Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Jyri Pietikäinen

Mobiilirobotiikan hyödyntäminen koneautomaation opinnoissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Konetekniikka

Insinöörityö

14.4.2021

Tekijä Otsikko Sivumäärä Aika	Jyri Pietikäinen Mobiilirobotiikan hyödyntäminen koneautomaation opinnoissa 62 sivua, joista 20 sivua liitteitä. 14.4.2021
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Konetekniikka
Ammatillinen pääaine	Koneautomaatio
Ohjaajat	Lehtori Antti Liljaniemi

Tässä opinnäytetyössä tutkittiin mobiilirobotiikkaa ja niihin liittyviä toimintaperiaatteita yleisellä tasolla. Opinnäytetyössä kirjoitettiin teoriaosuus mobiilirobotiikkaan liittyen ja pohdittiin robotiikan käyttöä Metropolian koneautomaation opinnoissa.

Opinnäytetyössä tutustuttiin myös DOBOT-opetusrobotiikkaan, jota hyödyntämällä luotiin opetusmateriaalia Metropolian koneautomaation opintopolun tarpeisiin. Opetusrobotiikkana käytettiin auton kaltaista laitetta, joka on varusteltu rikkaasti erilaisin ohjelmoitavin komponentein. Opinnäytetyössä tutustuttiin tarkemmin komponenttien toimintaperiaatteisiin sekä suoritettiin kokoonpano laitteistolle, joka kuvattiin opinnäytetyössä.

Kokoonpanon jälkeen opinnäytetyössä tutustuttiin DobotBlock- ja Arduino IDE- ohjelmistoihin. Ohjelmistoja käyttäen luotiin kaksi eri ohjelmaa kahdella eri metodilla samalla näihin tutustuen. Opinnäytetyössä kuvattiin ohjelmien toimintaperiaatteet sekä ohjelmointimetodien perusteita.

Ohjelmien pohjalta luotiin kaksi erilaista harjoitustyötä, jotka perustuvat näihin kahteen eri ohjelmointimetodiin. Harjoitustöistä luotiin laboratoriotyöohjeet. Ensimmäisessä työssä luotiin graafisella ohjelmoinnilla ohjelma, jonka avulla robottiauto kykenee liikkumaan itsenäisesti. Toisessa harjoituksessa käsiteltiin C++-ohjelmointikieltä ja hyväksikäytettiin kehittäjän luomaa ohjelmaa. Työssä ohjelmoitiin robottiauto seuraamaan sille merkittyä polkua.

Avainsanat	Arduino, Mobiilirobotti, Robotiikka

Author Title Number of Pages Date	Jyri Pietikäinen Utilization of Mobile Robotics in Machine Automation Studies 62 pages including 20 pages of appendices. 14 April 2021
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Professional Major	Machine Automation
Instructors	Antti Liljaniemi, Senior Lecturer

This thesis examines mobile robotics and its operating principles on a general level. The theoretical part of the thesis deals with mobile robotics. In addition, this thesis discusses the use of robotics in the studies of machine automation at Metropolia University of Applied Sciences.

The thesis introduced DOBOT's robotics solutions for teaching, which were used to create teaching material for the needs of machine automation studies at Metropolia. A robot car equipped with a rich variety of programmable components is used as a teaching tool for programming and electronics. In the thesis, the operating principles of the components were studied and described in more detail. After studying the theory of the components, an assembly was performed for the robot car. The process of the assembly is described in the thesis.

The robotics used in the studies were programmed with two different methods for learning purposes. The first method uses graphical programming, and the code is made with DOBOT's own software called DobotBlock. The second method utilizes the C++-programming language coded with Arduino IDE-software. Based on these programs, two different exercises were created with laboratory instructions.

The first laboratory exercise consists of programming the robot car so that it can navigate autonomously with the help of its ultrasonic sensors. In the second exercise, the robot car is programmed to follow a marked path with its infrared sensors.

Keywords	Arduino, Mobile Robot, Robotics

Sisällysluettelo

Lyhenteet

1 Johdanto 1

2 Teoria 2

2.1 Yleistä 2
UGV 4
2.2 Mobiilirobotti 5
2.2.1 Paikantaminen 6
2.2.2 Navigointi 8
2.2.3 Sense – Think – Act 14

3 VRBOT – AI starter 15

3.1 DOBOT 15
3.2 AI-Starter 16
3.3 Osat 17
3.3.1 Moottori 17
3.3.2 Sensorit 18
3.3.3 Mikrokontrolleri 20
3.4 Kokoonpano 21
3.5 Ohjelmat 23
3.5.1 DobotBlock 23
3.5.2 Arduino IDE 25

4 Laboratoriotyöt 27

4.1 Laboratoriotyöt 27
4.2 Työ 1 27
4.3 Työ 2 30

5 Yhteenveto 38

Lähteet 40

Liitteet

Liite 1. Harjoitukset

Liite 2. Harjoitusten ratkaisut

Lyhenteet

UGV	Unmanned ground vehicle, miehittämätön maa-ajoneuvo
AGV	Automated Guided Vehicle, vihivaunu
AMR	Autonomous Mobile Robot, autonominen mobiilirobotti
CPU	Central Processing Unit, prosessori
GPS	Global Positioning System, globaali satelliittipaikannusjärjestelmä
IDE	Integrated Development Environment, Integroitu ohjelmointiympäristö
LMR	Legged Mobile Robot, jalallinen mobiilirobotti
PRM	Probabilistic roadmaps, tiekartta
RAM	Random Access Memory, Hajasaantimuisti eli luku-/kirjoitusmuisti
ROM	Read Only Memory, lukumuisti
RPM	Revolutions per minute, kierrosta minuutissa, käyntinopeuden kuvaaminen
RRT	Rapidly exploring random trees, puukaavio
SLAM	Simultaneous Localization and Mapping, SLAM-tekniikka
UAV	Unmanned Aerial Vehicle, miehittämätön ilma-alus
UMV	Unmanned Vehicle, miehittämätön alus
USV	Unmanned Surface Vehicle, miehittämätön vedenpäällinen alus
UUV	Unmanned Underwater Vehicle, miehittämätön vedenalainen alus

1 Johdanto

Mobiilirobotiikan käyttö on kasvanut viime vuosina eri teollisuuden toimialoilla tekniikan jatkuvan kehityksen myötä. Tämän insinöörityön tarkoituksena on tutustua miehittämättömään teknologiaan, ja perehtyä syvemmin autonomisen mobiilirobotiikan toimintaperiaatteisiin.

Opinnäytetyössä perehdytään myös yleisen teoriaosion lisäksi mobiilirobotiikan toimintaedellytyksiin käytännön harjoittelun kautta, ja luodaan koneautomaation opintoihin soveltuvaa opetusmateriaalia. Kahta suoritettavaa tehtävää varten hyödynnetään DOBOT-opetusrobotiikkaa, jollaista Metropolian ammattikorkeakoulu on hankkinut oppimistyökaluksi opiskelijoilleen. Opetusmateriaaleissa on tarkoitus myös soveltaa opinnoista jo olemassa olevaa osaamista.

Suoritettavia tehtäviä varten opiskellaan ensin teoriaosio. Teoriaosion opiskelun jälkeen voidaan suorittaa käyttöönotto opetusrobotiikkalaitteelle, samalla kuvaten tämä prosessi. Käyttöönoton jälkeen suunnitellaan DOBOT-robottiautolle sellaisia toimintaohjelmia, joissa hyödynnettäisiin mahdollisimman kokonaisvaltaisesti siinä olevaa teknologiaa. Ohjelmista luodaan yksinkertaisia, jotta ne soveltuisivat hyvin myös opiskelijoille, joilla ei välttämättä löydy vielä aiempaa kokemusta ohjelmoinnista. Opinnäytetyössä kuvataan myös, mitä ohjelmat tekevät. Luoduista ohjelmista laaditaan myös laboratoriotyöskentelyä varten toimintaohjeet, jotta opiskelijat voisivat hyödyntää niitä koneautomaation opinnoissa.

Tehtäviä on kaksi, ja ne luodaan kahdella eri ohjelmistoympäristöllä. Ohjelmissa käytetään myös kahta erilaista ohjelmointimetodia. Ensimmäisessä työssä tehdään ohjelma DOBOTin kehittämässä ohjelmointiympäristössä, jossa käytetään graafista ohjelmointikieltä. Robottiauto ohjelmoidaan niin, että se kykenee liikkumaan itsenäisesti. Toisessa työssä käsitellään ohjelmaa, jonka avulla robottiauto kykenee seuraamaan sille annettua reittiä, hyödyntäen sen sensoreita. Tämä toinen ohjelma luodaan Arduino IDE-toimintaympäristössä käyttäen C++-ohjelmointikieltä.

2 Teoria

2.1 Yleistä

UMV tarkoittaa täysin miehittämätöntä, elektromekaanista laitetta. Elektromekaaninen laite kykenee suorittamaan sille määrättyjä tehtäviä joko etäältä ohjattuna, tai sille valmiiksi ohjelmoituja reittejä pitkin. Laite voidaan ohjelmoida toimimaan myös täysin itsenäisesti, eli autonomisesti. Autonominen laite lukee sekä havainnoi lähistöään jatkuvasti, ja on näin vuorovaikutuksessa sitä ympäröivän tilan kanssa. Kyetäkseen autonomiseen toimintaan, laitteessa on oltava sisäänrakennettu tietokone. Miehittämättömiä aluksia hyödynnetään eri toimintaympäristöissä. Ne voivat toimia niin maalla, merellä, ilmassa kuin avaruudessakin. (Huang 2004: 20.), (Gage 1995: 3.)

Miehittämätöntä teknologiaa voidaan hyödyntää myös sellaisissa tilanteissa, jotka ovat esimerkiksi ihmiselle vaarallisia, työläitä, vaativia tai vaikeasti saavutettavissa paikoissa. Miehittämätöntä teknologiaa käytetään monilla erilaisilla toimialoilla, kuten esimerkiksi teollisuudessa, terveydenhuollossa, maataloudessa, logistiikassa, ajoneuvoteollisuudessa, avaruustutkimuksissa, kotitalouksissa ja maanpuolustuksessa. (Siegwart & Roland 2004:1–10).

Kuvasarja 1**.** Miehittämättömiä aluksia vedenpinnan yläpuolella (USV), ja vedenpinnan alapuolella (UUV), suorittamassa sotilaallisia tehtäviä ja tutkimassa merta.

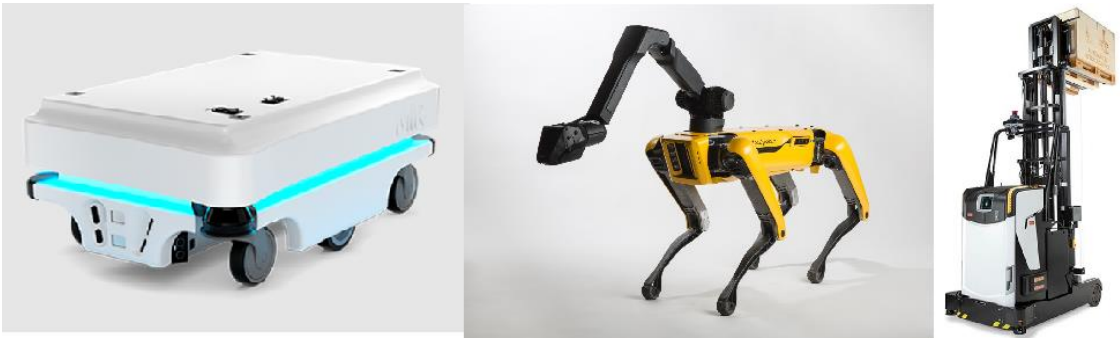
Kuvasarja 2. Miehittämättömiä ilma-aluksia (UAV), kutsutaan myös nimellä *Drone*. Autonominen kopteri tarkoitettu kasvien lannoitukseen, lennokki puolestaan sotilaallisiin tehtäviin.

UGV

UGV tarkoittaa miehittämätöntä, mekaanista maa-ajoneuvoa, joka liikkuu maanpinnan yläpuolella. Miehittämättömiä maa-ajoneuvoja on kahdenlaisia, jotka jaotellaan eri alaluokkiin. Näitä luokkia ovat AGV, johon kuuluvat erilaiset automaattisesti ohjautuvat ajoneuvot, sekä AMR, jolla tarkoitetaan autonomisia mobiilirobotteja. AGV- laitteita hyödynnetään tavallisimmin tehdasteollisuudessa sekä varastologistissa tehtävissä. Arkikielessä AGV-laitteistot tunnetaan paremmin itseohjautuvina kuljetusvaunuina tai vihivaunuina.

UGV-laitteiden erilaisista käyttötarkoituksista riippuen niiden fyysiset sekä rakenteelliset ominaisuudet voivat vaihdella. Laitteen liikkumistapa määritetään myös käyttöympäristön kannalta sille soveltuvimmaksi; esimerkiksi tasaisella pinnalla laite voi edetä renkaiden avulla, kun taas epätasaisessa maastossa liikkuessaan saattaa se tarvita mekaani-

set jalat edetäkseen. UGV- laitteiden yleisimmät rakenteelliset osat ovat runko, manipulaattorit, sensorit, ohjausyksikkö sekä liikkumistavan määrittävä mekaniikka. (Siegwart & Roland 2004:13–15)



Kuvasarja 3. *MIR100*-mobiilirobotti, jota hyödynnetään sisälogistisissa tehtävissä (AMR). Keskellä *BostonDynamics Spot*, jalallinen mobiilirobotti (LMR), joka kykenee kulkemaan epätasaisessakin maastossa. Oikealla *Rocla*n automaattitrukki, joka on varusteltu nostohaarukalla (AGV).

AGV- ja AMR- laitteiden eroavaisuudet ovat niiden autonomisuudessa, sekä älyllisissä ominaisuuksissa. AGV-laitteet ovat yhteydessä erilliseen automaattiseen järjestelmään, joka ohjaa niiden toimintaa. AGV-laitteiden toiminta perustuu siis ennalta rakennettuun infrastruktuuriin, sillä ne eivät kykene luomaan reittiään AMR-laitteiden tavoin itse, vaan ne tarvitsevat liikkuakseen yksinkertaisemmat navigointiohjeet. AGV-laitteiden tyypillisimmät ohjaustekniikat ovat niiden liikkumispinnalle asetetut magneettinauhat, joita laite seuraa sensoriensa avulla, sekä lasernavigointi, jossa laite suunnistaa eteenpäin laseria heijastavien peilien avulla. AGV:t eivät omaa AMR-laitteiden tavoin älykästä autonomista tekniikkaa, ja tästä syystä myös AGV-laitteen havaitessa sille määrätyllä reitillä jonkin esteen, se ei kykene navigoimaan vaihtoehtoista reittiä itselleen, vaan pysähtyy, kunnes este on poistettu.

2.2 Mobiilirobotti

Robotti tarkoittaa uudelleenohjelmoitavissa olevaa, monikäyttöistä, vähintään kolme niveltä omaavaa mekaanista laitetta. Robotit sisältävät runsaasti tietotekniikkaa, antureita,

yhden tai useamman moottorin, sekä muuta mekaniikkaa. Robotti voi liikuttaa sen ohjainyksikköön syötetyn ohjelman avulla kappaleita, manipulaattoreita tai erikoislaitteita. Robotti kykenee liikkumaan sille luodun ohjelman sekä sisäänrakennettujen sensoreidensa avulla käyttäen samalla ympäristöstään havainnoimia tietoja.

AMR–robotit ovat ominaisuuksiltaan sellaisia, jotka pystyvät liikkumaan itsenäisesti myös ennalta tuntemattomassa sekä muuttuvassa ympäristössä (Keinänen & Sumujärvi 2019: 298). Autonominen mobiilirobotti voi siis toimia monissa erilaisissa toimintaympäristöissä, kuten merellä, ilmassa tai maalla.

Mobiilirobotilla on oltava kyky havainnoida omaa ympäristöään, jotta se kykenisi keräämään ympäriltään tarvitsemansa datan oman sijaintinsa paikantamiseen. Sen tulee myös määrittää optimaalisin liikkumistapa haluttuun määränpäähän. Halutun määränpään robotti saavuttaa prosessoimalla annetun tehtävän tietoja luoden tästä reittisuunnitelman kognitionsa avulla. Reittisuunnitelma syötetään mobiilirobotin liikeohjaimelle, joka ohjaa sen toimilaitteita siten, että mobiilirobotti seuraa suunniteltua reittiä.

2.2.1 Paikantaminen

Ennalta annetulta kartalta sijaintinsa määrittämiseksi mobiilirobotti tarvitsee sensoreita, joilla sen havainnointi tapahtuu. Sensori on mittalaite, joka tunnistaa tai mittaa jonkin kemiallisen tai fysikaalisen ilmiön, ja muuntaa saadun tiedon tarvittavaan muotoon (Keinänen 2019: 206). Sensorit voidaan jakaa kahteen ryhmään; sisäisiin, sekä ulkoisiin. Sisäiset sensorit mittaavat laitteen sisäisiä ilmiöitä, kuten esimerkiksi lämpötilaa, moottorin pyörimisnopeutta tai nivelen kulmaa. Ulkoiset sensorit keräävät tietoa laitteen ympäristöstä; esimerkiksi etäisyyksiä, värejä tai ääniä. (Siegwart & Roland 2004: 89–90.)

Mobiilirobotti voi paikantaa sijaintinsa kahdella eri keinolla; suhteellisella, ja absoluuttisella paikanmittauskeinolla. Mobiiliroboteissa käytetään yleensä molempia paikantamiskeinoja samanaikaisesti, jotta sen sijaintitiedoista saavutettaisiin mahdollisimman tarkka.

Suhteellisessa mittauksessa mobiilirobotin nykyisen sijainnin data verrataan keskenään aiemmin saadun sijaintitiedon kanssa. Mittaukset lasketaan yhteen, ja näin saadaan arvio mobiilirobotin jo liikkumasta matkasta. Kokonaisarvio sen kulkemasta matkasta saadaan, kun laskelmia verrataan sen liikkeellelähtöpisteeseen. Paikallisista sijaintiarvoista kerätty data ei välttämättä ole tarpeeksi tarkkaa, joten tässä mittausmetodissa mahdolliset virheet voivat kerääntyä. Tämä aikaansaa sen, että absoluuttinen virhe kokonaismatkan arvioinnista kasvaa kumulatiivisesti kuljetun matkan myötä. (Tzafestas, 2014.)

Suhteellinen paikannus koostuu:

- Odometriasta:
  Odometrian avulla mobiilirobotti kykenee laskemaan kulkemaansa matkaa, jonka se mittaa liikesensoreidensa avulla. Liikesensoreilla mitataan mobiilirobotin akselin liikettä. Robotin edetessä, akselin samalla pyörien, voidaan akselin kiertymän muutoksesta laskea matemaattisen yhtälön avulla liikuttu matka. Odometrisen mittauksen tarkkuus voi kärsiä tai altistaa virheille laitteen renkaiden pyöriessä, mutta laitteen pysyessä kuitenkin paikoillaan. Näin voi tapahtua liukkaalla pinnalla tai epätasaisella alustalla.

- Inertiasta:
  Inertia kertoo, millä nopeudella kappaleet reagoivat liikkeen muutokseen. Sen avulla voidaan siis arvioida mobiilirobotin sijaintia mittaamalla sen kiihtyvyyttä ja suunnanmuutoksia. Mittaaminen tapahtuu IMU-laitteen avulla. Laite sisältää erilaisia kiihtyvyysantureita, sekä gyroskooppeja. Suhteellisen mittausmetodin mukaisesti myös inertiallisten mittaustulosten virheet kasvavat kumulatiivisesti sen mukaan, mitä pidempiä matkoja kuljetaan. Absoluuttisessa mittauksessa saatuja tietoja verrataan ennalta annettuun referenssikarttaan tai koordinaatistoon.

Absoluuttinen paikannus koostuu:

- Aktiivisista lähettimistä:
  Mobiilirobotti voi paikantaa sijaintinsa aktiivisten lähettimien avulla. Absoluutti-

nen sijainti saadaan etäisyyksiä laskemalla kolmea tai useampaa lähetintä hyödyntäen. Lähettimet ovat asennettu valmiiksi ennalta määrätyille sijainneille. Lähettimet käyttävät valo- tai radioaaltoja datan välittämiseen. Tällaista menetelmää hyödynnetään esimerkiksi satelliitti- ja ultraäänipaikannuksissa.

- Maamerkkien tunnistuksesta:
  Mobiilirobotti voi paikantaa sijaintinsa myös tunnistamalla luonnollisia, tai keinotekoisia maamerkkejä. Niiden tunnistamiseksi se voi käyttää joko konenäköä, tai infrapuna-, laser- ja ultraäänisensoreita. Luonnollisia maamerkkejä voivat olla esimerkiksi konenäöllä havaittavat pinnan muodot, kuten puut tai esineet. Keinotekoisia maamerkkejä taas voivat olla esimerkiksi hyllykköihin asennetut peilit, jotka heijastavat laserin takaisin mobiilirobotin havaittavaksi.

- Malleihin perustuvasta paikannuksesta:
  Mobiilirobotti kerää sensoreillaan vallitsevasta ympäristöstä tietoja, joita se vertailee ennalta annettuun referenssikarttaan. Jos sensoreista saatu tieto täsmää kartan tietoihin, tietää mobiilirobotti absoluuttisen sijaintinsa.

Mikäli mobiilirobotilla ei ole ennakkoon annettua karttaa ympäristöstään, voi se luoda sellaisen itse. SLAM–metodissa mobiilirobotti liikkuu sille osoitetussa ympäristössä luoden siitä samanaikaisesti karttaa, sekä odometrian avulla paikantaen samalla itseään tästä kartasta. Datan keräämiseksi mobiilirobotti käyttää laser- tai ultraäänisensoreita, tai konenäköä.

Kun haluttu toimintaympäristö on skannattu, eritellään kartasta staattiset sekä dynaamiset kappaleet. Kohdatessaan tiellään jonkin esteen, kykenee se edellä mainittujen toimenpiteiden avulla väistämään tämän. Valmiiksi luotua karttaa voidaan hyödyntää myös muissa toimintaympäristöön liittyvissä tehtävissä. (Siegwart & Roland 2004: 248–252.)

2.2.2 Navigointi

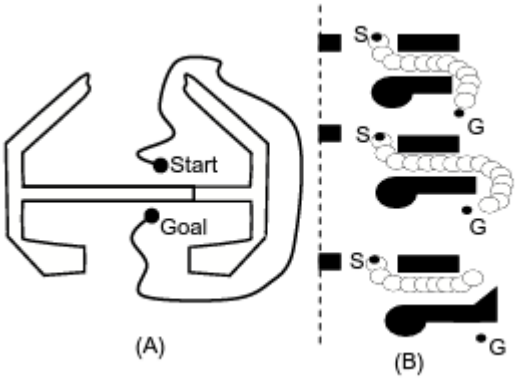
Mobiilirobotin navigointi muodostuu sen kognitiivisesta kyvystä prosessoida informaatiota, ja kyvystä ohjata autonomisesti sille määriteltyjä toimintoja. Mobiilirobottiin ohjelmoitu kognitio, eli tekoäly, navigoi sille optimaalisinta kulkureittiä. Samalla se analysoi

sensoreistaan sekä muista mekaanisista osista tulevaa informaatiota, muuntaen datan oikeaan muotoon robotin ohjausjärjestelmälle.

Tekoälyä hyödyntävä mobiilirobotti pyrkii suunnittelemaan siis itselleen optimaalisen kulkureitin aina sen lähtöpisteestä sille asetettuun päämäärään saakka. Törmäysvapaan reitin lisäksi se huomioi samalla myös sille mahdollisesti asetetut muut tehtävät tai tavoitteet, joiden tarkoitus on rajata tuloksia. Tällaisia tavoitteita voivat olla esimerkiksi nopein matka määränpäähän, tai esteiden kiertäminen tiettyä etäisyyttä noudattaen. Mobiilirobotti suorittaa siis jatkuvaa ongelmanratkaisua. Ongelmanratkaisu on pohjimmiltaan hakuohjelma, jossa ratkaistava ongelma määritellään tietyn alkutilan ja halutun tavoitteen perusteella. Ohjelma ohjaa hakua arvioimalla nykyistä, ja haluttua tilaa käyttäen vain sellaisia operaattoreita, jotka lupaavat parhaimman mahdollisen tuloksen. Tieto toimintaympäristöstä vaikuttaa suuresti mobiilirobotin kulkureitin suunnitteluun. (Tzafestas, 2014.)

2.2.2.1 Reitin suunnittelu

Mobiilirobotin toimintaympäristö voi olla kokonaan tiedetty, osittain tiedetty tai täysin tuntematon. Yleisimmin toimintaympäristö on osittain tiedetty. Mobiilirobotin reitti koostuu sarjasta pisteitä, jossa kukin piste sisältää informaatiota mobiilirobotin orientaatiosta, sijainnista tai suunnasta. Reitin suunnittelu voi olla joko globaalia tai paikallista. Paikallinen reittisuunnittelu tapahtuu reaaliajassa, mobiilirobotin ollessa liikkeellä. Näin ollen se kykenee suunnittelemaan uuden reitin tuntemattoman esteen sattuessa sen kulkureitille. Globaali reittisuunnittelu taas toimii vain staattisessa ympäristössä, ja reittisuunnittelu tapahtuu tällöin ennen mobiilirobotin liikkeellelähtöä. (Tzafestas, 2014.)



Kuva 4. Vasemmalla esimerkkikuvaus globaalista reitistä, sekä oikealla kuvaus paikallisesta reitistä.

2.2.2.2 Konfiguraatioavaruus

Konfiguraatioavaruudella tarkoitetaan mobiilirobotille ohjelmoitua, sille määrättyä toimintaympäristöä, jonka sisällä se voi liikkua vapaasti. Konfiguraatioavaruus käsittää kappaleen kaikki mahdolliset konfiguraatiot, eli positiot ja asemat. Lisäksi se määrittää käsiteltävän alueen esteettömän sekä esteellisen tilan.

Reittien suunnittelutyöhön on monia eri keinoja riippuen ympäristön tuntemuksesta, esteiden määrästä sekä vapausasteista.

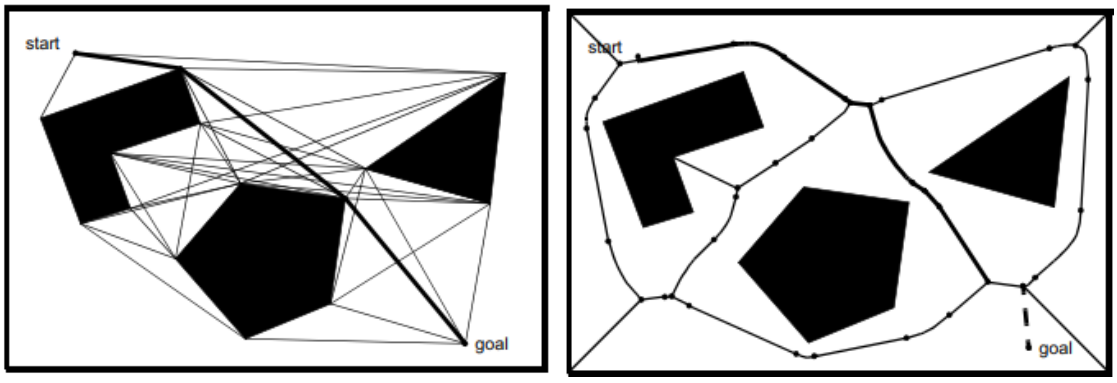
Erilaisia reittisuunnittelumenetelmiä:

1. Tiekarttamenetelmässä vapaa tila yhdistetään viivoilla konfiguraatioavaruudessa olevia esteitä hyödyntäen. Vapaa tila yhdistetään siis niin, että jokainen positio ja asema tilan sisällä voidaan saavuttaa.

- ”Visibility graph” -menetelmässä vapaa tila yhdistetään viivoin käyttäen hyväksi esteiden kulmia. Jokaisesta kulmasta muodostetaan viiva seuraaviin näkyvissä oleviin kulmiin. Mikäli esteet ovat pyöreitä, täytyy niistä muodostaa kulmikkaita. Kun koko tila on kartoitettu, asetetaan reitin aloitus- sekä lopetuspiste karttaan.

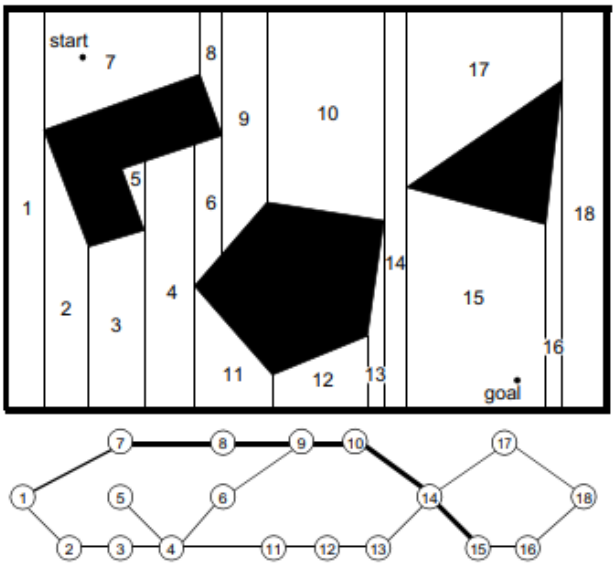
Yhdistetyistä viivoista muodostuu lopulta kaikki mahdolliset kulkureitit mobiilirobotille. (Siegwart 2004: 262.)

- Voronoin diagrammissa kaikki tilassa esiintyvät esteet pyritään kiertämään mahdollisimman kaukaa. Jokaisesta esteestä siis merkataan piste tietylle etäisyydelle, joka sijaitsee mahdollisimman kaukana varatusta tilasta. Näiden merkintöjen avulla muodostetaan kulkureitit mobiilirobotille. (Siegwart 2004: 263.)



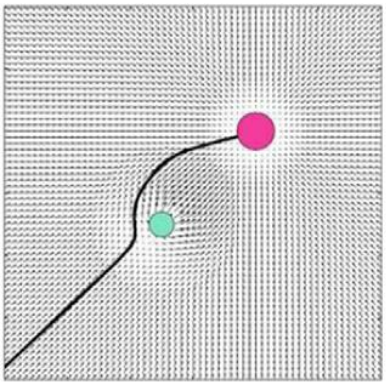
Kuva 5. Erilaiset tiekarttamenetelmät. Vasemmalla visibility graph, ja oikealla Voronoin diagrammi.

2. Soluihin jakamisen menetelmässä määritelty vapaa, esteetön tila jaotellaan eri soluihin. Kun vapaat solut on määritelty, selvitetään seuraavaksi vierekkäiset solut. Vierekkäisistä soluista voidaan muodostaa reitin yhteneväisyyskuvaaja. Kun reitin aloituspiste sekä päämäärä ovat tiedossa, voidaan mobiilirobotille muodostaa kulkureitti. Mobiilirobotti kulkee vapaiden solujen läpi niiden keskipisteiden kautta. (Siegwart 2004: 264.)



Kuva 6. Soluihin jako. Ylemmässä kuvassa vapaa tila on jaoteltu soluihin. Alemassa kuvassa on kuvattuna optimaalisin reitti maaliin.

3. Keinotekoisessa potentiaalikenttämenetelmässä tilassa sijaitseva haluttu määränpää, eli maali, on merkitty puoleensa vetäväksi kiintopisteeksi. Tilassa olevat esteet ovat puolestaan merkitty hylkiviksi kappaleiksi. Mobiilirobotti liikkuu kohti puoleensa vetävää määränpäätä samalla, kun se pyrkii poispäin hylkivistä esteistä.



Kuva 7. Keinotekoinen potentiaalikenttämenetelmä. Punainen pallo esittää kuvaajassa määränpäätä, jolla on puoleensa vetävä voima. Vihreä pallo kuvastaa estettä, jolla on hylkivä voima.

4. Otantamenetelmässä reittisuunnittelu ei vaadi ennakkotietoa ympäristöstä eli konfiguraatioavaruuden vapaasta tilasta, toisin kuin monet muut reitin suunnittelumenetelmät. Otantamenetelmässä tilasta otetaan sattumanvaraisesti näytteitä. Näytteidenotolla tarkoitetaan törmäysvaaran tarkistamista, ja optimaalisten pisteiden etsintää. Mikäli törmäysvaaraa ei ole, kuuluu piste vapaaseen tilaan. Näytteitä ottamalla saadaan lopulta joukko pisteitä, jotka kuuluvat vapaaseen tilaan, ja näistä saadaan muodostettua laitteelle sopiva kulkureitti. Otantamenetelmän pisteiden sattumanvaraiseen tarkistamiseen löytyy kaksi erilaista algoritmia; Rapidly exploring random trees (RRT), ja Probabilistic roadmaps (RPM). Kummankin menetelmän tarkkuuteen vaikuttaa pisteiden välinen etäisyys. (Klancar 2017: 162.)

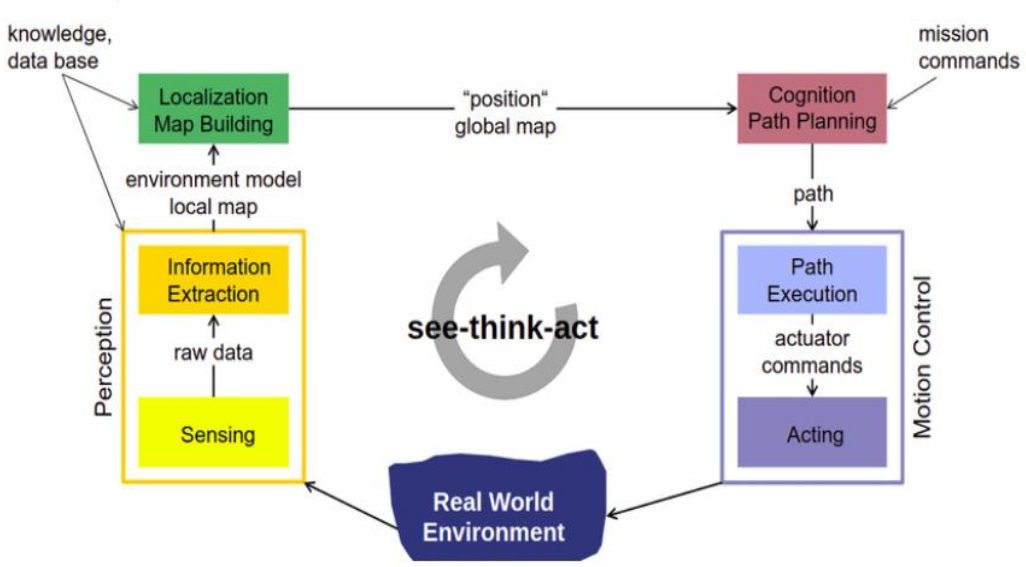
- RRT–menetelmässä mobiilirobotti luo oksamaista reittiä sille annetusta lähtöpisteestä pyrkien kohti määränpäätä. Se lisää reittiä lähimpänä olevan pisteen ennalta määritellyn etäisyyden mukaisesti. Kun määritelty määrä iteraatioita on tehty, katsotaan, onko loppupiste mahdollista yhdistää niin kutsuttuun puukaavioon. Mikäli se ei ole mahdollista, jatketaan iteraatioita. Lopulta koko konfiguraatioavaruuden kaikki pisteet on saavutettu.

- RPM–menetelmässä muodostetaan tiekartta vapaasta tilasta. Menetelmässä valitaan pisteitä sattumanvaraiseen suuntaan ennalta määritellyn etäisyyden mukaan. Kun vapaa tila on täysin skannattu, muodostetaan reittien yhtymäkohdat. Lopuksi tiekarttaan sijoitetaan aloitus- ja lopetuspiste. (Klancar 2017: 163).

Liikkeen suunnittelu on prosessi, jossa valikoidaan mobiilirobotille optimaalinen liiketoiminto sekä sitä vastaava syöte niin, että robotit noudattavat samalla niille asetettuja ja määriteltyjä rajoitteita. Liikkeen suunnittelua voidaan pitää joukkona laskelmia, jotka tarjoavat pienempiä alatavoitteita tai asetettuja pisteitä mobiilirobotin ohjaamiseksi. (Tzafestas, 2014).

2.2.3 Sense – Think – Act

Mobiilirobotin toimintaa voidaan kuvata kaaviolla, jossa jokainen toimi vaikuttaa seuraavaan.



Kuva 8. Sense – Think – Act -kaavio

Kaavio koostuu neljästä haarasta, joita ovat:

1. Havainnointi
   Mobiilirobotti havainnoi ympäristöään sensoriensa avulla keräten siitä dataa. Datasta se erittelee tarvitsemansa informaation; Esimerkiksi paikantamiseen se voi hyödyntää konenäköä tunnistaakseen erilaisia ominaisuuksia, liikkuessaan se voi mitata kulkemaansa matkaa enkooderilla, ultraäänisensorinsa avulla se kykenee puolestaan väistelemään esteitä.

2. Paikannus
   Sijaintinsa paikantamiseen ja kartan muodostamiseen mobiilirobotti käyttää ennalta annettua tietoa, sekä sensoreilla havaitsemaansa dataa. Kun tieto kartasta on koottu, kykenee mobiilirobotti paikantamaan sijaintinsa.

3. Kognitio
   Tekoäly luo mobiilirobotille kulkureitin ja suunnitelman halutun tahtotilan ja tehtävänannon saavuttamiseksi. Tehtävänannosta riippuen mobiilirobotille asetetaan rajoitteita, joita noudattaen se suunnittelee reaaliaikaisesti liikkumistansa reitillä.

4. Liikkeenohjaus
   Kulkureitin suunnittelun jälkeen mobiilirobotin ohjausyksikkö ohjaa käskyjä esimerkiksi mobiilirobotin moottorille liikuttaakseen sen renkaita, jotta se kykenisi liikkumaan optimilla tavalla toimintaympäristössään. Mobiilirobotin liikkeitä on rajattu käytöstapaisilla algoritmeilla, jotka määrittävät myös sen, kuinka mobiilirobotin tulisi liikkua. Tällaisia algoritmeja voivat olla esimerkiksi nopeuden säätäminen tai esteiden väisteleminen.

3 VRBOT – AI starter

3.1 DOBOT

Shenzhen Yuejiang Technology Co., eli DOBOT, on vuonna 2015 Kiinassa perustettu organisaatio. DOBOT tuottaa erilaisia helppokäyttöisiä ja moderneja robotiikan sovelluksia opetuskäyttöön, teollisuuteen ja kaupan alalle. Laitteet ovat suunnattuja niin yksityis- kuin yritysasiakkaillekin. Organisaation pääsegmentit ovat niin kutsuttuja yhteistyörobotteja, eli sellaisia robotteja, jotka ovat tarkoitettuja operoimaan ihmisten kanssa samassa tilassa. Yhteistyörobottien lisäksi DOBOT tarjoaa myös 3D–tulostimia, sekä erilaisia oheislaitteita opetuskäyttöä varten. Opetusrobotiikkaa varten luodut laitteet ovat

suunniteltu monikäyttöisiksi sekä helposti muokkailtaviksi.

Kuva 9. DOBOT-yhteistyörobotti, sekä opetuskäyttöön tarkoitettu linjasto.

DOBOT tarjoaa tuotteilleen *DobotStudio*- ja *DobotBlock*- ohjelmointiympäristöt, joita voidaan hyödyntää niin opetuksessa, kuin teollisuudessakin.

DobotStudio on ohjelma, joka mahdollistaa robotiikan ohjelmoinnin monilla eri ohjelmointikielillä. Tämän lisäksi ohjelma tarjoaa mahdollisuuksia esimerkiksi töiden simulointiin sekä mallinnukseen. Ohjelmasta löytyy myös laitteita varten luotuja esimerkkiohjelmia sekä opetusmateriaalia.

3.2 AI-Starter

AI-Starter on DOBOTin opetuskäyttöön suunnattu robottiauto, joka on ohjelmoitavissa useisiin eri tehtäviin. AI-starter sisältää robottiauton, sen kokoamista varten tarvittavat ohjeet, työkalut sekä alustan, johon on valmiiksi printattu kulkureitti laitetta varten. Robottia maahantuo *RobotLine*-niminen organisaatio.

3.3 Osat

AI-Starter-robottiauto koostuu kahdesta metallisesta runko-osasta, ja niiden sisällä olevista komponenteista. Robottiauto liikkuu kahden renkaan avulla, jotka sijaitsevat sen takaosassa. Robotin etuosaan kiinnitetään myös universaali rengas, eli rengas, joka liikkuu kaikkiin suuntiin. Robotti kääntyy, mikäli sen takarenkaita ohjataan pyörimään eri nopeuksilla. Robottiauton virtalähteenä toimii kaksi uudelleenladattavaa 18650- koon 3.7V, ja 2400mAh- litiumparistoa. Alkuperäisestä pakkauksesta nämä puuttuivat, joten ne täytyvät tilata erikseen.

3.3.1 Moottori

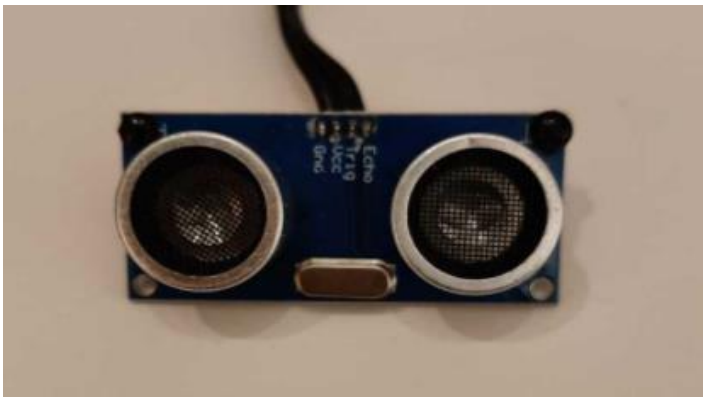
Robottiautossa käytetään kahta 7V-vaihteellista tasavirtamoottoria, jotka on varustettu enkoodereilla. Moottoreilla on 48:1 vaihdesuhde. Niiden suurimmaksi kierrosnopeudeksi on annettu 100 rpm. Moottoreissa on pidennetty taka-akseli, johon on kiinnitetty magneettinen enkooderi. Enkooderilla voidaan seurata moottorin akselin kiertymää ja muuttaa saatu data digitaaliseksi tiedoksi. Tästä datasta voidaan määrittää moottorin pyörimisnopeus ja suunta.

Kuva 10. Kuva laitteen moottorista.

3.3.2 Sensorit

1. Ultraääni

   Ultraäänisensori toimii samanaikaisesti lähettimenä ja vastaanottimena. Sensori lähettää korkeataajuista ääntä, joka heijastuu takaisin osuessaan fyysiseen esteeseen. Signaalin palatessa takasin sensoriin, voidaan laskea signaalin matkaan kulunut aika, josta saadaan laitteen etäisyys esteeseen. Robottiauto on varustettu kolmella ultraäänisensorilla, jotka mahdollistavat 120 asteen laajuisen alueen havainnoinnin. Ultraäänisensorien mittausetäisyys on säädeltävissä 3–500 millimetrin etäisyyksillä.

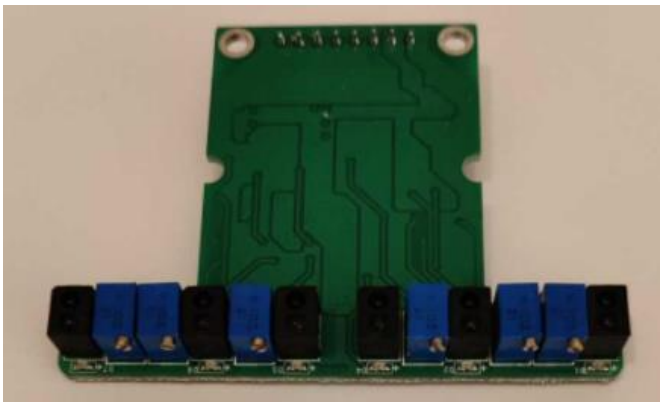


Kuva 11. Kuva laitteen ultraäänisensorista.

2. Infrapuna

   Robottiautossa on kuusi säädettävää infrapunasensoria, jotka sijaitsevat robotin pohjassa. Infrapunasensori lähettää infrapunavaloa, joka heijastuu laitteeseen takaisin. Havaittavan kohteen ominaisuudet vaikuttavat valon heijastumaan,

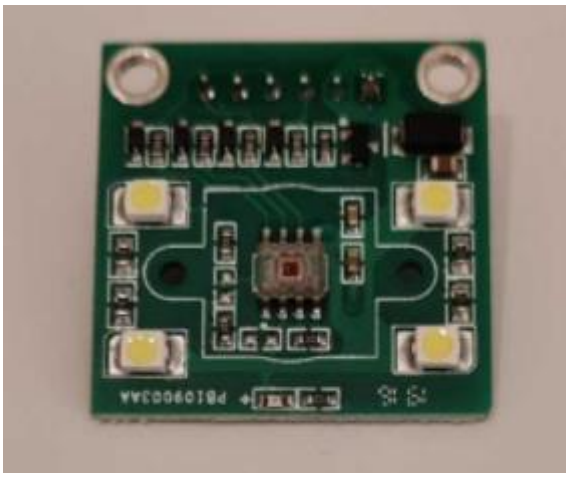
sillä erilaiset pinnat sekä värit heijastavat valoa eri tavoin. Infrapunasensorien mittausetäisyyttä voidaan säätää 1–30 millimetrin välillä.



Kuva 12. Kuvassa laitteen infrapunasensorit.

3. Väri

   Robottiauton pohjassa on kaksi värisensoria, sekä led-valot, joiden tarkoitus on valaista havaittavaa kohdetta. Värisensorien toimintaperiaate perustuu punaisen, sinisen sekä vihreän valon lähettämisestä tarkistettavalle pinnalle. Sensori mittaa takaisin heijastuneen säteilyn väriarvojen vaihtelun ja vertaa näitä referenssiarvoihin. Vertailun avulla saadaan selville tarkistetun kohteen väri.



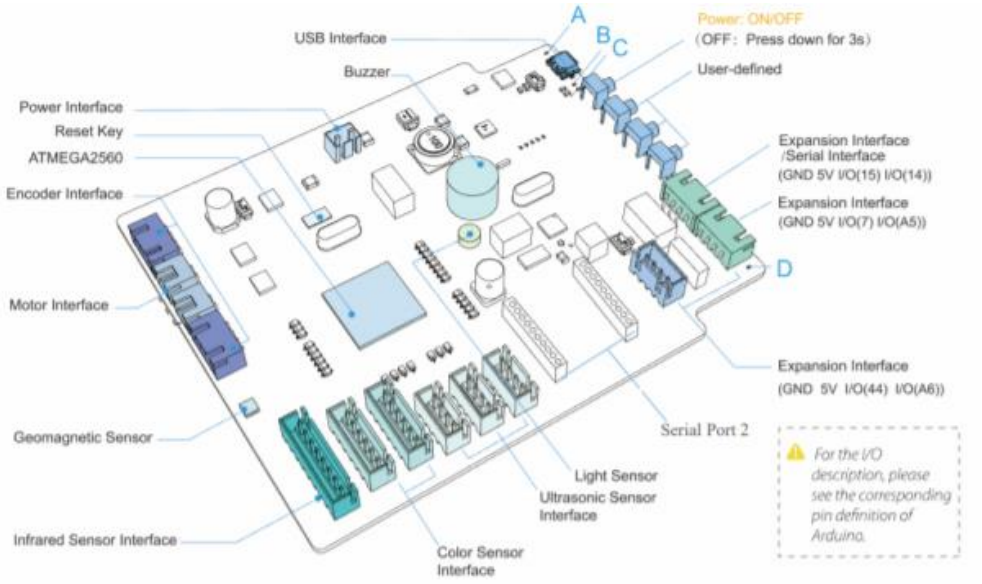
Kuva 13. Komponentti, jossa laitteen värisensorit sijaitsevat.

3.3.3 Mikrokontrolleri

Mikrokontrolleri on mikropiiri, joka koostuu mikroprosessorista, muisti- ja liitäntälohkoista sekä mahdollisista oheiskomponenteista. Mikroprosessori eli suoritin (CPU), on mikrokontrollerin komentokeskus. Mikroprosessorin tehtävä on vastaanottaa tietoa, prosessoida ja muuntaa sitä, ja lähettää edelleen muualle laitteistoon todetut toimenpiteet.

Mikrokontrollerin muisti koostuu kahdenlaisesta muistista: ROM-muistista eli lukumuistista, johon on tarkoitus ladata tietokoneella kirjoitettu ohjelma, sekä RAM-muistista, eli työmuistista, jota voidaan kirjoittaa sekä lukea. RAM-muisti tyhjenee aina laitteen virran katketessa, kun taas ROM-muisti on laitteella pysyvä muisti. Liitäntälohkoja eli sarjaportteja ja I/O-liittimiä käytetään tiedonsiirtoon ja ulkopuolisten laitteiden hallinnoimiseen. Liitäntälohkot voidaan ohjelmoida suorittamaan joitakin tiettyjä toimintoja.

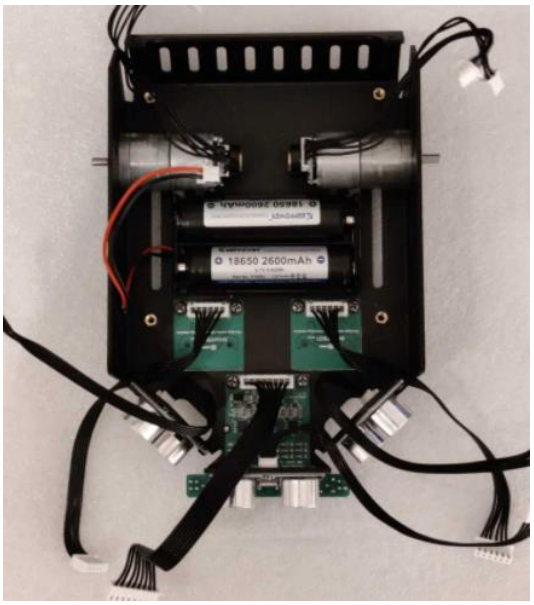
Arduino Mega2560 on avoimen lähdekoodin elektroniikka-alusta, joka perustuu helppokäyttöiseen laitteistoon ja ohjelmistoon. AI-Starterin mikrokontrollerina toimii *Arduino Mega2560* -malliin pohjautuva kopio, *DuDuino Mega 2560*. Mikrokontrollerin prosessorina käytetään *atmega2560*-nimistä prosessoria, joka on ohjelmoitavissa Arduino IDE -ohjelmointiympäristössä. *DuDuino*-mikrokontrolleriin on integroitu moottoriohjain, valo- sekä geomagneettinen sensori, sekä liitännät infrapuna-, ultraääni- ja värisensoreille. Mikrokontrollerissa on kaksi sarjaliitäntää, jotka mahdollistavat kustomoitujen kytkentöjen tekemisen. Mikrokontrollerissa on micro-USB- liitäntä, jota käytetään tiedonsiirtoon sekä virranlataukseen. Mikrokontrollerissa on myös integroitu Bluetooth- toiminto. Bluetooth on langaton tiedonsiirtotekniikka.



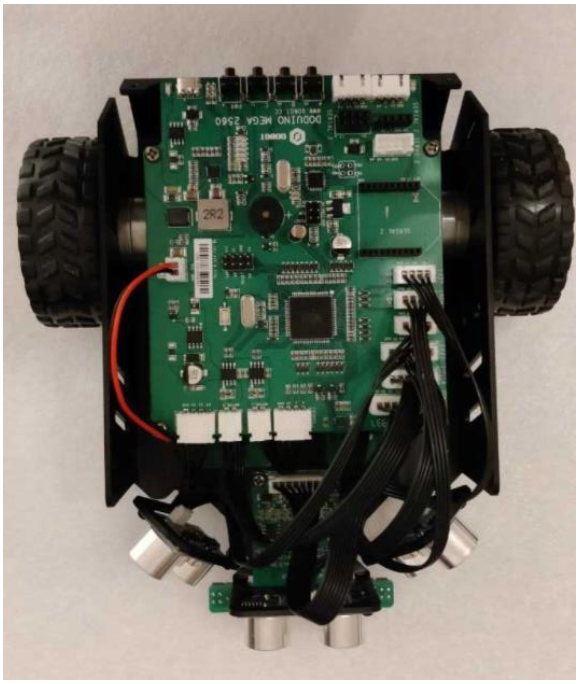
Kuva 14. AI-Starterin mikrokontrolleri, *DuDuino Mega 2560*.

3.4 Kokoonpano

AI-Starter-paketin mukana tulevien ohjeiden avulla robottiauton kokoonpano on nopeahko sekä yksinkertainen tehtävä. Ohjeet ovat kuvitetut sekä monivaiheiset. Robottiauton kaikki komponentit kiinnitetään yhteen lukuisin ruuvein. Robottiauton kokoonpanosta vielä yksinkertaisemman tekee se, että minkäänlaisia juotostöitä ei tähän vaadita. Kaikkia komponentteja varten paketissa on valmiit johdot liitoksineen, jotka tulee kiinnittää oikeisiin portteihin mikrokontrollerissa.



Kuva 15. Kuvaus laitteen pohjasta, johon on asennettu moottorit, sensorit, sekä paristoteline paristoineen. Sensoreihin kiinnitetään kaapelit tässä vaiheessa kokoonpanoa.



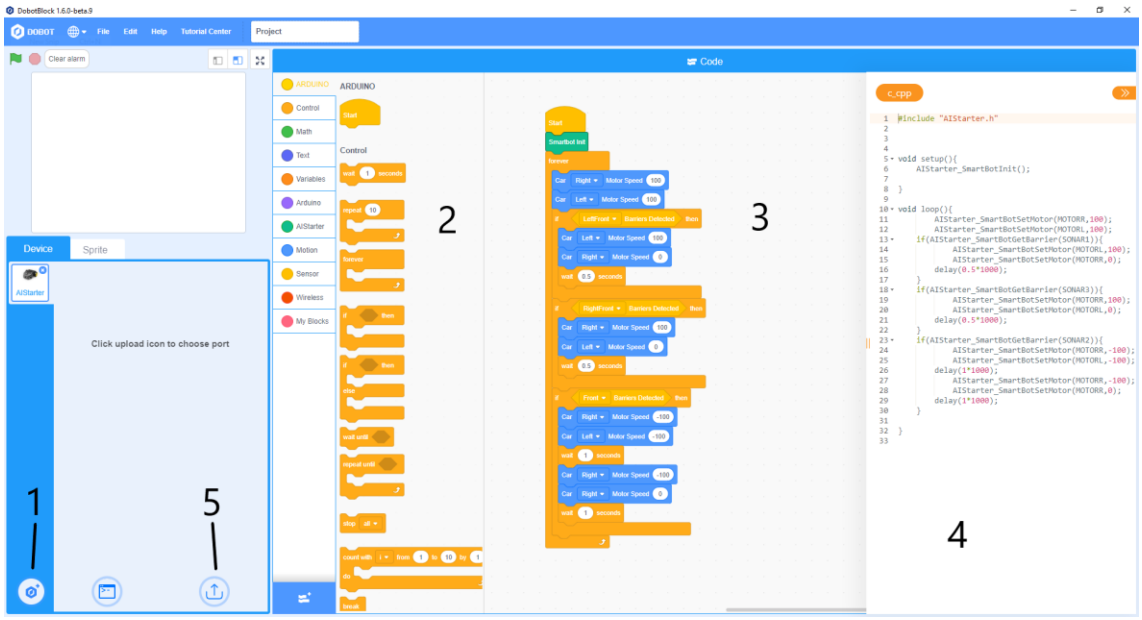
Kuva 16. Pohjaan asennetaan neljä holkkia, joiden päälle mikrokontrolleri kiinnitetään. Paristo sekä sensoreilta tulevat kaapelit kiinnitetään mikrokontrolleriin. Tässä vaiheessa kokoonpanoa asennetaan myös robottiauton renkaat laitteen sivuille, sekä universaali rengas pohjaan.

Kuva 17. Jäljelle jää vain rungon yläosan kiinnittäminen.

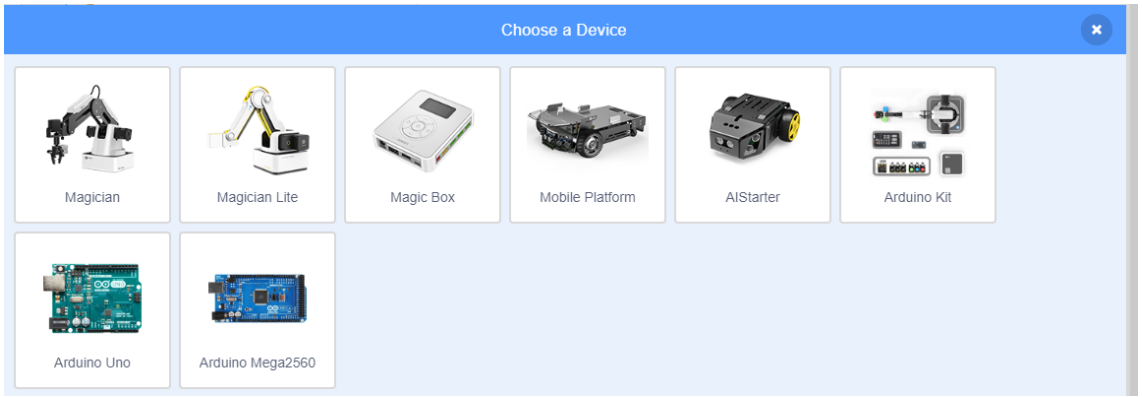
3.5 Ohjelmat

3.5.1 DobotBlock

DobotBlock on helppokäyttöinen opetuskäyttöön suunnattu graafinen ohjelmointiympäristö. Graafinen ohjelmointi ei vaadi käyttäjältä tavanomaisia ohjelmointitaitoja, sillä se perustuu graafisiin komentolohkoihin, joita yhdistelemällä saadaan robotille muodostettua ohjelma. DobotBlock-ohjelmasta löytyy valmiit komentolohkokirjastot eri tuotteille. Tuotekohtainen kirjasto täytyy aina valita kullekin laitteelle oikein. Ohjelmassa on myös mahdollista luoda omia graafisia komentolohkoja. DobotBlock-ohjelma luo koodia komentolohkojen mukaan joko python- tai C++-ohjelmointikielillä. DobotBlock-ohjelmointiympäristön ollessa vielä kehitysvaiheessa, ei sieltä löydy vielä kaikille sen tukemille laitteille esimerkkiohjelmia, eikä opetusmateriaalia. Lisäksi osa sen kirjastosta, sekä toiminnoista saattavat olla puutteellisia.



Kuva 18. Kuvassa DobotBlock-ohjelma. Kohdasta yksi avataan tuotevalikko, josta valitaan laitekohtainen kirjasto. Kohdassa kaksi esiintyy komentolohkokirjaston valikko. Kohdassa kolme on koodausikkuna, jonne lohkot vedetään komentolohkokirjastosta. Kohdassa neljä C++-koodi, graafisten komentolohkojen mukaan. Kohdasta viisi löytyy ohjelman latauspainike, josta valitaan haluttu laite.



Kuva 19. Kuvassa *DobotBlock*-tuotevalikko.

Kuva 20. Esimerkkikuvaus vilkkuvasta LED-valosta ja sen ohjelmoinnista DobotBlock-ohjelmointiympäristössä.

3.5.2 Arduino IDE

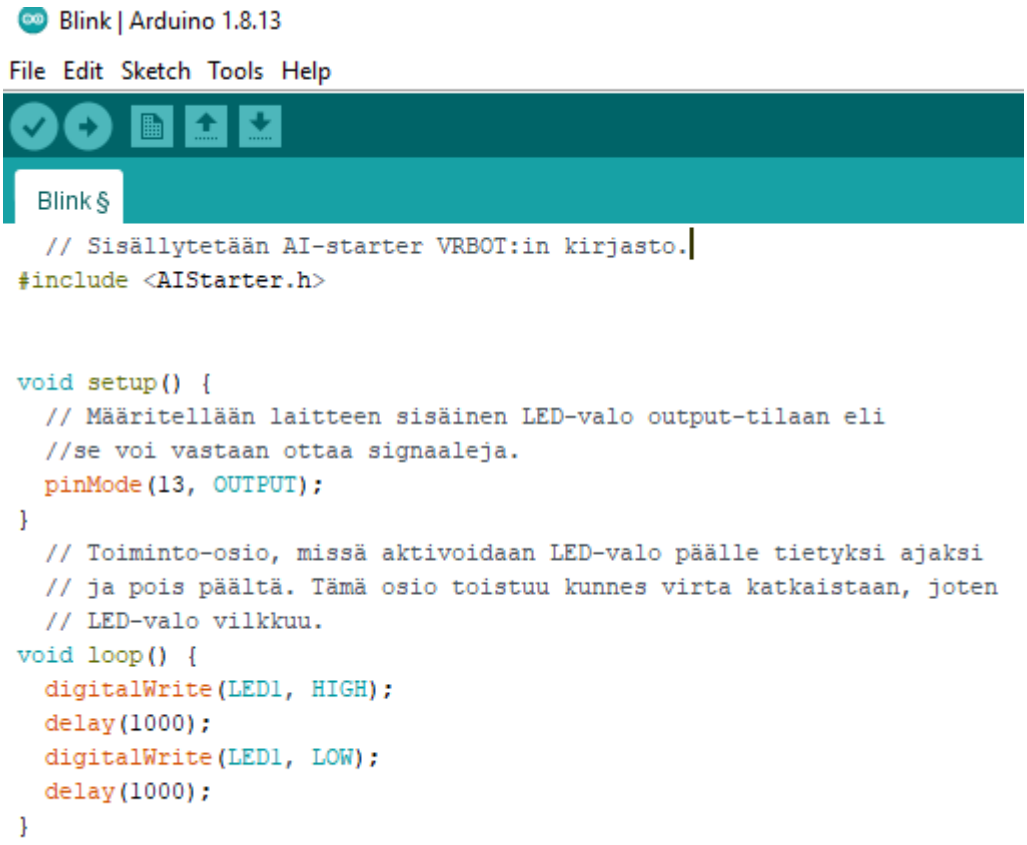
Arduino IDE on avoimeen laitteistoon perustuva mikro-ohjainalusta ja ohjelmointiympäristö. Avoimella tarkoitetaan sitä, että ohjelmisto on julkinen, ja kuka tahansa voi kopioida sitä. Tämän johdosta Arduinon mikro-ohjaimet ja niihin pohjautuvat mallit ovat saavuttaneet suuren suosion harrastelijoiden keskuudessa. Arduino IDE, eli integroitu kehitysympäristö on ohjelma, joka perustuu siis avoimeen lähdekoodiin ja on vapaasti ladattavissa.

Arduinon ohjelmointiympäristö soveltuu myös aloitteleville ohjelmoijille, sillä sen oppimisympäristö on pyritty pitämään mahdollisimman yksinkertaisena. Sen saavuttaman suuren suosion sekä avoimuutensa johdosta verkosta on ladattavissa runsaasti jo valmista oppimismateriaalia siihen liittyen. Valmismateriaalia on mahdollista soveltaa omiin ohjelmointiin liittyviin töihin.

IDE:n avulla luodaan ohjelma, jolla määritellään, mitä Arduinon mikrokontrollerialusta siis tekee. IDE:n ohjelmointikielinä toimii C- ja C++ -kielet. IDE:tä voidaan käyttää useimmilla käyttöjärjestelmillä. IDE-ohjelmisto sisältää valmiita ohjelmakirjastoja, jotka helpottavat ohjelmoimista. IDE- ohjelma koostuu kahdesta pääfunktiosta, jotka ovat:

- Void Setup () – funktio. Funktion tarkoitus on alustaa laite, eli määritellä kaikkien muuttujien asetukset. Funktio luetaan vain kerran, kun virta kytketään päälle.

- Void Loop () - funktio sisältää ohjelman pääkoodin, eli laitteen toiminnot. Loop-funktio toistaa itseään niin kauan, kunnes laitteen virta katkaistaan.



Kuva 21. Esimerkki vilkkuvan LED-valon ohjelmoinnista Arduino IDE- ohjelmointiympäristössä.

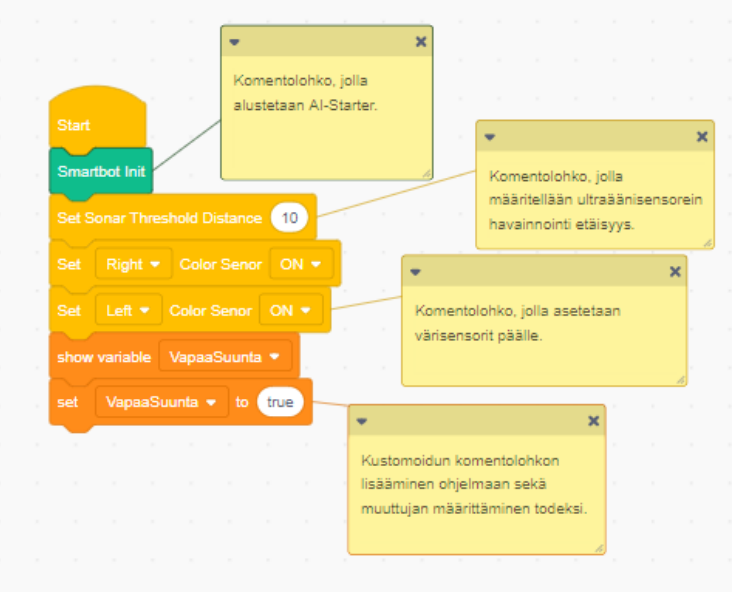
4 Laboratoriotyöt

4.1 Laboratoriotyöt

Metropolian ammattikorkeakoulun konetekniikan koneautomaatiota käsittelevillä kurs-seilla kurssikokonaisuudet muodostuvat teoriaosuudesta, sekä erilaisista käytännönlä-heisistä laboratoriotöistä. Laboratoriotyöskentelyssä suoritetaan erilaisia harjoituksia hyödyntäen myös aiempia teoriaoppeja eri kurssien aihepiirien osioista. AI-Starter-lait-teen avulla tuotettu opetusmateriaali soveltuu koneautomaatiosuuntauksen ”sulautetut järjestelmät ja ohjelmointi” -kurssille hyvin sen käytännönläheisen ohjelmointipainotte-isuuden vuoksi.

4.2 Työ 1

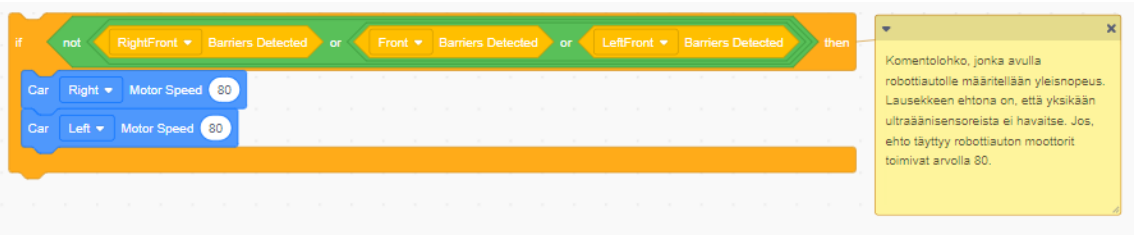
Tässä työssä luodaan ohjelma, jonka tavoite on saada VRBOT, eli robottiauto, liikku-maan sille rajatussa tilassa, ja ohjaamaan mobiilirobotti ilman törmäyksiä haluttuun maa-liin. Maalina toimii sinisellä värillä merkattu alue, johon laite pysähtyy sen tunnistaessa oikean värin. Sinisen värin mobiilirobotti tunnistaa kahdella sen pohjaosassa sijaitsevan värisensorin avulla. Ohjelma luodaan DobotBlock-ympäristössä graafisten komentoloh-kojen avulla. Graafinen ohjelmointi DobotBlockilla koostuu kahdesta osiosta, jotka ovat ”Setup”, ja ”Loop”. Ohjelman aluksi määritellään asetukset, alkuarvot ja muuttujat sekä liittimet input- ja output-tilaan.



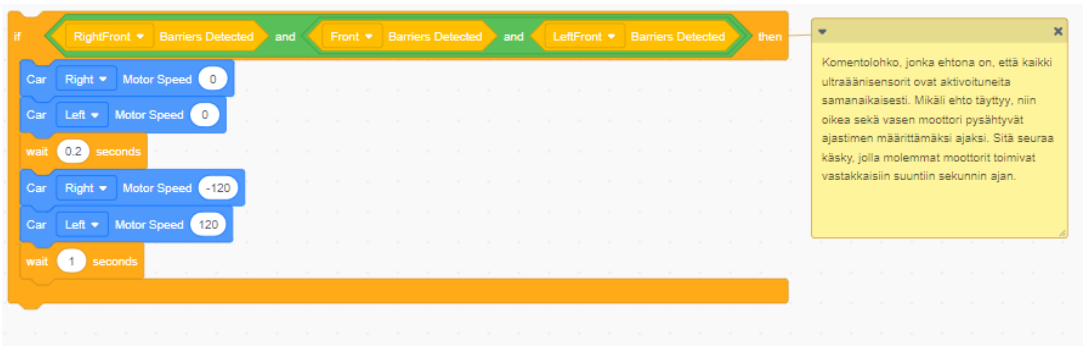
Kuva 22. Kuvaus siitä, kuinka suorittaa ohjelman asetusten määrittely.

Asetuksien määrittämisen jälkeen seuraa toimintaosio, eli *Loop*. Loop-silmukassa siis määritellään, kuinka VRBOT-laitteen tulee toimia. Robottiauto liikkuu moottoriensa avulla. Laitteen liikettä voi muuttaa säätämällä ohjelmasta sen nopeusarvoja, ja määrittämällä ohjelman ajastinten avulla, kuinka kauan haluttua liikettä tapahtuu. Ajastinten avulla määritellään siis liikkeille niiden halutut kestot. Moottoreiden nopeuksia voidaan säädellä numeroarvoilla -255–255.

Ohjelma koostuu pääsääntöisesti if-else-ehtolausekkeista, jotka ovat erilaisia väittämiä. Ohjelmaan asetetun väittämän ollessa tosi, suorittaa ohjelma määrätyn toimen. Tämä toimi seuraa heti lausekkeen ”if”-osuuden jälkeen. Väittämän taas ollessa epätosi, ohjelman suoritus jatkuu else-osasta. Ehtolauseke noudattaa siis loogista kaavaa. If-ehtolauseketta voidaan käyttää myös ilman else-osaa.

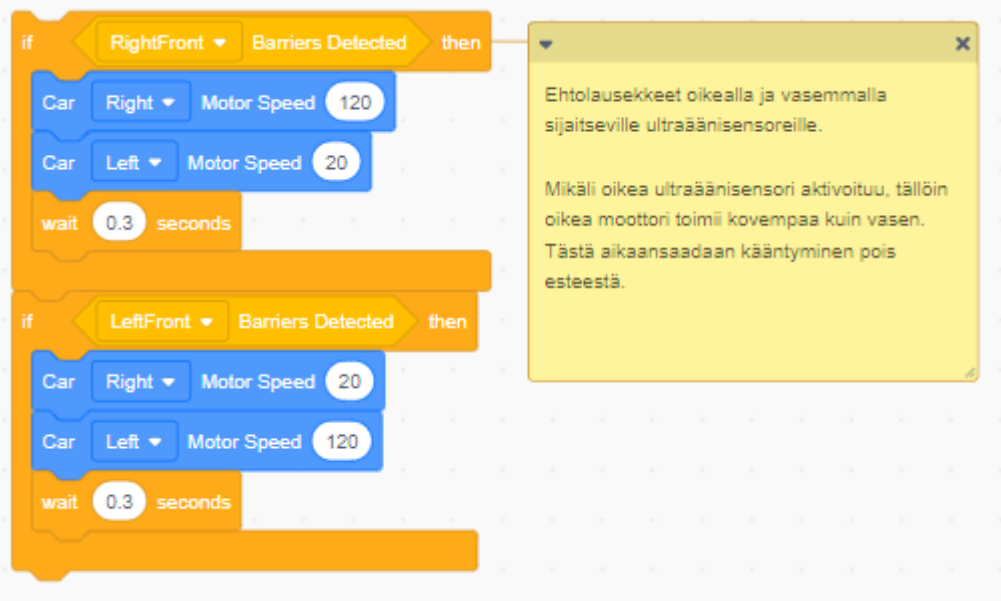


Kuva 23. If-ehtolauseke, jonka avulla saadaan robottiauto ajamaan eteenpäin, jos mikään ultraäänisensoreista ei havaitse esteitä.

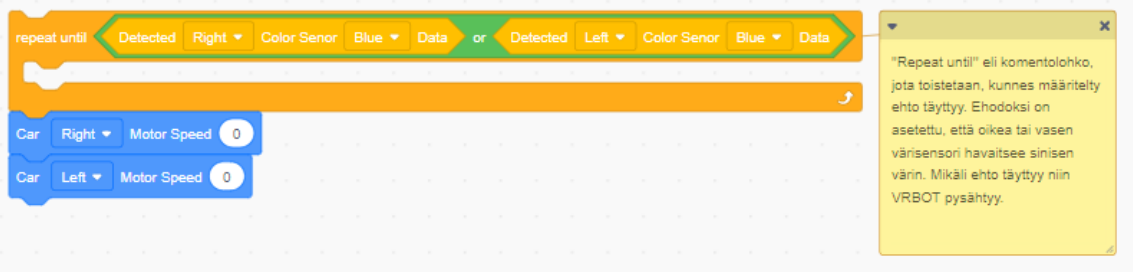


Kuva 24. If-ehtolauseke, jonka avulla robottiauto kääntyy ympäri, mikäli kaikki sen ultraäänisensoreista aktivoituvat. Tämä estää laitteen jumittumisen umpikujaan.

Kuva 25. If-ehtolauseke, jossa määritellään tilanne, mitä tapahtuu laitteen etusensorin havaitessa. Lausekkeen sisälle määritellään tapahtumat, jotka toteutuvat vain, mikäli ensimmäinen ehtolause toteutuu.



Kuva 26. If-ehtolauseet, joiden avulla robottiauto väistää oikealla sekä vasemmalla havaitut esteet.

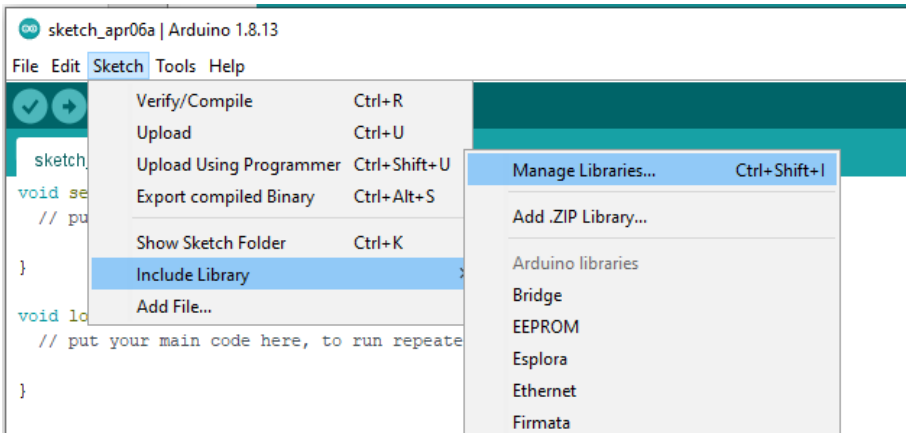


Kuva 27. Repeat until -ehtolause, jonka sisään kaikki muut ehtolauseet kuuluvat. Ehtolausetta toistetaan, kunnes ehto maalista havaitaan. Maaliksi tässä tehtävässä on määritelty sinisellä merkitty alue.

4.3 Työ 2

Tässä työssä on tarkoitus tutustua Arduinon luomaan IDE-kehitysympäristöön, sekä C++-ohjelmointikieleen hyödyntäen AI-Starter- opetusrobotiikkaa. Työssä luodaan ohjelma, jonka pyrkimyksenä on saada robottiauto seuraamaan sille määrättyä polkua, ja myös pysymään sille määrätyllä reitillä. Työssä hyödynnetään AI-Starter-paketin mukana tullutta valkoista reittialustaa, johon on luotu mustalla värillä polku. DOBOT-verkkosivustolta on ladattavissa laitteen kehittäjän luomia esimerkkiohjelmia. Ohjelmat on kirjoitettu C++-ohjelmointikielellä.

Näitä ohjelmia voi sisällyttää IDE:n polulla ”File – Examples – AIStarter” – ja valikoimalla haluttu ohjelma. Mikäli ohjelman luo itse, täytyy siihen sisällyttää AI-Starter-kirjastot, jotta robottiautoa kykenee ohjaamaan oikeanlaisesti. Kirjastot voidaan ladata joko IDE:n kirjastohakukoneen avulla, tai DOBOT- verkkosivustolta ”Zip”- muodossa. Kirjastot sisällytetään ohjelmaan IDE:n Library Managerin kautta seuraamalla polkua ”Sketch – Include Library”, ja valikoimalla haluttu kirjasto.



Kuva 28. Kuvaus polusta, kuinka kirjastot lisätään Arduino IDE- ohjelmointiympäristössä.

”*Color recognition and line tracking*” on ohjelma, jossa robottiauto ohjataan seuraamaan ohjausalustan viivaa ja suorittamaan sille määriteltyjä muitakin tapahtumia. Ohjelmassa laite hyödyntää reitin seuraamiseksi sen infrapunasensoreita. Ohjelmaa varten robottiautolle määritellään sen niin kutsuttu nolla-asento, joka sijaitsee robottiauton kuuden infrapunasensorin keskipisteessä. Robotin kääntyessä esimerkiksi sivuttaissuunnassa, positio nolla-asennosta muuttuu, luoden ”virhettä”, eli erotusta nollapositioon nähden. Virheen kasvaessa ohjelma säätelee moottoreiden nopeuksia kääntääkseen laitteen takaisin oikealle reitille, eli suuntaan kohti nollapositiota. Moottoreiden nopeuksia ohjataan PI-säädön avulla.

Termi ”PI” muodostuu englanninkielisistä sanoista, joita ovat proportional (suhteellinen), ja integral (integroiva). PI-säädön avulla määritellään, kuinka voimakkaasti ja nopeasti robottiauto reagoi sen position poikkeamiin.

Suhteellisella osalla (P) tarkoitetaan toimintoa, jossa säätimen ulostulo (ohjaussuure) on suoraan verrannollinen säätimen sisäänmenoon. Tyypillisesti sisäänmenona käytetään

erosuuretta E, joka on asetusarvon ja mittausarvon erotus, eli säädettävän suureen poikkeama halutusta arvosta. Tällöin P-osan ulostulon itseisarvo on siis sitä suurempi, mitä kauempana toivotusta säädettävän suureen arvo on. Vahvistus ”Kp” kuvaa säätötoimenpiteen voimakkuutta. P-säädön kertoimen suuruus siis vaikuttaa siihen, kuinka vahvasti se reagoi virheeseen. Yleensä P-säädöstä jää pysyvä offset-säätöpoikkeama, joka aiheuttaa turhaa edestakaista liikettä. (Servo- ja säätötekniikka, luentomateriaali)

Integroiva osa (I) kertoo erotuksen määrän suureeseen P tarkasteltavalta ajanjaksolta. Sen ulostulo on siis suhteessa paitsi erosuureen suuruuteen, myös sen kestoaikaan. Integroivan termin vahvistuksena käytetään integrointivahvistusta ”Ki”. I-säätö laskee erosuureen poikkeamaa kumulatiivisesti, joka johtaa siihen, että termit kumoavat toisiaan ja säätö hidastuu kohti tasapainotilaa. (Servo- ja säätötekniikka, luentomateriaali)

Mikäli P-säätö on liian suuri tai pieni, reagoi laite virheeseen liian voimakkaasti tai hitaasti. Tämä voi aikaansaada laitteen suistumista sille määrätyltä. P- ja I-säätöjen optimaaliset parametrit löytyvät niitä hakemalla ja virittämällä.

```
#include <AIStarter.h>
#define  Threshold 20              //Set color threshold
#define  IR_NUM    6               //Set number of infrared pairs
#define  DBG_EN    0
#define  DBG_Loop 1
int lineState;                     //Set the status of line patrol, black line in and out
bool colorRec = false;             //Color recognition status bit
int beepFlag = LOW;                //Buzzer status

/************** lineState, Set the status of line patrol, black line in and out **************/
enum{
    LINEPATROL,                     //Line inspection
    ENTERBLACK,                     //Into the black line
    EXITBLACK                       //Black line
};

/************** The color status detected by the color sensor, red, green and others **********
enum{
    OTHER,
    RED,
    GREEN
};
/************** initialization **************/
void setup()
{
    Serial.begin(115200);          //Set the serial port baud rate
    pinMode(13, OUTPUT);
    pinMode(36, INPUT);
    AIStarter_SmartBotInit();       //Car initialization
    AIStarter_SmartBotSetLED(LED1,BLINK);       //LED1闪烁
}
```

Kuva 29. Ohjelman alussa sisällytetään laitteistoon siihen liittyvät kirjastot. Tämän lisäksi määritellään laitteen sen asetuksia, alkuarvoja, muuttujia sekä sen liittimet input- ja output-tilaan**.**

```
/************** Get line tracking sensor status **************/
void getCurrentIRState(int *irstate)
{
    *irstate = 0;
    for (int i = 0; i < IR_NUM; i++) {
        *irstate |= AIStarter_SmartBotGetIRModuleValue(i) << i;
    }
}
```

Kuva 30. Ohjelmalla luodaan muuttuja, joka kertoo infrapunasensorien tilasta.

Infrapunasensorien tila voi olla inaktiivinen, jolloin sen arvo on 0. Aktiivisena infrapunasensorin arvo on 1.

```
/************** Get the current car body posture ***************/
float getCurrentPos(const int irstate)
{
    const float coeff = 0.7;
    const int irPos[] = {-30, -18, -6, 6, 18, 30};//mm
    static float lastPos;
    float curPos;
    float readPos;
    int total = 0;
    int irOffCnt = 0;
    //calculate the car position offset
    for (int i = 0; i < IR_NUM; i++) {
        if (irstate & (1 << i)) {
            total += irPos[i];
            irOffCnt++;
        }
    }
    if (irOffCnt) {
        readPos = total / irOffCnt;
    }
    else {
        readPos = lastPos;
    }
    //calculate the current position
    curPos = (1 - coeff) * lastPos + coeff * readPos;
    lastPos = curPos;
    return curPos;
}
```

Kuva 31. Ohjelman osio, jossa asetetaan robottiautolle sen nollapiste. Tässä osassa lasketaan myös laitteen siirtymä nollapisteeseen verrattuna, ja asetetaan tämä nykyiseksi sijainniksi.

Infrapunasensoreille määritellään etäisyydet nollapisteestä eli kohdasta, joka on keskellä robottiautoa. Tässä ohjelman osiossa luodaan myös sellainen lauseke, jonka avulla saadaan robottiauton nykyinen sijainti. Laitteen nykyinen sijainti määritetään sen edellisen sijainnin, ja nykyisen siirtymän summasta.

```
/************* Set car speed **************/
void setCarSpeed(const float curPos)
{
    const int baseSpeed = 100; //rpm
    //baseSpeed 70  kp 1  ki 0.06

    const float kp = 1.6;
    const float ki = 0.06;
    const float kd = 0.0;
    const float errorsumLimit = 100;


    float error = curPos;
    static float lastError;
    static float errorsum;
    float errorChange;
    int speedLeftWheel;
    int speedRightWheel;
    int speedOffset;

    //pid
    errorsum += error;
    if (errorsum > errorsumLimit) {
        errorsum = errorsumLimit;
    }
    else if (errorsum < -errorsumLimit){
        errorsum = -errorsumLimit;
    }
    errorChange = error - lastError;
    speedOffset = kp * error + ki * errorsum + kd * errorChange;
    lastError = error;

    //calculate the wheel speed
    speedLeftWheel  = baseSpeed + speedOffset;
    speedRightWheel = baseSpeed - speedOffset;
```

Kuva 32. Tässä ohjelmiston osiossa luodaan robottiauton nopeuteen vaikuttavia tekijöitä.

Moottoreille määritetään yleisnopeus, joita noudattaen robotti etenee.
Robotin kääntyminen aikaansaadaan, kun moottoreiden yleisnopeuteen lisätään muuttujana termi ”*speedOffset*”. *SpeedOffset* lasketaan laitteen nykyisestä sijainnista eli virheen ja virheen itseisarvon summasta. Virhettä ja virheen itseisarvoa kerrotaan määritetyillä kertoimilla Kp ja Ki, jotta haluttu reaktio saadaan aikaiseksi.

Robottiauton yleisnopeudeksi asetetaan arvo 100. Suhteelliseksi kertoimeksi (Kp) asetetaan 1.6 ja integroivaksi kertoimeksi (Ki) 0.06. Näillä kertoimilla robottiauto seuraa viivaa ilman turhaa edestakaista liikettä.

```
/************* The color sensor detects events other than the red and green lines **************/
void otherLineEvent()
{
    delay(3000);
}

/************* The color sensor detects an event that occurred on the red line **************/
void rLineEvent()
{
    for(uint8_t i = 0; i<6; i++) {
        digitalWrite(BEEP,beepFlag = !beepFlag);
        delay(500);
    }
}

/************* The color sensor detects an event that occurred on the green line **************/
void gLineEvent()
{
    digitalWrite(BEEP,beepFlag = !beepFlag);
    delay(1000);
    digitalWrite(BEEP,beepFlag = !beepFlag);
}
```

Kuva 33. Tässä ohjelman osiossa luodaan halutut tapahtumat laitteen tunnistaessa vihreää, tai punaista väriä.

Tapahtumaksi on valittu äänimerkki, jonka kesto määritetään ajastimen avulla.

```
/************* Main loop **************/
void loop()
{
    if(!AIStarter_SmartBotGetKeyValue(37)) {
        delay(5);
        if(!AIStarter_SmartBotGetKeyValue(37)) {
            digitalWrite(BEEP,HIGH);
            delay(10);
        }
    } else {
        if(AIStarter_SmartBotGetKeyValue(37)) {
            digitalWrite(BEEP,LOW);
        }
    }
    int irstate;
    float curPos;
    int colorState;
    getCurrentIRState(&irstate);
    Serial.print("irstate = ");
    Serial.println(irstate);
    if(irstate == 60) {
        if(lineState == LINEPATROL) {
            lineState = ENTERBLACK;
        } else if(lineState == EXITBLACK) {
            AIStarter_SmartBotSetMotor(MOTORL, 50);
            AIStarter_SmartBotSetMotor(MOTORR, 50);
        }
    } else {
        lineState = LINEPATROL;
        curPos = getCurrentPos(irstate);
        setCarSpeed(curPos);
    }
```

Kuva 34. Kuva pääohjelmasta.

Pääohjelmassa määritellään seuraavat tilat: viivan, eli kulkureitin seuranta, värillinen alue, ja poistuminen värilliseltä alueelta. If-else-ehtolausekkeella määritellään, mikäli infrapunasensorien tila saa arvon 60, on laite värillisellä alueella. Muissa tapauksissa jatketaan kulkureitin seurantaa. Arvo 60 saadaan silloin, kun kaikki kuusi infrapunasensoria ovat aktiivisena. Ohjelmassa luodaan myös värilliseltä alueelta poistumistapahtuma (EXITBLACK), jotta viivan seuraaminen voi jatkua.

```
switch (lineState) {
    case LINEPATROL:
        colorRec = false;
    break;
    case ENTERBLACK:
        colorRec = true;
    break;
    case EXITBLACK:
        colorRec = false;
    break;
    default:
    break;
}
while(colorRec){
    AIStarter_SmartBotSetMotor(MOTORL, 0);
    AIStarter_SmartBotSetMotor(MOTORR, 0);
    Serial.println("colorRec Stop");
    delay(50);
    if(AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR) - AIStarter_SmartBotGetColorSenor(COLORSENOR1,GCOLOR) > Threshold &&
       AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR) - AIStarter_SmartBotGetColorSenor(COLORSENOR1,BCOLOR) > Threshold) {
        colorState = RED;
        Serial.println("ENTER RLINE");
    } else if(AIStarter_SmartBotGetColorSenor(COLORSENOR1,GCOLOR) - AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR) > Threshold &&
       AIStarter_SmartBotGetColorSenor(COLORSENOR1,GCOLOR) - AIStarter_SmartBotGetColorSenor(COLORSENOR1,BCOLOR) > Threshold) {
        colorState = GREEN;
        Serial.println("ENTER GLINE");
    } else {
        colorState = OTHER;
        Serial.println("ENTER OTHERLINE");
    }
    switch(colorState) {
        case OTHER:
            otherLineEvent();
            colorRec = false;
            lineState = EXITBLACK;
        break;
        case RED:
            rLineEvent();
            colorRec = false;
            lineState = EXITBLACK;
        break;
        case GREEN:
            gLineEvent();
            colorRec = false;
            lineState = EXITBLACK;
        break;
        default:
        break;
    }
}
```

Kuva 35. Pääohjelma.

Tässä pääohjelman osassa määritellään, onko laitteen värien tunnistus kytketty aktiiviseksi, vaiko inaktiiviseksi, ja ehdot näihin liittyen. Värien tunnistaminen aktivoituu, jos ehto värillisestä alueesta täyttyy. Silloin, kun värien tunnistaminen laitteessa on aktivoitunut, saa sen moottorit nopeusarvoksi arvon 0. Robottiauton ollessa pysähtyneenä, määritellään, määrittää laite havaittua väriä. Kun havaittu väri on tunnistettu, seuraa tästä kyseiselle värille määritelty tapahtuma. Tapahtuman jälkeen värintunnistus muutetaan

inaktiiviseksi, jota seuraa toiminto ”EXITBLACK”. Robottiauto liikkuu tällöin eteenpäin ja poistuu värilliseltä alueelta. Tästä seuraa laitteen infrapunasensoreiden arvojen muutos. Arvon muutoksesta tila ”LINEPATROL” astuu voimaan, ja kierto alkaa täten alusta.

5 Yhteenveto

Opinnäytetyön tavoitteena oli luoda opetusrobotiikan avulla Metropolian ammattikorkeakoulun koneautomaation opintoihin soveltuvaa opintomateriaalia. Tavoitteeseen päästiin ensin tutustumalla ja opiskelemalla huomattava määrä aiheeseen liittyvää materiaalia.

Harjoituksissa on käytetty kahta eri lähestymistapaa robotiikan ohjelmointiin. Ensimmäisen harjoituksen graafinen ohjelmointi ei vaadi sen oppijalta perinteisiä ohjelmointitaitoja, mutta noudattaa kuitenkin samoja periaatteita, kuin perinteinen ohjelmointi. Toisen harjoituksen ohjelma on kirjoitettu perinteisellä C++-kielellä. Mielestäni graafisten komentolohkojen käyttö oli helppo sekä opettavainen tapa lähestyä ohjelmointia tilanteessa, jossa en tuntenut edes ohjelmoinnin perusteita. Toisessa harjoituksessa on myös mahdollista käyttää kehittäjän luomaa esimerkkiohjelmaa ja muokata sitä niin, että se on optimoitu käyttöympäristöön. Harjoituksessa on tarkoitus oppia lukemalla jo valmista koodia, mikäli käyttäjällä ei ole entuudestaan ohjelmointitaitoja. Mikäli oppijalta löytyy jo peruskäsitys ohjelmoinnista, voi hän tällöin luoda harjoituksen mukaisen ohjelman myös itse. Graafinen ohjelmointi on oiva tapa tutustua ohjelmointiin, mutta konkretiassa sitä ei juurikaan käytetä.

AI-Starter luo perinteisestä ohjelmoinnin opiskelusta mielekkäämpää, sillä robottiautoa ohjelmoidessaan oppija voi nähdä konkreettisesti, kuinka kirjoitetut koodit vaikuttavat laitteen liikkeisiin. Opinnäytetyössä luotujen oppimateriaalien kautta oppija saa myös käsityksen mikrokontrolleri- sekä sensoritekniikasta. AI-Starter-opetusrobotiikka kannustaa oppijaa itsenäiseen kokeiluun, joka edesauttaa oppimista.

Harjoitusten ollessa yksinkertaisia, monia AI-Starterin ominaisuuksia jäi käyttämättä. Robottiauto on varustettu todella rikkaalla anturitekniikalla, ja tekniikan avulla on mahdollista luoda monimutkaisempiakin toimintoja. Laitteen alusta on myös hyvin muokattavissa, mikä mahdollistaa kustomoitujen lisätoimintojen tekemisen.

Muutamia esimerkkejä jatkokehityksistä, joita robottiautolla voisi tehdä:

1. Bluetooth-ominaisuuden hyötykäyttö, esimerkiksi tiedonsiirtoon muiden laitteiden kanssa, tai robotin ohjaamiseen ulkoisesti.

2. Toisen luodun harjoituksen perusteella opiskelijat voisivat luoda radan, jossa kilpailtaisiin parhaasta kierrosajasta. Jokainen ryhmä loisi ohjelmia, jonka avulla laite seuraisi sille luotua rataa. Se oppilasryhmä, jonka ohjelman avulla VRBOT navigoisi radan läpi kaikista nopeimmin, voittaisi kilpailun. Tämä harjoitus voisi tehdä ohjelmoinnin opiskelusta mielenkiintoisempaa.

3. AI-Starter-robottiauton voisi myös yhdistää muihin DOBOT-sarjan tuotteisiin, joita löytyy Metropolian ammattikorkeakoululta, ja muodostaa esimerkiksi tuotantolinjan.

Lähteet

Arduino 2021. Verkkoaineisto. < https://www.arduino.cc/ >. Luettu 26.03.2021.

Blazic, Saso; Klancar, Gregor; Skrianc, Igor & Zdesar, Andrej. Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems. E-kirja.

Carlson, Jennifer. 2004. Analysis of How Mobile Robots Fail in the Field. Master thesis. Department of Computer Science and Engineering, University of South Florida.

Keinänen, Tomi & Sumujärvi, Matti. 2019. Automaatiotekniikka. Sanoma pro.

Dalamagkidis, Konstantinos. 2014. Handbook of Unmanned Aerial Vehicles. E-kirja. Springer, Dordrecht.

Furgale, Paul. EDX Autonomous Mobile Robots. Verkkokurssi. Katsottu 20.02.2021. https://courses.edx.org/courses/course-v1:ETHx+AMRx+2T2020/course/.

Gage, Douglas W. UGV HISTORY 101: A Brief History of Unmanned Ground Vehicle (UGV) Development Efforts. Unmanned Systems Magazine Summer 1995. volume 13. number 3.

Huang, Hui-Min. 2004. Autonomy Levels for Unmanned Systems (ALFUS) Framework Volume I: Terminology Version 1.1. Verkkoaineisto. National Institute of Standards and Technology.

Liljaniemi, Antti. Servo- ja säätötekniikka. Luentomateriaali. Metropolia Ammattikorkeakoulu.

Nourbakhsh, Illah R & Siegwart, Roland. 2004. Introduction to Autonomous Mobile Robots. E-kirja. Massachusetts Institute of Technology.

Reliability and Failure in Unmanned Ground Vehicle (UGV). 2009. Verkkoaineisto. GRRC Technical Report. https://d1wqtxts1xzle7.cloudfront.net/53238031/200901_ReliabilityUGV.pdf?1495493803=&response-content-disposition=inline%3B+filename%3DReliability_and_Failure_in_Unmanned_Grou.pdf&Expires=1606999448&Signature=BK8ay6Y6Myb~o-y~IWcSjBmIgFpSndKBv3GPbUH8kIeStasofqfEaaLqBpqooyQITL~n769gabygJ2EQQ6ae4~UdYoD4WovlLbae6lJI-JeL8gHq~g29TC0AHirweHoIHRf8QkZgrT3qTaBUk4xhaPyTfNd~sa5tYQYgRjMQfOufgxPj2XWraXam92mdQFUOdU1EM4C5qm7bz2QpfCS1Ur6DkGDjH-FOZTMn7-i6ivNw2TrxRkXea9Pp8ZCA2fxjehxcrq2X8gz~o4O0ZORTWztZn-OiZegxFvXKVTqQckXrgo3k6QkRV6JdpcVWqKAwEQmrBictX0XhTPx-Tb9njJWA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA. Luettu 24.02.2021.

Sensors in robotics – 7 common sensors used in robots. 2019. Verkkoaineisto. https://roboticsbiz.com/sensors-in-robotics-7-common-sensors-used-in-robots/. Luettu 24.2.2021.

Vachtsevanos, George J. 2014. Handbook of Unmanned Aerial Vehicles. E-kirja. Springer Dordrecht.

Kuvasarja 1. https://www.saildrone.com/news/what-is-saildrone-how-work, https://www.saab.com/products/sabertooth, https://www.darpa.mil/news-events/2018-01-30a, https://jamhouri.com/product/uav-rov/.

Kuvasarja 2. https://www.ga-asi.com/remotely-piloted-aircraft/gray-eagle-extended-range, https://store.dji.bg/en/agras-mg-1s-agriculture-drone.html.

Kuvasarja 3. https://www.somagnews.com/boston-dynamics-robot-dog-spot-tested-youtuber/, https://www.rocla-agv.com/en/products/reach-mast-agv,

https://www.unmannedsystemstechnology.com/2019/06/weaponized-ugv-demonstrated-during-live-fire-exercise/.

Kuva 8. Sense- Think- Act cycle
https://www.researchgate.net/figure/Robots-See-Think-Act-Cycle-3_fig2_335127899.

Kuva 9. Dobot yhteistyörobotti
https://www.dobot.fi/shop/product/r0001-dobot-magician-basic-28.

Kuva 13. AI-Starter Kontrolleri, Ai-Starter-User-GuideV1.1.0.pdf
https://www.robotline.fi/attachment/download?attachment_id=5956.

TYÖ 1. Itseohjautuva auto

SISÄLLYSLUETTELO

1. YLEISTÄ
2. TYÖOHJE
3. TYÖSELOSTUS

1. Tehtävän johdanto

Työssä tutustutaan AI-starter- opetusrobotiikkalaitteeseen ja sen toimintaan. Työ toteutetaan graafisella ohjelmointimenetelmällä DobotBlock- ohjelmointiympäristössä. Työssä luodaan ohjelma, jonka avulla robottiauto kykenee liikkumaan autonomisesti suorittaen samalla haluttuja toimintoja.

AI-Starter sisältää robottiauton, jota voidaan ohjata sen mikrokontrollerin avulla. Laite sisältää kaksi moottoria, kaksi värisensoria, kolme ultraäänisensoria sekä kuusi infrapunasensoria.

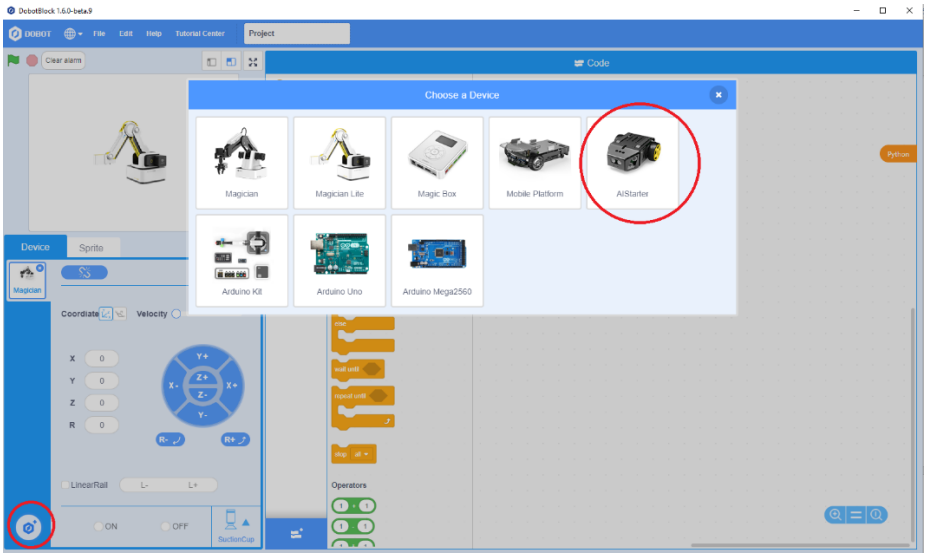
2. Työohje

1. Tutustutaan aluksi AI-Starter opetusrobotiikkaan:

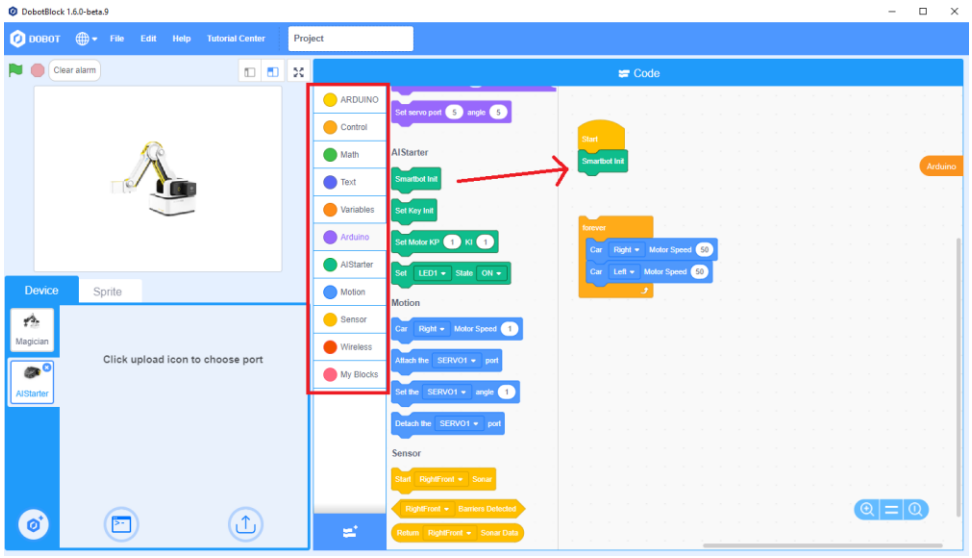
    - Tutustutaan siis laitteiston käyttöohjeisiin
    - Suoritetaan robottiautolle kokoonpano

2. Seuraavaksi ladataan DobotBlock-ohjelmisto alla olevasta verkko-osoitteesta. (https://robotline.fi/robotline_download_center/static/DobotBlock%20Setup%201.6.0-beta.9.zip).

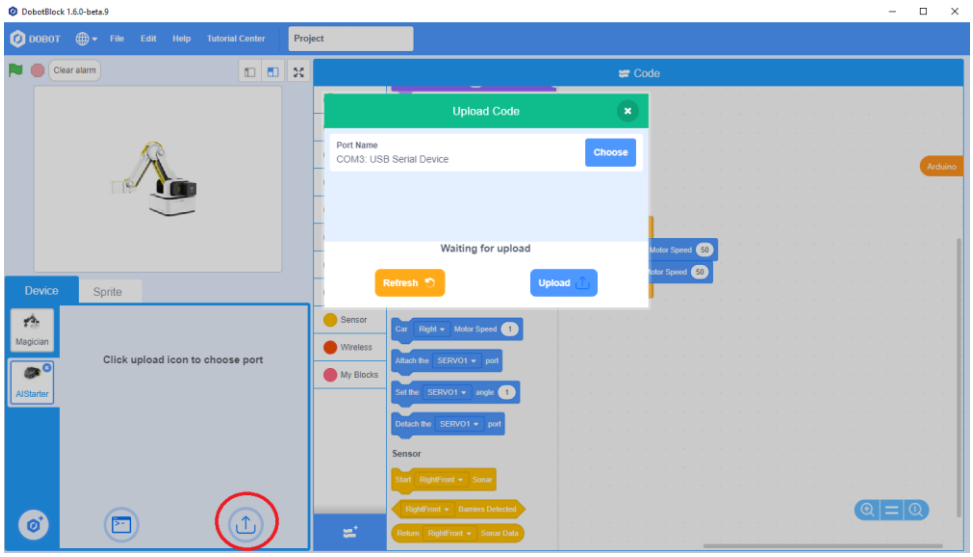
3. Kun ohjelmisto on ladattu, valikoidaan DobotBlock- ohjelmistoympäristöstä laitteelle soveltuva kirjasto. Kirjastot sisältävät komentolohkoja, joiden avulla ohjataan valittua laitetta.



Kuva 36. Kuvassa näkyy DobotBlock- laitteiden valikoima, josta valitaan ohjelmoitava laite.



Kuva 37. Kuvassa näkyy erilaisia komentolohkoja. Halutut komentolohkot vedetään ohjelmointi-ikkunaan, ja niistä muodostetaan ohjelma.

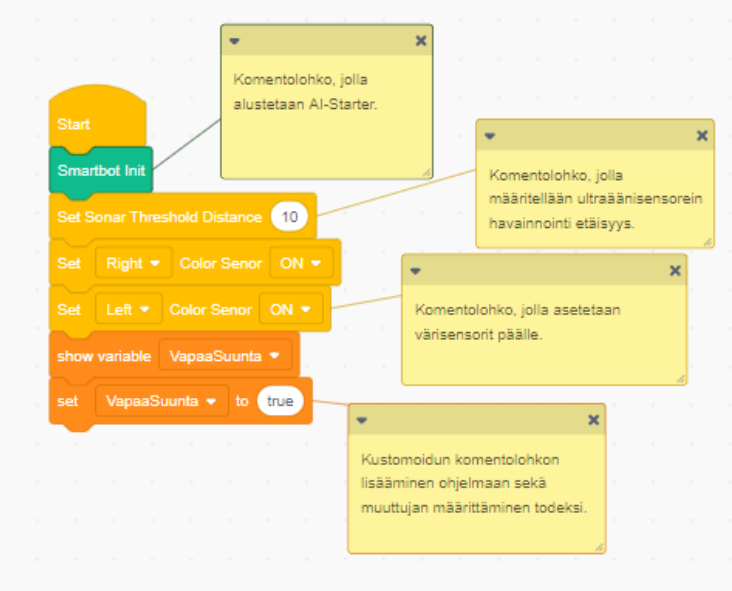


Kuva 38. Ohjelman ollessa valmis, yhdistetään robottiauto tietokoneeseen usb-kaapelilla, ja ladataan laitteeseen sille suunnattu ohjelma.

4. Itseohjautuva auto

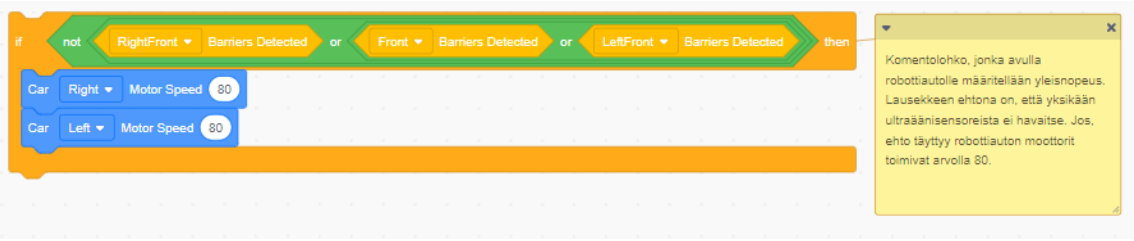
Tarkoituksena on luoda ohjelma, jonka avulla robottiauto kykenisi liikkumaan sille annetussa ympäristössä ilman törmäyksiä. Esteiden havainnointiin hyödynnetään laitteen kolmea ultraäänisensoria.

Graafinen ohjelmointi DobotBlockilla koostuu kahdesta osiosta, joita ovat ”Setup”, sekä ”Loop”. Ohjelman aluksi määritetään laitteelle suunnatut asetukset, alkuarvot ja muuttujat sekä liittimet input- sekä output- tilaan.



Kuva 39. Esimerkkikuvaus siitä, kuinka suorittaa ohjelman Setup, eli asetusten määrittely.

Ohjelmaan täytyy aluksi sisällyttää alustuskomentolohko, joka on ’’Smartbot Init’’. Ohjelman asetusten määrittämisen jälkeen seuraa ”Loop”, eli ohjelman toimintaosio, jossa luodaan robottiautolle sen toiminnot. Tässä tulee huomioonottaa, että laitteen ultraäänisensorien havainnointietäisyydet ovat välillä 10–25 cm. Mikäli ultraäänisensoreille antaa pienemmän arvon kuin 10, ovat ne pysyvästi aktiivisina.



Kuva 40. Esimerkkikuvaus If-ehtolausekkeesta, jonka avulla saadaan robottiauto ajamaan eteenpäin, mikäli yksikään sen ultraäänisensoreista ei havaitse esteitä.

Huom. Moottorien nopeusarvot ovat säädeltävissä arvojen -255–255 välillä ja niiden suurin kierrosnopeus on 100rpm.

Tehtävät:

Ohjelmaa luodessa tulee huomioonottaa, millaisessa toimintaympäristössä laitteen tehtävä toteutetaan. Erilaisissa ympäristössä löytyy erilaisia muuttujia, kuten esteitä sekä muita laitteen toimintaan vaikuttavia rajoitteita.

Luokaa ohjelmaan toiminnot, joiden avulla robottiauto osaa toimia seuraavissa tilanteissa:

- Este laitteen oikealla tai vasemmalla puolella
- Umpikuja
- Kohtisuoraan laitteen edessä oleva este

3. TYÖSELOSTUS

Demonstroikaa valmis ohjelma opettajalle, sekä kirjoittakaa työstänne raportti.

Raporttiin tulee sisällyttää:

- Kuvaus työnkulusta
- Kuvat ohjelmista
- Vastaukset

TYÖ 2. Color Recognition and Line Tracking

SISÄLLYSLUETTELO

1. YLEISTÄ
2. TYÖOHJE
3. TYÖSELOSTUS

1. YLEISTÄ

Työssä tutustutaan AI-Starter opetusrobotiikkalaitteeseen ja sen toimintaan. Laitteen ohjelmointi suoritetaan Arduino IDE-kehitysympäristössä, ja ohjelmointikielenä käytetään C++-kieltä. Työssä luodaan ohjelma, jonka avulla robottiauto kykenee seuraamaan sille merkittyä polkua.

AI-Starter sisältää robottiauton, jota voidaan ohjata sen mikrokontrollerin avulla. Laite sisältää kaksi moottoria, kaksi värisensoria, kolme ultraäänisensoria sekä kuusi infrapunasensoria.

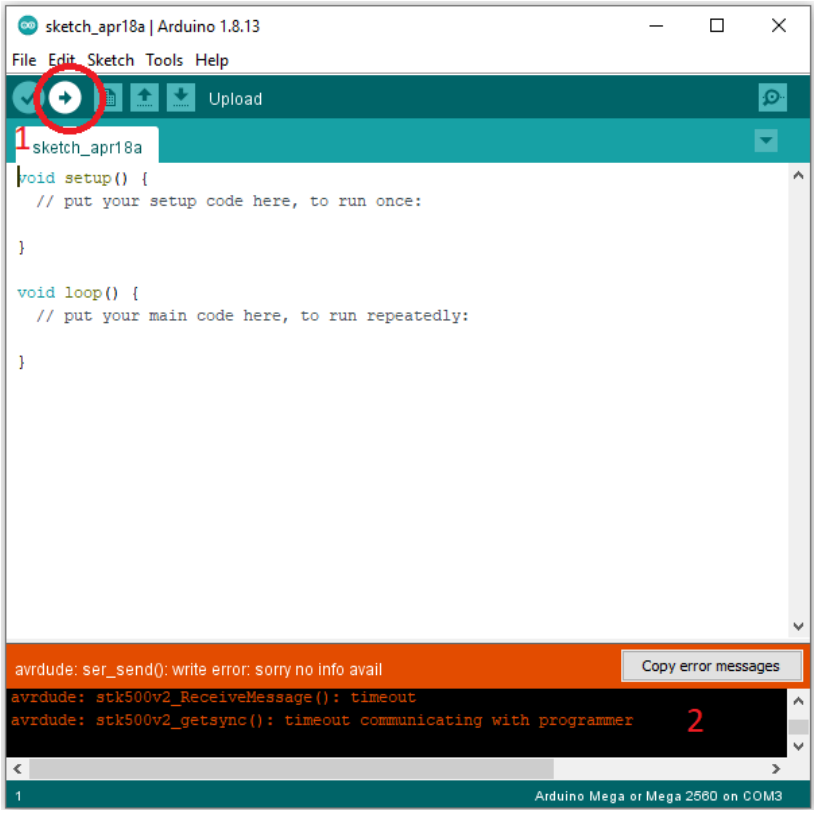
2. TYÖOHJE

1. Tutustutaan aluksi AI-Starter opetusrobotiikkaan:

    - Tutustutaan siis laitteiston käyttöohjeisiin
    - Suoritetaan robottiautolle kokoonpano

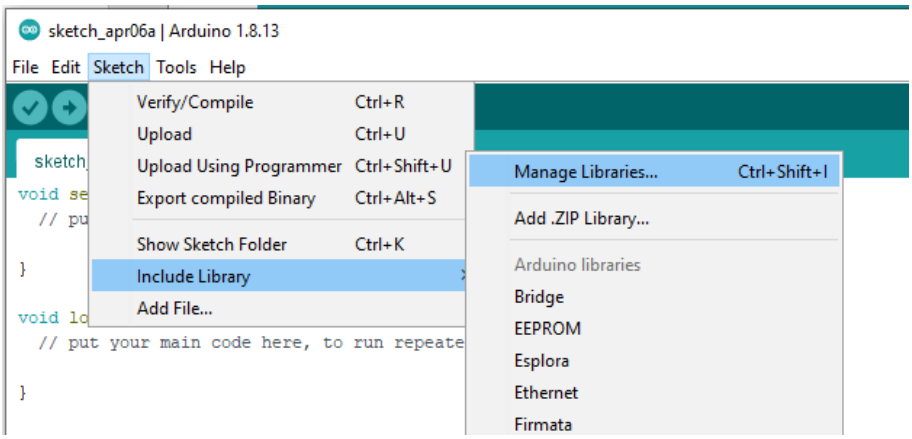
2. Ladataan Arduino IDE-ohjelmisto osoitteesta (https://www.arduino.cc/en/software)

    - Ohjelman lataamisen jälkeen tutustutaan ohjelmistoon.
    - Tämän jälkeen tutustutaan C++-ohjelmointikieleen. Arduinon kotisivuilta löytyy ohjelmoinnin komentokohtaiset selitykset esimerkkeineen. (https://www.arduino.cc/reference/en/)

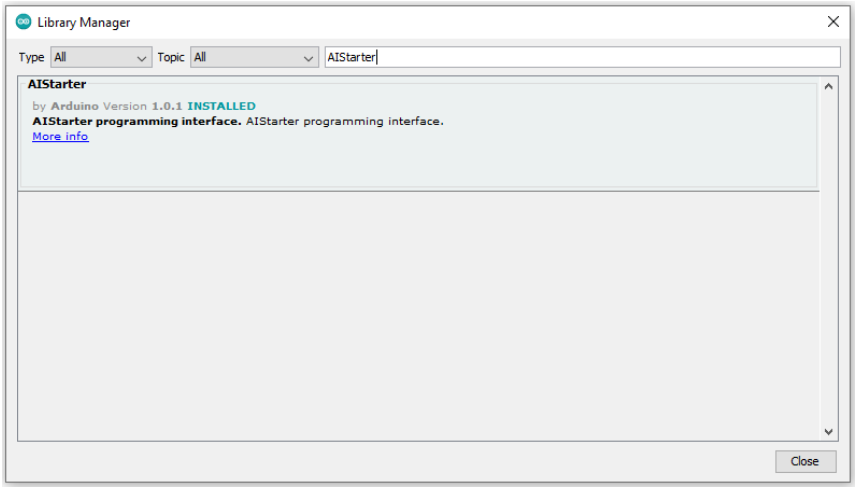


Kuva 41. Kuvassa Arduino IDE-ohjelma. 1) Painike, jolla ladataan ohjelma laitteelle usb-kaapelin kautta 2) Ohjelman debug-ikkuna, joka kertoo mahdolliset virheet koodissa.

3. Arduino IDE-ohjelmaan täytyy sisällyttää AI-Starter- ohjelmakirjasto, jotta laite ohjautuu oikein. Kirjastot sisällytetään ohjelmaan polun ”Sketch – Include Library”, ja valikoimalla haluttu kirjasto.



Kuva 42. Kuvaus polusta, kuinka kirjastot lisätään Arduino IDE- ohjelmointiympäristössä.



Kuva 43. Oikeat kirjastot löytyvät hakusanalla ''AI-Starter''.

4. Color Recognition and Line Tracking

Tarkoituksena on luoda ohjelma, jonka avulla robottiauto pystyy seuraamaan mustalla värillä merkittyä polkua. Polkuna voi toimia AI-Starter- paketin mukana tullut alusta, tai polun voi halutessaan luoda myös itse. Tarkoituksena on hyödyntää laitteesta löytyviä värisensoreita, ja luoda jokin tapahtuma, mikä seuraa värisensorien aktivoitumisesta.

Tehtävässä hyödynnetään kehittäjän luomaa demo-ohjelmaa, joka löytyy polulla ”File – Examples – AIStarter – Color Recognition and Line Tracking''. Kun ohjelma on ladattu IDE:en, tutkikaa sitä ja vastatkaa seuraaviin kysymyksiin:

- Miten robottiauton kulkureitin seuranta toteutetaan?
- Mikä on PI-säädin, ja miten se on toteutettu ohjelmassa?

Tehtävät:

- Asettakaa ajonopeudeksi 100 ja suhteelliseksi kertoimeksi Kp = 1, miten robottiauto seuraa polkua? Entä kertoimen arvolla Kp=5?
- Virittäkää PI-säätö niin, että robottiauto seuraa reittiä optimaalisesti
- Asettakaa nopeudeksi 150 ja virittäkää PI-säätö

Huom. Infrapunasensorit on säädettävä, jotta sensorit toimisivat oikein. Infrapunasensorien havainnointietäisyyttä säädetään sensorin sivusta ruuvimeisselillä.

3. TYÖSELOSTUS

Demonstroikaa valmis ohjelma opettajalle, sekä kirjoittakaa raportti työstä.

Raporttiin:

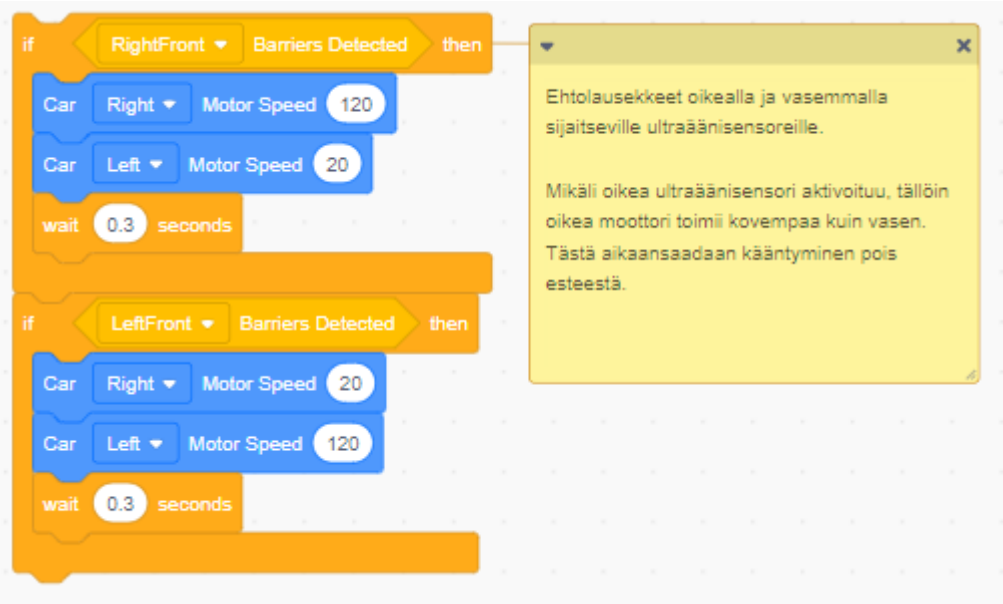
- Kuvaus työnkulusta
- Kuvat ohjelmista
- Kysymyksiin vastaaminen

Liite 3.

TYÖ 1. ITSEOHJAUTUVA AUTO

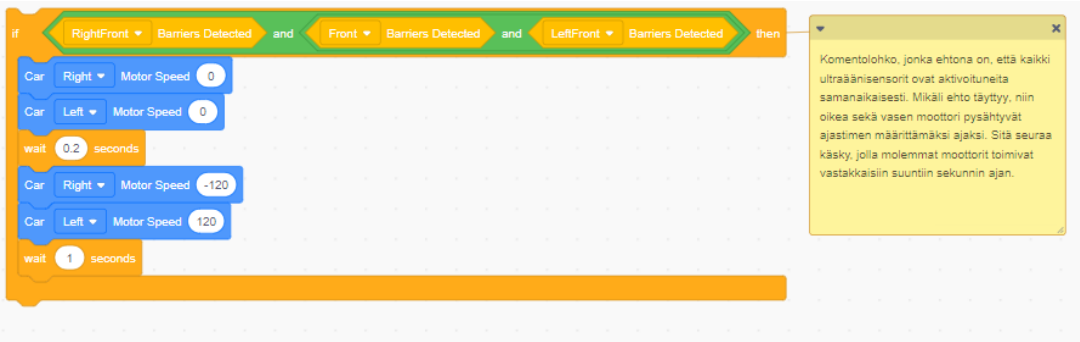
RATKAISUT:

1. **Este oikealla tai vasemmalla**



Kuva 44. If-ehtolauseet, joiden avulla robottiauto väistää oikealla sekä vasemmalla havaitut esteet.

2. **Umpikuja**



Kuva 45. If-ehtolauseke, jonka avulla robottiauto kääntyy ympäri, mikäli kaikki sen ultraäänisensoreista aktivoituvat. Tämä estää laitteen jumittumisen umpikujaan.

3. Kohtisuoraan havaittu este

Kuva 46. If-ehtolauseke, jossa määritellään tilanne, mitä tapahtuu laitteen etusensorin havaitessa. Lausekkeen sisälle määritellään tapahtumat, jotka toteutuvat vain, mikäli ensimmäinen ehtolause toteutuu.

Liite 4.

TYÖ 2: Color Recogition and Line Tracking

RATKAISUT:

Color Recogition and Line Tracking Arduino IDE-ohjelmistossa. Harjoituksessa käytetyt arvot, joilla saatiin parhaat tulokset.

```
/************** Set car speed ***************/
void setCarSpeed(const float curPos)
{
    const int baseSpeed = 100; //rpm

    const float kp = 1.6;
    const float ki = 0.06;
    const float kd = 0.0;
    const float errorsumLimit = 100;
```

Kuva 47. PI-säätimen arvot nopeudelle 100.

```
/************** Set car speed ***************/
void setCarSpeed(const float curPos)
{
    const int baseSpeed = 150; //rpm

    const float kp = 3.6;
    const float ki = 0.02;
    const float kd = 0.0;
    const float errorsumLimit = 100;
```

Kuva 48. PI-säätimen arvot nopeudelle 150.