # Jos kohde on liian lähellä, Pi ei pysty erottamaan
                    etäisyyttä hyvin
40              # Reagoidaan siihen ja kerrotaan mitä on tapahtunut
41              if LoppuAika - LahtoAika >= 0.04:
42                  print("Hei, olet liian lähellä, mene kauemmas")
43                  LoppuAika = LahtoAika
44                  break
45
46          # Lasketaan etäisyys senttimetreissä
47          etaisyys = (LoppuAika - LahtoAika)*34326/2
48          print("Etäisyys: %.1f cm" % etaisyys)
49          time.sleep(1) #Mittaus suoritetaan sekunnin välein
50
51  #Ohjelman lopetus painamalla ctrl+c
52  except KeyboardInterrupt:
53      GPIO.cleanup()
54      print(" Poistuit ohjelmasta")
```

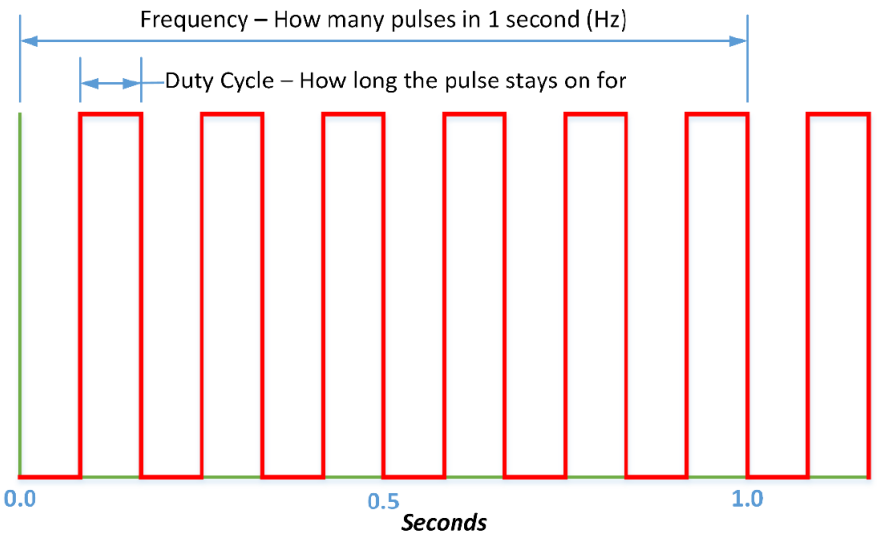
9 Moottorien kalibrointi

Testatessasi moottorien toimintaa aiemmassa luvussa huomasit ehkä, että ne pyörivät varsin lujaa, miikä tekee robotin ohjaamisesta haastavaa. Lisäksi robotti saattaa kulkea hieman vinoon, vaikka olisit asentanut pyörät hyvin huolellisesti. Tämä johtuu siitä, että moottorit eivät ole välttämättä kovin tarkkoja, ja toinen saattaa pyöriä samalla jännitteellä hieman kovempaa kuin toinen. Tämän vuoksi moottoreita pitää hieman kalibroida, jotta robotin ohjaaminen onnistuu tarkasti.

Moottorin ohjaaminen tapahtuu vain kytkemällä pinni päälle tai pois, joten herää kysymys miten sitä voidaan säätää. Yksinkertainen sähkömoottori pyörii periaatteessa sitä nopeammin, mitä suurepi jännite siihen kytketään. Näin ollen robottia voisi hidastaa

esimerkiksi vastuksen avulla. Tämä tekisi nopeuden säätämisestä ohjelmallisesti kuitenkin mahdotonta, joten ohjelmallinen ratkaisu on tarpeen.

Pythonin GPIO-kirjastossa on olemassa juuri sopiva työkalu tähän tarkoitukseen. Se on nimeltään PWM (pulse width modulation). PWM:n avulla voidaan moottorille syötettyä pulssia säätää siten, että pulssia katkotaan niin nopeasti että moottorin pyörintä näyttää kuitenkin jatkuvalta, mutta tapahtuu hitaammin.



Kuva 15: Virransyöttöä moottoriin voidaan jaksottaa

Moottorin säätöön vaikuttaa kaksi parametria, vapaasti suomennettuna taajuus (frequency) ja työosa (duty cycle). Taajuus määrittää kuinka monta pulssia sekunnin aikana tapahtuu, eli kuinka tiheään moottorilta katkaistaan virta. Yksi pulssi sisältää ajan, jonka moottori on päällä, sekä ajan jonka se on poissa. Kun moottoriin seuraavan kerran kytketään virta alkaa uusi pulssi. Tätä havainnollistetaan kuvassa 15.

Työosa-parametri kertoo prosentteina kuinka suuren osan ajasta virta on kytkettynä. Näin ollen esimerkiksi 50 Hz taajuus ja 50% työosa määrittää moottoriin syötetyn pulssin pituudeksi 0,02 sekuntia ja siitä puolet syötetään virtaa, eli 0,01 sekuntia kerrallaan. Pinni syöttää 3,3 voltin jännitteen, joka on päällä puolet ajasta, joten moottori kokee jännitteen puolittuneena, eli 1,65 volttina. Tämän voi itseasiassa mitata myös yleismittarin vaihtojännite-asetuksella. Arvossa saattaa esiintyä pientä heiluntaa.

Työosaparametria säätämällä voidaan siis suoraan asettaa prosentteina moottorin pyörimisnopeus 100% ollessa täysillä ja siitä alaspäin aina osia maksiminopeudesta.

Muokataan seuraavaksi koodia, joka kirjoitettiin aiemmin moottorien liikuttamiseksi.

```
1 import RPi.GPIO as GPIO # GPIO-pinnien käyttöön tarkoitettu
      kirjasto
2 import time # Aikaa vaativien tehtävien käyttöön tarkoitettu
      kirjasto
3
4 # Asetetaan GPIO-tila
5 GPIO.setmode(GPIO.BCM)
6 GPIO.setwarnings(False)
7
8 #Tallennetaan muuttujat pinneille
```

```
9  #A tarkoittaa oikeanpuoleista moottoria ja B vasenta
10 pinAEteen = 10
11 pinATaakse = 9
12 pinBEteen = 8
13 pinBTaakse = 7
14
15 # Määritetään taajuus ja työosa muuttujiin
16 freq = 35
17 duty = 40 #työosa välillä 0-100 määrää nopeuden
18 seis = 0 #tätä käytetään työosana, kun moottori pysäytetään
19
20 #Asetetaan jokainen pinni
21 GPIO.setup(pinAEteen, GPIO.OUT)
22 GPIO.setup(pinATaakse, GPIO.OUT)
23 GPIO.setup(pinBEteen, GPIO.OUT)
24 GPIO.setup(pinBTaakse, GPIO.OUT)
25
26 # Asetetaan GPIO käyttämään PWM-ohjelmaa edellä määrätyllä
       taajuudella
27 pwmAEteen = GPIO.PWM(pinAEteen, freq)
28 pwmATaakse = GPIO.PWM(pinATaakse, freq)
29 pwmBEteen = GPIO.PWM(pinBEteen, freq)
30 pwmBTaakse = GPIO.PWM(pinBTaakse,freq)
31
32 # Aloitetaan ohjelma siitä, että moottorit ovat pysähdyksissä
33 pwmAEteen.start(seis)
34 pwmATaakse.start(seis)
35 pwmBEteen.start(seis)
36 pwmBTaakse.start(seis)
37
38 #Funktio, jolla sammutetaan moottorit
39 def stop():
40     pwmAEteen.ChangeDutyCycle(seis)
41     pwmATaakse.ChangeDutyCycle(seis)
42     pwmBEteen.ChangeDutyCycle(seis)
43     pwmBTaakse.ChangeDutyCycle(seis)
44
45 #Funktio jolla robotti kulkee eteenpäin aika-muuttujan kertoman
       ajan
46 def eteen(aika):
47     pwmAEteen.ChangeDutyCycle(duty)
48     pwmATaakse.ChangeDutyCycle(seis)
49     pwmBEteen.ChangeDutyCycle(duty)
50     pwmBTaakse.ChangeDutyCycle(seis)
51     time.sleep(aika)
52     stop()
53
54 #Funktio jolla robotti kulkee taaksepäin aika-muuttujan kertoman
```

```
    ajan
def taakse(aika):
    pwmAEteen.ChangeDutyCycle(seis)
    pwmATaakse.ChangeDutyCycle(duty)
    pwmBEteen.ChangeDutyCycle(seis)
    pwmBTaakse.ChangeDutyCycle(duty)
    time.sleep(aika)
    stop()

def vasemmalle(aika):
    pwmAEteen.ChangeDutyCycle(duty)
    pwmATaakse.ChangeDutyCycle(seis)
    pwmBEteen.ChangeDutyCycle(seis)
    pwmBTaakse.ChangeDutyCycle(duty)
    time.sleep(aika)
    stop()

def oikealle(aika):
    pwmAEteen.ChangeDutyCycle(seis)
    pwmATaakse.ChangeDutyCycle(duty)
    pwmBEteen.ChangeDutyCycle(duty)
    pwmBTaakse.ChangeDutyCycle(seis)
    time.sleep(aika)
    stop()

#Kutsutaan funktioita. Aika-argumentti annetaan sekunteina.
eteen(3)
vasemmalle(2)
taakse(2)

#Tämä komento kannattaa ajaa jokaisen ohjelman päätteeksi
GPIO.cleanup()
```

Koska moottorit eivät välttämättä pyöri samalla jännitteellä täsmälleen samalla nopeudella, voidaan ne säätää pyörimään yhtä nopeasti säätämällä työosat molemmille erikseen. Voit lisätä koodiin molemmille moottoreille oman duty-muuttujan ja käyttämällä sitä.

```
dutyA = 40
dutyB = 40
```

Jotta saat robotin kulkemaan suoraan, joudut tekemään hieman testejä ja säätämään työosaa erikseen siten että molemmat moottorit toimivat suurinpiirtein samalla nopeudella. Voit tarvittaessa käyttää myös desimaalilukuja, jolloin säätö onnistuu hyvinkin tarkasti.

10 Viivan seuraaminen

Nyt kun robotti osaa erottaa värejä ja sen liikkeet ovat hallitumpia, voidaan siirtyä tekemään robotista jollain tavalla älykästä. Värin tunnistuksella robotin saa esimerkiksi seuraamaan mustaa viivaa valkoisella alustalla, tai ajelemaan mustan alueen sisäpuolella.

Viivan seuraaminen vaatii kahta toimintoa. Jos sensori näkee mustaa, niin ajetaan eteenpäin. Jos taas sensori näkee valkoista, pysähdytään ja etsitään mustaa. Tämän voi tehdä esimerkiksi kääntymällä hieman vasemmalle, oikealle ja taas vasemmalle kunnes viiva löytyy.

Etsin voidaan toteuttaa seuraavan esimerkkikoodin avulla. Se perustuu aiemmassa luvussa tehtyyn koodiin jolla tunnistettiin värejä, joten ota se pohjaksi ja rakenna siitä uusi koodi.

```
1  # Viivsensorin testausohjelma
2
3  import RPi.GPIO as GPIO
4  import time
5
6  # Alustetaan GPIO-pinnit
7  GPIO.setmode(GPIO.BCM)
8  GPIO.setwarnings(False)
9
10 # Tallennetaan pinnin numero muuttujaan
11 viivasensori = 25
12
13 # Tehdään tästä pinnistä input-pinni, jotta sen tila voidaan
       lukea
14 GPIO.setup(viivasensori, GPIO.IN)
15
16 #Tallennetaan muuttujat pinneille
17 #A tarkoittaa oikeanpuoleista moottoria ja B vasenta
18 pinAEteen = 10
19 pinATaakse = 9
20 pinBEteen = 8
21 pinBTaakse = 7
22
23 # Määritetään taajuus ja työosa muuttujiin
24 freq = 35
25 duty = 40 #työosa välillä 0-100 määrää nopeuden
26 seis = 0 #tätä käytetään työosana, kun moottori pysäytetään
27
28 #Asetetaan jokainen pinni
29 GPIO.setup(pinAEteen, GPIO.OUT)
30 GPIO.setup(pinATaakse, GPIO.OUT)
31 GPIO.setup(pinBEteen, GPIO.OUT)
32 GPIO.setup(pinBTaakse, GPIO.OUT)
33
34 # Asetetaan GPIO käyttämään PWM-ohjelmaa edellä määrätyllä
       taajuudella
35 pwmAEteen = GPIO.PWM(pinAEteen, freq)
36 pwmATaakse = GPIO.PWM(pinATaakse, freq)
37 pwmBEteen = GPIO.PWM(pinBEteen, freq)
38 pwmBTaakse = GPIO.PWM(pinBTaakse, freq)
39
```

```
40 # Aloitetaan ohjelma siitä, että moottorit ovat pysähdyksissä
41 pwmAEteen.start(seis)
42 pwmATaakse.start(seis)
43 pwmBEteen.start(seis)
44 pwmBTaakse.start(seis)
45
46 #Funktio, jolla sammutetaan moottorit
47 def stop():
48     pwmAEteen.ChangeDutyCycle(seis)
49     pwmATaakse.ChangeDutyCycle(seis)
50     pwmBEteen.ChangeDutyCycle(seis)
51     pwmBTaakse.ChangeDutyCycle(seis)
52
53 #Funktio jolla robotti kulkee eteenpäin aika-muuttujan kertoman
      ajan
54 def eteen():
55     pwmAEteen.ChangeDutyCycle(duty)
56     pwmATaakse.ChangeDutyCycle(seis)
57     pwmBEteen.ChangeDutyCycle(duty)
58     pwmBTaakse.ChangeDutyCycle(seis)
59
60 #Funktio jolla robotti kulkee taaksepäin aika-muuttujan kertoman
      ajan
61 def taakse():
62     pwmAEteen.ChangeDutyCycle(seis)
63     pwmATaakse.ChangeDutyCycle(duty)
64     pwmBEteen.ChangeDutyCycle(seis)
65     pwmBTaakse.ChangeDutyCycle(duty)
66
67 def vasemmalle():
68     pwmAEteen.ChangeDutyCycle(duty)
69     pwmATaakse.ChangeDutyCycle(seis)
70     pwmBEteen.ChangeDutyCycle(seis)
71     pwmBTaakse.ChangeDutyCycle(duty)
72
73 def oikealle():
74     pwmAEteen.ChangeDutyCycle(seis)
75     pwmATaakse.ChangeDutyCycle(duty)
76     pwmBEteen.ChangeDutyCycle(duty)
77     pwmBTaakse.ChangeDutyCycle(seis)
78
79 # Funktio, jolla tarkistetaan näkeekö sensori viivan
80 # Funktio saa arvon tosi, jos sensori on mustan päällä
81 def mustanpaalla():
82     if GPIO.input(viivasensori) == 0:
83         return True
84     else:
85         return False
```

```
86
87 ## Funktio, jolla etsitään mustaa viivaa
88 def etsiviiva():
89     print("Etsitään viivaa")
90     # Valitaan suunnat totuusarvoiksi. True = vasen, False =
       oikea
91     suunta = True
92     etsintaAika = 0.2 # Käännytään 0.2 sekuntia
93     etsinta = 1 # lasketaan monestiko robotti on kääntynyt
94     maxEtsinta = 5 # Tämän enempää ei kääntyillä
95
96     # Käännellään robottia kunnes viiva löytyy tai se on etsinyt
       riittävän kauan
97     while etsinta <= maxEtsinta:
98         # asetetaan kääntymisaika
99         etsintaAika = etsinta*etsintaAika
100
101        if suunta:
102            print("Etsitään vasemmalta")
103            vasemmalle()
104        else:
105            print("Etsitään oikealta")
106            oikealle()
107
108        # Talletetaan aika jolloin etsintä aloitetiin
109        alkuaika = time.time()
110
111        while time.time() - alkuaika <= etsintaAika:
112            if mustanpaalla():
113                stop()
114                return True # lopetetaan etsintäfunktio ja
                       palautetaan tosi
115
116        stop() # ei löytynyt joten pysähdytään
117        etsinta = etsinta+1 # lisätään etsintäkertoja
118        suunta = not suunta # Muutetaan suuntaa
119
120    # Jos viivaa ei löydy ajoissa, palautetaan epätosi
121    return False
122
123 try:
124     print("Viivan seuraaja")
125     while True:
126         if mustanpaalla():
127             eteen()
128         else:
129             stop()
130             if etsiviiva():
```

```
131                     print("Seuraan viivaa!")
132                 else:
133                     stop()
134                     print("Viiva hukkui!")
135                     GPIO.cleanup()
136                     exit()
137 
138 except KeyboardInterrupt:
139     GPIO.cleanup()
```