TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKA TEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Nikolai Konovalov

LEGO Mindstorms NXT robotite programmeerimine NXC keeles

Bakalaureusetöö (6 EAP)

Juhendaja: Anne Villems
Kaasjuhendaja: Taavi Duvin

Autor: ....."....." mai 2013
Juhendaja: ....."....." mai 2013
Juhendaja: ....."....." mai 2013

Lubada kaitsmisele
Professor: ...."....." mai 2013

TARTU 2013

Sisukord

Sissejuhatus

Erinevalt teadusulme (*sci-fi*) filmidest, kus robotid on inimeste jaoks enesestmõistetavad kaaslased igapäevaelus, täites erinevaid majapidamisülesandeid, assisteerides oma omanikke töö juures, aga ka tormates lihast-luust sõdurite asemel lahingusse, pole tänapäeva ühiskonnas robotid veel nii laia haaret saavutanud. Hetkel on robotid siiski küllaltki kallihinnalised ja väga valdkonnaspetsiifilised masinad, mille kasutamist saavad endale lubada vaid suuremad ja rikkamad institutsioonid. Need on ka põhjused, miks täiesti tavalise inimese jaoks on robot praegu veel midagi väga erakordset ja vaimustavat. Viimaste aastate arenguid vaadates võib aga kindlalt öelda, et inimesi robotitest lahutav piir hakkab üha hõredamaks jääma.

Kui seni teenis robootika põhiliselt teadlaste, töösturite ja sõjaväelaste huve, siis nüüd on võetud suunaks tuua robotid lähemale ka tavainimesele. Esimesed sammud selle eesmärgi saavutamiseks on juba tehtud ja nii on poodides saadaval roboteid, mis on mõeldud näiteks põrandapindade koristamiseks või muru niitmiseks. Paraku on nende robotite hinnad veel küllaltki kõrged, seega kuuluvad need seadmed jätkuvalt eksootilisema kodutehnika kategooriasse [1].

Hoopis aktiivsemalt on roboteid hakatud kasutama aga haridusvaldkonnas. Juba 2007. aastast tegutseb Eestis Kooliroboti projekt [2], mille eesmärgiks on populariseerida õpilaste seas reaalaineid, muuta nende ainete õpetamine huvitavamaks ja tuua rohkem inimesi õppima inseneriteadusi. Viimane punkt on eriti tähtis seetõttu, et Eesti tööturul on juba pikemat aega valitsenud puudus inseneride järele, mille leevendamiseks käivitati Kooliroboti projektiga samal aastal ka projekt “Insener on looja”. Teadupoolest põhinevad inseneriteadused suuresti reaalainetel, mis kipuvad võrdluses humanitaarainetega olema keerulisemad ja nõuavad õppurilt rohkem vaeva ning pühendumist materjali mõistmisel. Tekkinud olukorrale ei mõju kuigi hästi ka praeguse haridussüsteemi orienteeritus faktipõhisele õppele: lapsed on sunnitud sageli pähe õppima väga suurt kogust informatsiooni, teadmata sealjuures, kas ja kuidas see reaalse eluga seondub ja millised on nende teadmiste rakendamisvõimalused. Praktilise töö vähese osakaalu tõttu ei teki õpilastel täit ettekujutust omandatavast materjalist, rääkimata motivatsioonist üldse midagi selgeks saada.

Kooliroboti projekti raames propageeritakse LEGO Mindstorms NXT robotikomplektide kaasamist erinevate reaalainete õppevahendite hulka. Projekti raames korraldatakse koolitusi õpetajatele ja huviringe õpilastele ning luuakse ja täiustatakse järjepidevalt eestikeelseid õppematerjale. Robotite soetamise kulud katab 50% (varasematel aastatel on osamäär ulatunud ka kuni 80%) ulatuses Tiigrihüppe Sihtasutus[3], ülejäänud summa peab kool leidma oma vahenditest. Vaadates projektiga liitunud koolide hulka (viimastel andmetel ligi 120 kooli) näib, et robotite hankimine on õppeasutuste jaoks siiski üsnagi jõukohane ülesanne.

Robotitel on mitmeid trumpe. Tegemist on väga mitmekülgse õppevahendiga, mille abil on võimalik muuta õpetamise protsessi huvitavamaks ja tuua õpilasteni materjali sisu selgemalt ja ilmekamalt. Kuigi kõige otsesemat rakendust leiavad LEGO robotid informaatika tundides, aidates õpilastel mõista programmeerimise alustõdesid, võimaldavad LEGO Mindstorms NXT komplektidesse kuuluvad andurid kasutada roboteid ka näiteks keemia- ja füüsikatundides. Lisaks põhikomplektis olevatele anduritele on võimalik juurde soetada tervet rida erinevaid andureid, mida toodavad kolmandad osapooled, kusjuures lisade valik on väga lai ja nii on võimalik hankida sensoreid väga spetsiifiliste ülesannete lahendamiseks - näiteks võib tuua temperatuuri-, vererõhu- või hoopis GPS-andurid.

Lisaks oma mitmekesisusele on robotid ka väga köitvad õppevahendid. Ei saa mööda vaadata asjaolust, et robotid kuuluvad hetkel jätkuvalt eksootiliste vidinate valdkonda ning äratavad seetõttu õpilastes uudishimu ja soovi proovida robotit “käsutada”. Robot on väga interaktiivne õppevahend, mis võimaldab õpilasel näha reaalajas oma tegevuse tulemusi ning seeläbi ka efektiivselt oma töös vigu otsida või täiendusi lisada ning katsetada erinevate töövõtetega. Tänu eespool loetletud omadustele aitavad robotid lisaks õppeainetekesksetele teadmistele arendada ka õpilaste loogilist mõtlemist ja analüüsivõimet, mis on koolis õppimise seisukohalt väga olulised oskused.

LEGO Mindstorms NXT robotite programmeerimiseks on olemas mitmeid erinevaid programmeerimisvahendeid - programmeerimiskeeli - mis erinevad üksteisest nii omaduste,

süntaksi kui ka kasutamise keerukuse poolest. Käesolev töö keskendub NXC (*Not eXactly C*) keelele, pakkudes üldhariduskoolide õpilastele ja õpetajatele õppematerjale selle keelega tutvumiseks.

Üheks selle bakalaureusetöö eesmärgiks on luua eestikeelne dokumentatsioon NXC enimkasutatavatele funktsioonidele. Lisaks tutvustatakse lugejale ka NXC keelt üldiselt ning pakutakse õpetusi selles keeles programmeerimisega alustamiseks. Teoreetilise osa kõrval on oma koht ka praktilistel ülesannetel, mis aitavad lugejal kirjutada oma esimesed programmid NXC keeles.

Käesolev bakalaureusetöö on kolmeosaline:

- Esimene osa annab lugejale põhjaliku ülevaate LEGO Mindstorms NXT robotitest ja nende programmeerimisest. Lisaks tehakse sissejuhatus NXC keelde, kirjeldades keele omadusi ja süntaksit.
- Teises osas pakutakse lugejale valikut enimkasutatavaid NXC funktsioone. Põhjalikke funktsioonide kirjeldusi saadavad ka aktuaalsed kasutusnäited.
- Töö kolmandast osast leiab lugeja harjutusülesandeid roboti programmeerimise peale. Ülesannete lahendamisel eeldatakse töö eelnevates osades omandatud teadmiste rakendamist.

Kuna antud töö sihtgrupiks on õppejõudude ja kaastudengite kõrval ka põhikooli ja gümnaasiumiastme õpilased ja õpetajad, on kirjutamisel püütud vältida liigkeerulist terminoloogiat ning seletustes jääda lihtsaks ja konkreetseks.

1. Ülevaade robotitest ja programmeerimisvahenditest

LEGO Mindstorms NXT robotite programmeerimine on väga lai valdkond ning enne spetsiifilisemate teemade juurde asumist on mõistlik tutvuda selle valdkonna alustega: mida kujutavad endast LEGO robotid ning kuidas toimub nende robotite programmeerimine.

1.1 LEGO Mindstorms NXT

LEGO Mindstorms NXT on mänguasjatootja LEGO poolt väljatöötatud programmeeritav robootikakomplekt. Komplekt jõudis müüki 2006 aasta keskpaiku ning alates sellest ajast on olnud pidevas arengus, võites endale järjest rohkem kasutajaid ja kasutusalasid. Komplektis on terve hulk traditsioonilisi LEGO ehitusklotse, millest saab konstrueerida roboti "keha", kuid lisaks neile on seal palju detaile, mida tavalisest klotsikarbist ei leia:

NXT põhiplokk (NXT Intelligent Brick)

NXT põhiplokk (vaata Joonis 1.) on LEGO roboti "aju" - miniatuurne arvuti, milles saab käivitada eelnevalt valmis kirjutatud ja NXT plokki laaditud programme. NXT põhiploki külge on võimalik ühendada kuni 3 mootorit ja kuni 4 andurit, mida juhitakse vastavalt programmides antud korraldustele. Lisaks arvutile on selle komponendi küljes veel LCD ekraan (suurusega 100 x 64 pikslit) ja kõlar kasutajale tagasiside väljastamiseks ning nupud roboti menüüs navigeerimiseks. Programme saab robotisse laadida USB või Bluetooth ühenduse vahendusel. Robot saab toidet Li-Ion akult (komplekti õppeversioonis) või patareidelt (laiatarbe versioonis). Toiteallikas kinnitub samuti NXT ploki külge.

Joonis 1. LEGO Mindstorms NXT põhiplokk. [12]

Mootorid

Komplekti kuulub 3 servomootorit, mis võimaldavad robotit liikuma panna. Servomootorite pöörlemist on võimalik juhtida ühe kraadi täpsusega, kuid reaalsuses mõjutab täpsust ka pind, millel robot liigub. Mootorite ühendamiseks on NXT põhiplokil väljundipesad A, B ja C.

Andurid

LEGO Mindstorms 2.0 komplekti kuulub 4 andurit: heli-, puute-, valguse ja ultraheliandur. Anduritega on võimalik teostada erinevaid mõõtmisi ja lähtuvalt tulemustest anda robotile korraldusi tegutsemiseks. Andurite ühendamiseks on robotil 4 sisendi pesa: 1, 2, 3 ja 4.

Ühenduskaablid

Andurite ja mootorite ühendamiseks NXT põhiploki külge kasutatakse RJ12 kaableid.

Ülaltoodud detailide abil on võimalik ehitada erineva konstruktsiooniga roboteid ning kasutada neid paljude erinevate ülesannete lahendamiseks. Lisaks põhidetailidele on võimalik juurde soetada tervet rida erinevaid andureid, mis avardavad roboti võimalusi veelgi.

1.2 Uus põlvkond

2013. aasta 4. jaanuaril kuulutas LEGO välja uue põlvkonna EV3 tootekomplekti, mis tõotab kaasa tuua palju uusi võimalusi, nagu roboti häälkäsklustega juhtimine või roboti Internetiga ühendamine, aga ka paremat riistvara - senisest võimsama põhiploki ja täiustatud andurid. Enamus NXT platvormi tarvikuid (andurid, ehitusklotsid jm) on ühilduvad ka uue EV3 platvormiga. Erandiks on põhiploki aku, mis on EV3 komplektis erinev. Uued komplektid peaksid müüki saabuma käesoleva aasta sügisel. [4][5]

1.2.1 NXT vs EV3

Järgnevas tabelis on toodud üldine võrdlus LEGO Mindstorms 2.0 NXT ja EV3 versioonide vahel:

	NXT	EV3
Roboti koostamise arhitektuur	LEGO Technic	LEGO Technic
Tarkvara	LABVIEW-l põhinev graafiline keskkond.	LABVIEW-l põhinev graafiline keskkond.
Riistvara	Põhiplokk, 5 andurit, 3 mootorit, sadu ehitusklotse.	Põhiplokk, 5 andurit, 3 mootorit, sadu ehitusklotse.
Õppematerjalid	Sadade tundide ulatuses õppekavasid põhi- ja keskkoolile, lisaks Interneti kogukonna poolt väljatöötatud veebilehed ja raamatud.	30+ tunni ulatuses STEM (Teadus, Tehnika, Inseneriteadused ja Matemaatika) õppekava (saadaval alates 2013. a sügissemestrist).
Tugiteenused	1 aasta garantiid, piiramatu kestvusega kasutajatugi.	1 aasta garantiid, piiramatu kestvusega kasutajatugi.
Saadavus	Müügil kuni 2015. a lõpuni.	Müügil alates 2013. a sügisest.
Hind	Õppeklassi lahendused hinnaga alates $4,000	Õppeklassi lahenduste hinnad algusega alla $5,000
Keskmine kasutamise kulu õpilase kohta 7-aastase kasutusaja jooksul	Alla $3	Alla $3

Tabel1: NXT ja EV3 platvormide üldine võrdlus [5]

Eelneva tabeli analüüsimise järel võib lugejal tekkida küsimus: miks peaks eelistama uut komplekti vanale, kui kahe vaadeldava põlvkonna vaheline erinevus on üsna väike? Lisaks sellele maksab EV3 lahendus rohkem kui tema eelkäija NXT...

Vastus sellele küsimusele peitub NXT ja EV3 põlvkondade tehnilises võrdluses, kuid tähelepanuta ei saa jätta ka EV3 arendamise tagamaid.

EV3 platvormi väljatöötamisel on arvestatud eelmise robotite põlvkonna kasutuskogemusega: LEGO on kogunud tagasisidet enam kui 800-lt õpetajalt üle maailma, selgitamaks välja, millised on võimalused robotite ja õppematerjalide paremaks muutmiseks. Uurimustöö tulemusi on selgelt näha uue platvormi võimalustes: [7]

EV3 platvorm on spetsiaalselt välja töötatud kasutamiseks reaalainete õpetamisel, aga ka õpilase loova mõtlemise, probleemide lahendamise oskuse ja meeskonnatöö võimekuse arendamiseks.

Mitmekesisemad ja interaktiivsemad õppematerjalid: kohaldatav õppekava, uued digitaalsed õppevahendid, sealhulgas töövihik õpilase edenemise jäädvustamiseks ja hindamise hõlbustamiseks.

Uued õppekavad on lähendatud reaalsele elule: õppetöö tegevused on tihedalt seotud teadus- ja tööstussektoris toimuvate protsessidega. Robotite ehitamisel ja programmeerimisel võetakse eeskujuks reaalselt kasutatavaid professionaalseid roboteid.

Roboti töövalmis seadmine on optimeeritud nii, et 45-minutilise ainetunni jooksul oleks õpilastel võimalik ehitada funktsioneeriv robot ja asuda seda programmeerima.

Lisaks õppematerjalidele ja -kavadele on põhjaliku uuenduskuuri läbinud ka komplektis olev riistvara. Järgnevas tabelis võrreldakse NXT ja EV3 platvormide riistvaralisi võimalusi:

	NXT	EV3
Ühenduvus Apple mobiilsete seadmetega	Ei	Jah
Häälkäsklustega rakenduste juhtimine	Ei	Jah
USB vahendusel roboti Internetiga ühendamine	Ei	Jah
Nuppude taustavalgus	Ei	Jah, 3 värvi
Roboti kõlar	Algelise kvaliteediga heli	Kõrgema kvaliteediga heli

Mootorite pesad	3	4
Põhiploki mälu	256 KB	16 MB
Põhiploki ekraani lahutusvõime	100x64 pikslit	178x128 pikslit
USB pesa lisade ühendamiseks	Ei	Jah
Infrapuna sensor	Ei	Jah
Infrapuna majakas	Ei	Jah
SD kaardi lugeja	Ei	Jah
Protsessor	ARM7	ARM9
Värvisensori tundlikkus	6 värvi	7 värvi + värvide puudumine, režiim taustavalguse eemaldamiseks
Andurite mõõtmiste tulemuste tagastamine	333 korda sekundis	1000 korda sekundis
Robotiga suhtlemine WiFi vahendusel	Ei	Jah, kuid vajalik lisaseade, mis ei kuulu komplekti

Tabel2: NXT ja EV3 riistvara võrdlus [6]

Tabelis esitatud andmete põhjal võib julgelt öelda, et uue platvormi tulekuga avaneb õpilastele mitmeid uusi ja kaasaegseid võimalusi roboti kasutamiseks: suurem mälumaht võimaldab kirjutada pikemaid programme, täiustatud andurid muudavad mõõtmised usaldusväärsemaks ning lisandub võimalus salvestada andmeid mälukaardile või laadida need Internetti.

1.3 Programmeerimisvahendid

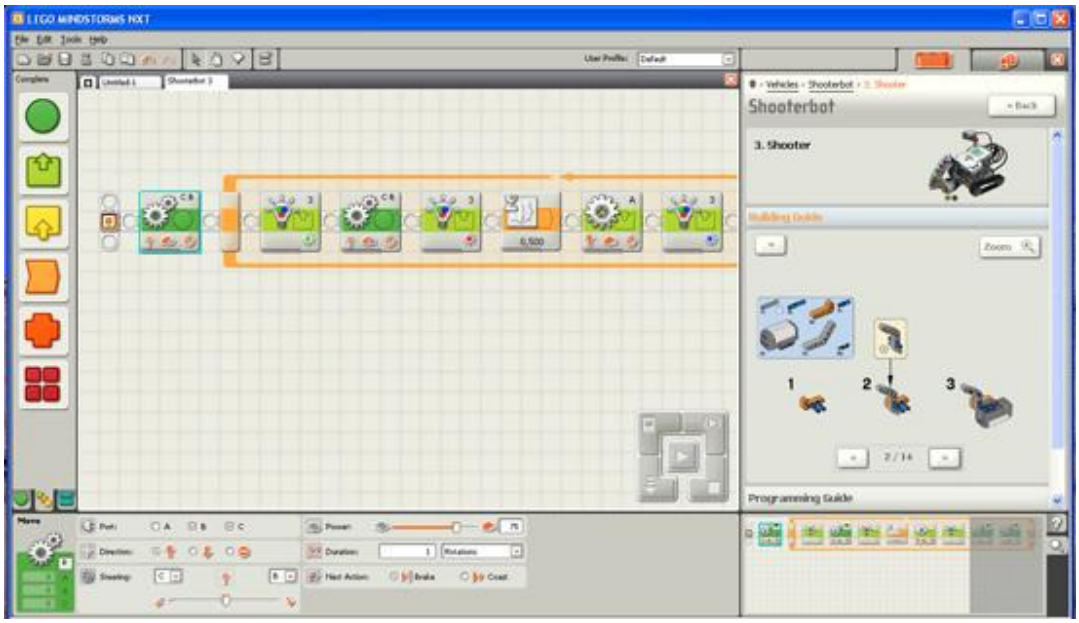
Selleks, et robot soovitud viisil tegutsema panna, tuleb kirjutada vastavasisuline programm ja laadida see roboti põhiplokki. Mindstorms NXT programmide kirjutamiseks on saadaval mitmeid erinevaid programmeerimiskeskkondi. Laias laastus võib need jagada kaheks: graafilised ja tekstipõhised programmeerimiskeskkonnad.

1.3.1 Graafilised programmeerimiskeskkonnad

Kui üldjuhul toimub programmeerimine teksti kujul, s.t. inimene kirjeldab programmi kirjutades käsklusi vastavalt programmeerimiskeele süntaksireeglitele tekstifaili sisse, siis graafilistes keskkondades asendavad teksti suures osas ikoonid ja pildikesed, mida saab tõsta selleks eraldatud alale, järjestades ja sidudes neid programmiks. Teksti kirjutamine piirdub sellisel juhul lahtritesse parameetritele väärtuste või pealkirjade kirjutamisega.

Selline lähenemine sobib hästi inimestele, kes pole varem programmeerimisega kokku puutunud: programmide koostamine ja ka lugemine on märkimisväärselt lihtsustatud, kuna visuaalset materjali on inimesel mugavam vastu võtta.

Üks selliseid programmeerimiskeskkondi on LEGO EDU NXT (vt. joonis 2).



Joonis 2. LEGO EDU NXT ekraanivaade [13]

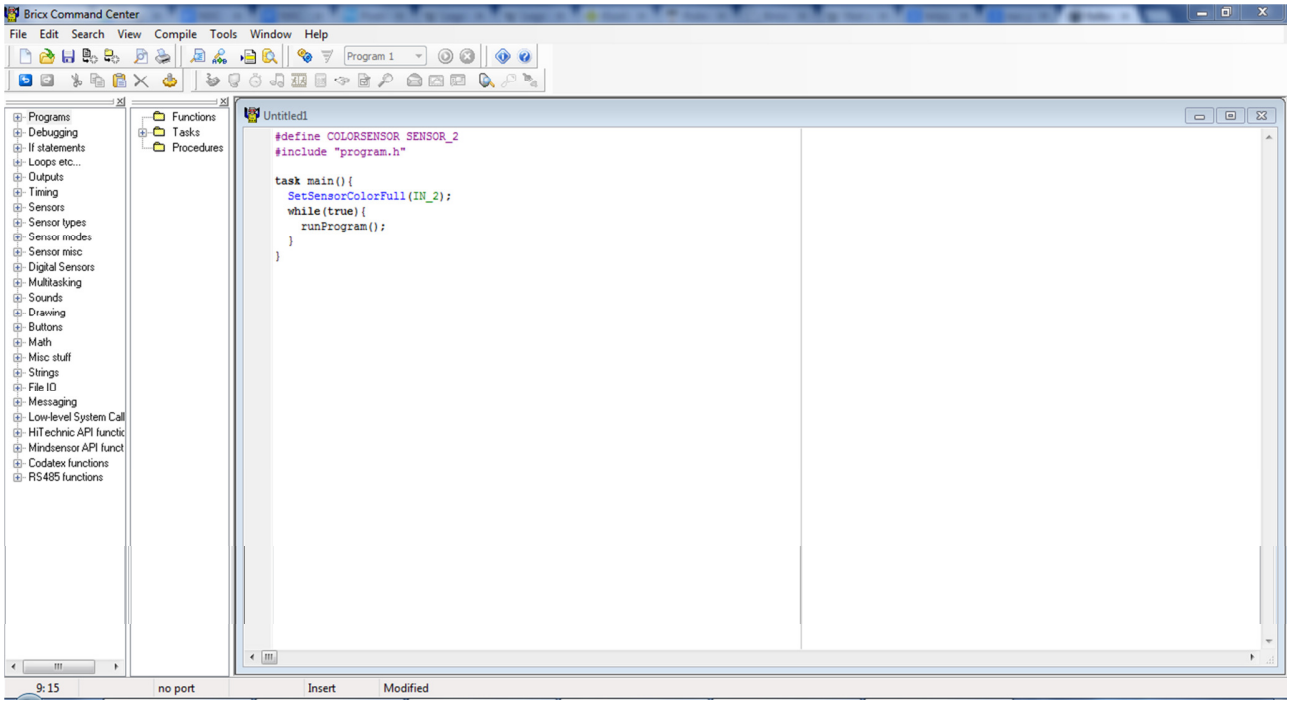
LEGO EDU NXT on LEGO ametlik arenduskeskkond, mis on loodud National Instruments LabVIEW programmi baasil ning programmeerimine toimub NXT-G keeles. Keskkonnas on kasutajal võimalik valida erinevaid tegevuste plokke ja lohistada neid tööalale, ühendades omavahel juhtmega. Valikus on näiteks mootorite ja andurite juhtimise, teksti kuvamise ja heli esitamise plokke aga ka tsükli ja tingimuslausete plokke edasijõudnute jaoks. Kuigi esmapilgul võib töövahendite valik tunduda üsna napp ja algeline, siis tegelikkuses annab LEGO EDU NXT keskkonnas kirjutada ka üsna keerulisi programme.

LEGO EDU NXT keskkond on lihtsasti paigaldatav ja kasutatav Windows ja Mac OS operatsioonisüsteemide all. Keskkonnas on olemas ka kompilaator, mis töötleb kasutaja koostatud programmi roboti jaoks "söödavale" kujule.

1.3.2 Tekstipõhised programmeerimiskeskkonnad

Kui graafilised programmeerimiskeskkonnad sobivad hästi algajatele programmeerimise põhimõtete õpetamiseks, siis järgmine samm edasi on üleminek tekstipõhisele programmeerimisele. Erinevalt NXT-G-st on paljud tekstipõhised programmeerimiskeeled (Java, Python, C, Ruby jm) kasutatavad lisaks robotite programmeerimisele ka "päris" tarkvara loomisel. Õpilasel on võimalik valida temale huvipakkuv keel, paigaldada vastav arenduskeskkond ning alustada keele süntaksi ja võimalustega tutvumist LEGO robotit programmeerides. Hiljem on võimalik omandatud teadmisi rakendada ka kõikvõimalike muude programmide kirjutamisel. Lisaks kujuneb õpilasel parem ettekujutus programmeerimisest üldiselt ning ka teiste tekstipõhiste programmeerimiskeelte iseseisev omandamine muutub lihtsamaks.

Kuna käesolev töö keskendub NXC keelele, siis tutvustaksin lühidalt BricxCC (Bricx Command Center) keskkonda, mis võimaldab kirjutada programme LEGO Mindstorms robotite jaoks kasutades C-le sarnanevat keelt.



Joonis 3. Bricx Command Center ekraanivaade

Bricx Command Center on arenduskeskkond, mis on loodud NXC (Not eXactly C), NBC (Next Byte Codes) ja NQC (Not Quite C) keeltes LEGO Mindstorms robotite programmeerimiseks. Keskkond on kirjutatud John Hanseni poolt Delphi keeles ning see on vabalt saadaval allalaadimiseks aadressil: http://bricxcc.sourceforge.net/. Samast on võimalik alla laadida ka rakenduse lähtekood. Hetkel on Bricx Command Center kasutatav Windows operatsioonisüsteemis, kuid arendamisel on ka Linuxi versioon.

Bricx Command Center võimaldab luua koodifaile ja kompileerida neid robotis käivitamisvalmis programmideks. Koodi kirjutamise hõlbustamiseks on võimalik kasutada suurt hulka erinevaid malle (nt. tingimuslausete koostamiseks või anduritelt näidu saamiseks). Mallid lihtsustavad oluliselt koodi kirjutamist algajate jaoks, kuna kaob vajadus meelde jätta erinevaid keelekonstruktsioone täpset süntaksit ning väheneb ka risk eksida süntaksi vastu [8].

Üheks Bricx Command Center eeliseks teiste arenduskeskkondade ees on ka asjaolu, et selles keskkonnas kirjutatud programmide käivitamiseks roboti peal ei ole vaja asendada roboti püsivara - see tähendab, et kasutaja saab asuda programmeerima tehaselise seadistusega robotit.

Bricx Command Center sobib ka uue põlvkonna EV3 robotite programmeerimiseks.

1.4 NXC

NXC (Not eXactly C) on John Hanseni poolt loodud programmeerimiskeel, mis on mõeldud LEGO Mindstorms NXT robotite programmeerimiseks. Kuigi keel on süntaksi poolest väga sarnane C keelega, ei ole NXC siiski mõeldud laiaotstarbeliseks kasutamiseks. NXC on loogiliselt defineeritud kahe eraldi osana: NXC keel kirjeldab ära süntaksi, mida tuleb järgida programmide kirjutamisel ja NXC API (Application Programming Interface - programmiliides) kirjeldab ära süsteemi funktsioonid, konstantid ja makrod, mida saab kasutada programmide kirjutamisel. Järgnevalt vaatleme NXC süntaksit ja põhilisi programmeerimisvõtteid selles keeles programmide kirjutamiseks. Koodikonstruktsioonide näited on märgitud rohelise värviga.

1.4.1 Programmi koodi kirjutamine NXC keeles

Kirjutamine

NXC on tõstutundlik keel - see tähendab, et keel teeb vahet suurtel ja väikestel tähtedel muutujate nimedes. Muutuja “abc” ja muutuja “Abc” on seega vaatamata sarnastele nimedele siiski kaks eri muutujat.

NXC-s ei ole reguleeritud tühikute kasutamine. Kasutaja võib eraldada võtmesõnu ja operaatoreid rohkem kui ühe tühikuga või mitte eraldada neid üldse. Seega omistamised `x = 1;` ja `x = 1;` on samaväärsed. Probleemid tühikute kasutamisega võivad ilmneda siis, kui kasutaja “tükeldab” ära operaatori, mis koosneb mitmest sümbolist - selliseid operaatoreid NXC-s tükeldada ei tohi. Heaks näiteks on paremnihke operaator (>>). Rida `x = 1 >> 4;` tähendab seda, et x väärtuseks seatakse 1 nihutatuna 4 biti võrra paremale. Rida `x = 1 > > 4`; annab aga süntaksivea.

Koodiread lõppevad NXC-s üldiselt semikooloniga (;). Erandiks on while- ja for-tsüklid ning ülesannete ja funktsioonide definitsioonid, kus lõpetatakse esimene koodirida looksuluga ({).

Koodi kommenteerimine

Koodi kommenteerimine NXC-s toimub sarnaselt C ja C++ keeltega: kommentaari kirjutamiseks tuleb kommentaari tekst teatud viisil ära märgistada. Ühe rea kommenteerimiseks võib kommentaari ette kirjutada kaks kaldkriipsu (//), mitme rea kommenteerimiseks tuleb teha vastavad märgised teksti algusesse (/*) ja lõppu (*/).

Näide:

```
/* Siin on
mitmerealine
kommentaar */
// Siin on kommentaar
```

Kommentaarid võimaldavad parandada koodi loetavust: koodiridadele saab inimestele arusaadavas keeles juurde märkida, mis tegevused antud real toimuvad ja mis on nende tegevuste eesmärk. Kompilaator ei kaasa neid programmifaili ning need ei mõjuta kuidagi roboti tegutsemist.

Arvulised konstandid

NXC võimaldab kasutajal kirja panna arve nii kümnend- kui kuueteistkümnendsüsteemis. Kümnendmurdude puhul kasutatakse komakoha eraldamisel punkti (.). Kuueteistkümnendsüsteemi arvud algavad kombinatsiooniga 0x (sobib ka 0X), millele järgneb üks kuni mitu numbrit.

Näiteid:

```
x = 10;     // x väärtuseks seatakse 10
x = 0x10; // x väärtus 16 (16-nd süsteemis)
x = 10.5  // x väärtus 10,5
```

Tekstilised konstandid

Sõnede (mitmest sümbolist koosnevad tekstiread) kirjapanekuks tuleb ümbritseda sõne sisu jutumärkidega (“ ”). Näide: `string tekst = “jutt jutt” ;`

Üksikute tähtede omistamiseks vastavale muutujale tuleb täht ümbritseda ühekordsete jutumärkidega (‘ ’): `char taht = ‘a’;`

1.4.2 Programmi struktuur

NXC-s kirjutatud programmide kood koosneb reeglina järgmistest osadest: muutujate deklaratsioonid ja koodiplokid. Koodiplokid jagunevad omakorda ülesanneteks ja funktsioonideks. Viimased on oma struktuuri poolest üsna sarnased.

Muutujad

Muutujad on programmi osad, mis võimaldavad olenevalt tüübist salvestada endasse erinevaid väärtusi. Muutujaid saab kasutada programmi töö juhtimiseks: neisse saab salvestada näiteks andurite näitusid või kontrollväärtusi ning seejärel tingimuslausete abil programmi töö vastavate tegevuste juurde suunata. Muutuja loomist nimetatakse muutuja deklareerimiseks. Muutuja deklareerimisel tuleb minimaalselt märkida muutuja tüüp (mis liiki väärtusi muutujasse salvestada saab) ja muutuja nimi. Soovi korral võib muutujale märkida kohe ka väärtuse, kuid seda saab teha ka hiljem. Näide muutuja deklareerimisest:

int taisarv = 15;

int - muutuja tüüp (täisarv tüüpi muutuja), taisarv - muutuja nimi, 15 - muutuja väärtus

Algajate programmeerimishuviliste jaoks võivad kasulikuks osutuda järgmised muutujate tüübid:

bool - tõeväärtus muutuja; võib omada väärtust “true” (tõene) või “false” (väär).

char - tähemärgi muutuja; võimaldab salvestada üksikut tähemärki sümboli või vastava ASCII koodi kujul.

int - täisarvu muutuja; väärtuseks võib olla täisarv vahemikust -32768 kuni 32767.

unsigned int - märgita täisarv, väärtused vahemikus 0 kuni 65535

float - kümnendarvu muutuja; võimaldab salvestada kümnendarve kuni 7 komakoha täpsusega

long - pikk täisarvu tüüpi muutuja väärtustega -2147483648 kuni 2147483647

unsigned long - märgita pikk täisarv, väärtusega 0 kuni 4294967296

byte - võimaldab salvestada märgita arve suuruses <=255.

string - sõne muutuja; võimaldab salvestada rida sümboleid (üht või mitut sõna)

Ülesanded (tasks)

Ülesande plokis on kasutajal võimalik kirjeldada tegevus (või tegevuste järjestus), mida saab siis robotile täitmiseks ette anda. Kuna NXT põhiploki arvuti toetab lõimtöötlust (programmi saab jagada mitmeks osaks - lõimeks - ja täita neid osi samaaegselt või vaheldumisi), vastab iga programmi ülesanne ühele NXT lõimele. Tulenevalt lõimtöötluse põhimõtetest saab ülesandeid täita samaaegselt või vaheldumisi ning käivitada või peatada nende täitmine vastavalt vajadusele. Näiteks on võimalik ühe ülesandega panna robot sõitma ja samal ajal teise ülesandega sõidu ajal muusikat esitama. Programmi kood ülaltoodud tegevuste läbiviimiseks on järgmine:

```
task muusika() {                          // ülesande "muusika" kirjeldamine
      while (true) {                            // tsükli loomine
            PlayTone(TONE_A4, MS_500);    // heli esitamise funktsioon
            Wait(MS_600);// ootamise funktsioon. MS_600 = 600 millisekundit
      }
}
task liikumine() {      // ülesande "liikumine" kirjeldamine
      while(true) {
            OnFwd(OUT_A, Random(100));    // mootorite käivitamise funktsioon
            Wait(Random(SEC_1));    /* ootamise aega saab ette anda ka
sekundites: SEC_1 on sama, mis MS_1000 */
      }
}

task juhtimine() {                              // ülesande "juhtimine"
kirjeldamine
      Wait(SEC_10);
      stop muusika;                             // ülesande "muusika"
seiskamine
      Wait(SEC_5);
      StopAllTasks();                     // kõigi ülesannete seiskamine
}

task main() {                                   // peaülesande kirjeldamine
```

```
/* “muusika”, “liikumine” ja “juhtimine” ülesannete käivitamine pärast
peaülesande käivitamist */
Precedes(muusika, liikumine, juhtimine);

}
```

Antud näidiskoodis defineeritakse 4 ülesannet: music, movement, controller ja main. Ülesande defineerimiseks tuleb kirjutada märksõna “task” (ülesanne), seejärel kirjutada ülesande nimi (antud juhul music, movement jm) ning lisada kohe nime järele sulupaar “()”. Ülesande alla kuuluvad tegevused (reeglina funktsioonid, väärtuste omistamised jm) kirjutatakse loogeliste sulgude “{ //kood }” vahele.

Näiteprogrammi täitmine toimub järgmiselt: esmalt käivitub ülesanne main, mis omakorda käivitab ülesanded muusika, liikumine ja juhtimine. Ülesanne muusika paneb roboti 600 millisekundilise vahega esitama 500 millisekundi pikkust helinooti. Ülesanne liikumine paneb roboti sirgjooneliselt sõitma. Ülesanne juhtimine ootab 10 sekundit, seejärel lõpetab ülesande muusika, ootab veel 5 sekundit ja lõpetab siis kõik ülesanded (main k.a). Sellega lõppeb programmi täitmine.

Programmi käivitades tuleb arvestada sellega, et ülesande deklareerimine ei tähenda automaatselt ka selle käivitamist. Esimesena käivitatakse alati peaülesanne main, mille sees saab käivitada juba kõiki teisi. Main-ülesanne peab igas programmis kindlasti olemas olema, vastasel juhul ei oska robot programmis kirjeldatud tegevusi läbi viia.

Funktsioonid

Funktsioonid on programmi korduvkasutatavad osad. Nagu ka ülesannete puhul, saab funktsioonidesse koondada tegevusi ning seejärel need sobival hetkel funktsiooni väljakutsumisega käivitada. Erinevalt ülesandest, ei saa funktsiooni täitmist peatada ja hiljem samast kohast jätkata. Funktsioon täidetakse algusest lõpuni, misjärel peab funktsioon tagastama kindlat tüüpi väärtuse. Funktsioonidele on ka võimalik (kuid ei ole kohustuslik) ette anda parameetreid - väärtusi, mida funktsioon oma töös kasutab. Näide funktsiooni defineerimisest:

```
int summa(int x, int y)
{
      return x+y;
}
```

Esimesel real kirjeldatakse esmalt ära funktsiooni tagastusväärtuse tüüp (int ehk täisarv). Sellele järgneb funktsiooni nimi (funktsioon1) ning sulgudes tuuakse ära parameetrid (int x, int y), mida funktsioon töötamiseks vajab.
Märksõnaga return tagastatakse töö tulemus - funktsiooni definitsioonis määratud tüüpi (int) väärtus (antud juhul argumentide x ja y summa). Seega kui funktsiooni tüübiks on määratud int ja soovitakse return käsu abil tagastada näiteks float tüüpi väärtust, annab kompilaator veateate. Kui soovitakse kirjutada funktsioon, mis viib läbi teatud tegevused, kuid ei tagasta mingit konkreetset väärtust, tuleb funktsiooni tüübiks määrata void (tühi). Funktsiooni töö lõpetamiseks tuleb sel juhul sobivasse kohta kirjutada märksõna return ilma järgneva väärtuseta.

Valmisfunktsioonid

Nii, nagu paljudes laialtkasutatavates programmeerimiskeeltes, on ka NXC-s saadaval suur hulk valmisfunktsioone. Valmisfunktsioonid on keele looja(te) poolt kirjutatud funktsioonid, mis aitavad kasutajal juhtida roboti riistvara - mootoreid ja andureid - väljastada heli või kuvada teksti roboti LCD ekraanile. Kõnealused funktsioonid on koondatud spetsiaalsesse teeki ning grupeeritud vastavalt otstarbele ja kasutusalale moodulitesse. Käesoleva töö järgmine osa keskendubki valmisfunktsioonide ja nende kasutusvõimaluste kirjeldamisele.

2. NXC funktsioonid

NXC valmisfunktsioonid on vahekihiks kasutaja ja roboti NXT põhiploki püsivara vahel. Püsivara on roboti mälusse salvestatud tarkvara, mida kasutatakse roboti komponentide (mootorid, andurid, aga ka põhiploki riistvara) juhtimiseks. Kui kasutaja kasutab koodis mõnda NXC funktsiooni, pöördub see funktsioon vastava püsivara funktsiooni poole ning viimane omakorda annab käskluse vastavale roboti riistavara komponendile. Lihtsuse mõttes võib öelda, et NXC funktsioonid juhivad püsivara vahendusel roboti erinevaid seadmeid.

NXC keele üheks eeliseks on ühilduvus LEGO Mindstorms robotite standardse püsivaraga. See tähendab, et NXC-s kirjutatud programme on võimalik laadida robotisse ning seal ka koheselt käivitada. Võrdluseks: Java keeles LEGO robotit programmeerides tuleb asendada roboti tehase püsivara leJOS NXJ püsivaraga, mis sisaldab endas Javas kirjutatud programmide käivitamiseks vajalikku Java virtuaalmasinat.

Järgnevalt tutvume lähemalt NXC funktsioonide moodulite ja nende sisuga. Funktsioonide kirjeldused pärinevad NXC ametlikust dokumentatsioonist. [9]

2.1 Moodulid

NXC funktsioonid jagunevad oma otstarbe alusel moodulitesse. Ühe mooduli funktsioonid tegelevad mingi kindla valdkonnaga, näiteks sisendandmete lugemine või NXT põhiploki nuppude kasutamine. NXC pakub kasutajale 11 püsivara juhtimisega seotud moodulit:

Nuppude moodul (*Button module*) - funktsioonid NXT põhiploki 4 nupu kasutamiseks.

Andmeside moodul (*Comm module*) - roboti ja teiste seadmete (ka robotite) vahelise suhtluse korraldamine USB või Sinihamba tehnoloogia vahendusel.

Juhtimise moodul (*Command module*) - funktsioonid programmi täitmise juhtimiseks.

Ekraani moodul (*Display module*) - teksti kirjutamine või kujundite joonistamine roboti põhiploki LCD-ekraanile.

Protsessori moodul (*IOCtrl module*) - roboti väljalülitamine või “magama panemine”.

Sisendi moodul (*Input module*) - anduritelt sisendandmete lugemine.

Failihalduse moodul (*Loader module*) - funktsioonid failidega töötamiseks (loomine, kustutamine, failide avamine ja neisse andmete kirjutamine).
Lowspeed protokolli moodul (*Lowspeed module*) - roboti ja sisendseadmete vahelise suhtluse korraldamine i2c protokolli kasutades. Kasutatakse kolmandate osapoolte anduritega suhtlemisel.
Mootorite moodul (*Output module*) - funktsioonid roboti mootorite juhtimiseks.
Heli moodul (*Sound module*) - funktsioonid helinootide ja -failide esitamiseks.
Kasutajaliidese moodul (*Ui module*) - funktsioonid NXT kasutajaliidesega suhtlemiseks, näiteks helitugevuse või aku näidu lugemine.

Lisaks roboti püsivara juhtimiseks mõeldud funktsioonide moodulitele on NXC-s olemas ka programmiliidese (ingl. k. API - Application Programming Interface) moodul. API-moodulisse on koondatud keelekesksed funktsioonid, mida ei kasutata otseselt roboti juhtimiseks, vaid pigem programmi kirjutamise hõlbustamiseks. Selliste funktsioonide abil on näiteks võimalik töödelda muutujaid või teostada keerulisemaid matemaatilisi operatsioone (astmesse võtmine, ruutjuure leidmine jms).

2.2 Moodulite dokumentatsioon

Järgnev töö osa keskendub enimkasutatavate NXC funktsioonide kirjeldamisele. Valikusse kuuluvad eelkõige funktsioonid, mida võib algajal programmeerimishuvilisel NXC keele õppimisel ja programmeerimise harjutamisel vaja minna.

2.2.1 Nuppude moodul

bool ButtonPressed(const byte btn, bool resetCount=false)
Kirjeldus: funktsioon fikseerib kasutaja poolt etteantud nupu vajutamise. Soovi korral võimalik nullida nupuvajutuste arvu loendur.
Parameetrid:
btn - nupu nimi, võimalik valida järgmiste konstantide seast:

- BTN1 või BTNEXIT - nupp “välja/katkesta”
- BTN2 või BTNRIGHT - nupp “nool paremale”
- BTN3 või BTNLEFT - nupp “nool vasakule”
- BTN4 või BTNCENTER - nupp “OK/kinnita”

resetCount - tõeväärtustüüpi argument, mille väärtus määrab, kas nupuvajutuste loendur nullitakse või mitte. Võimalikud väärtused:

- false - loendurit ei nullita
- true - loendur nullitakse

Tagastusväärtus: Funktsioon tagastab väärtuse true (tõene) kui nupp vajutatakse alla ning väärtuse false (väär) kui nupuvajutust ei toimu.

Näide kasutamisest:

```
while (ButtonPressed(BTNCENTER, false)) {
      PlayTone(440, 1000);
}
```

Selgitus näitele: programmis on kirjeldatud while-tsükkel. Tsükli täitmine toimub nii kaua, kuni funktsioon ButtonPressed tagastab tõest väärtust (ehk nupp - antud juhul OK-nupp - on alla vajutatud. Tsükli käigus mängitakse kasutajale funktsiooni PlayTone vahendusel ühe sekundi jooksul heli. Kui OK-nupp vabastatakse, tagastab funktsioon ButtonPressed väära väärtuse ja while-tsükli kordamine lõpetatakse. Vajutuste loendurit ei nullita.

byte ButtonCount(const byte btn, bool resetCount=false)

Kirjeldus*:* funktsioon tagastab arvu, mitu korda on kasutaja poolt ette antud nuppu vajutatud. Vajadusel saab loenduri nullida.

Parameetrid:

btn - nupu nimi, võimalik valida järgmiste konstantide seast:

- BTN1 või BTNEXIT - nupp “välja/katkesta”
- BTN2 või BTNRIGHT - nupp “nool paremale”
- BTN3 või BTNLEFT - nupp “nool vasakule”
- BTN4 või BTNCENTER - nupp “OK/kinnita”

resetCount - tõeväärtustüüpi argument, mille väärtus määrab, kas nupuvajutuste loendur nullitakse või mitte. Võimalikud väärtused:

- false - loendurit ei nullita
- true - loendur nullitakse

Tagastusväärtus*:* funktsioon tagastab arvulise väärtuse, mis võrdub antud nupu vajutamiste arvuga.

Näide kasutamisest:

```
vajutamised = ButtonCount(BTNEXIT, true);
```

Selgitus näitele: muutuja "vajutamised" väärtustamiseks kutsutakse välja ButtonCount funktsioon katkesta-nupu jaoks. Funktsioon tagastab nupu vajutamiste arvu ning kuna parameetri resetCount väärtuseks on seatud "true", nullitakse ka nupuvajutuste loendur. Muutuja "vajutamised" väärtuseks saab nupuvajutuste arv enne loenduri nullimist.

byte ButtonLongPressCount(const byte btn)

Kirjeldus: analoogne funktsiooniga ButtonCount, kuid puudub võimalus loendur nullida. Tagastab nupu pikkade vajutuste (> 2 sekundit) arvu. Detailsem info ButtonCount funktsiooni kirjeldusest.

byte ButtonShortReleaseCount(const byte btn) ja
byte ButtonLongReleaseCount(const byte btn)

Kirjeldus: funktsiooniga ButtonCount sarnanevad funktsioonid, mis tagastavad vastavalt nupu kiirete ja aeglaste vabastamiste arvu. Võimalus loendurit nullida puudub. Detailsem info ButtonCount funktsiooni kirjeldusest.

byte ButtonState(const byte btn)

Kirjeldus: funktsioon nupu oleku (alla vajutatud, vabastatud jms) tagastamiseks.

Parameetrid:

btn - nupu nimi, võimalik valida järgmiste konstantide seast:

- BTN1 või BTNEXIT - nupp "välja/katkesta"
- BTN2 või BTNRIGHT - nupp "nool paremale"
- BTN3 või BTNLEFT - nupp "nool vasakule"
- BTN4 või BTNCENTER - nupp "OK/kinnita"

Tagastusväärtus: funktsioon tagastab ühe järgmistest konstantidest:

- BTNSTATE_PRESSED_EV - nupp on alla vajutatud
- BTNSTATE_SHORT_RELEASED_EV - nuppu on kiirelt vabastatud

- BTNSTATE_LONG_PRESSED_EV - nuppu hoitakse pikalt all (> 2 sekundit)
- BTNSTATE_LONG_RELEASED_EV - nuppu on sujuvalt vabastatud
- BTNSTATE_NONE - nupu vaikimisi olek (nuppu pole programmi töö käigus kasutatud)

Näide kasutamisest:

```
if (ButtonState(BTNCENTER) == BTNSTATE_LONG_PRESSED_EV)
      TextOut(0, LCD_LINE1, "Pikk vajutus", 0);
```

Selgitus näitele:

Programm kontrollib, kas OK-nupp on pikalt alla vajutatud olekus. Kui see tingimus on täidetud, väljastatakse roboti ekraanile TextOut funktsiooni abil kiri "Pikk vajutus".

void SetButtonPressCount(const byte btn, const byte n)

Kirjeldus: funktsioon võimaldab muuta nupu vajutamiste loenduri näitu.

Parameetrid:

btn - nupu nimi, võimalik valida järgmiste konstantide seast:

- BTN1 või BTNEXIT - nupp "välja/katkesta"
- BTN2 või BTNRIGHT - nupp "nool paremale"
- BTN3 või BTNLEFT - nupp "nool vasakule"
- BTN4 või BTNCENTER - nupp "OK/kinnita"

n - uus väärtus, mida soovitakse loendurile anda

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
SetButtonPressCount(BTNRIGHT, 0);
```

Selgitus näitele: funktsiooni SetButtonPressCount abil seatakse nupu "nool paremale" vajutamiste loenduri väärtuseks 0, ehk sisuliselt nullitakse loendur ära.

void SetButtonState(const byte btn, const byte state)

Kirjeldus: funktsioon võimaldab muuta nupu olekut.

Parameetrid:

btn - nupu nimi, võimalik valida järgmiste konstantide seast:

- BTN1 või BTNEXIT - nupp “välja/katkesta”
- BTN2 või BTNRIGHT - nupp “nool paremale”
- BTN3 või BTNLEFT - nupp “nool vasakule”
- BTN4 või BTNCENTER - nupp “OK/kinnita”

state - nupu oleku väärtus, võimalik valida järgmiste konstantide seast:

- BTNSTATE_PRESSED_EV - nupp on alla vajutatud
- BTNSTATE_SHORT_RELEASED_EV - nuppu on kiirelt vabastatud
- BTNSTATE_LONG_PRESSED_EV - nuppu hoitakse pikalt all (> 2 sekundit)
- BTNSTATE_LONG_RELEASED_EV - nuppu on sujuvalt vabastatud
- BTNSTATE_NONE - nupu vaikimisi olek (nuppu pole programmi töö käigus kasutatud)

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk “tühi”, ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
SetButtonState(BTNLEFT, BTNSTATE_NONE);
```

Selgitus näitele: funktsiooni SetButtonState abil seatakse nupu olekuks BTNSTATE_NONE ehk nupu algne vaikimisi olek.

2.2.2 Andmeside moodul

Selle mooduli funktsioonide kasutamine eeldab põhjalikumaid teadmisi NXC keeles programmeerimisest, seega käesoleva töö raames me neil pikemalt ei peatu.

2.2.3 Juhtimise moodul

void Wait(unsigned long ms)

Kirjeldus: funktsioon paneb käimasoleva ülesande etteantud ajaks magama.

Parameetrid:

ms - aeg (sekundites või millisekundites) ühes järgnevatest vormingutest:

- SEC_”number” - aeg sekundites
- MS_”number” - aeg millisekundites

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk “tühi”, ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task main() {
	PlaySound(SOUND_CLICK);
	Wait(SEC_1);
	PlaySound(SOUND_CLICK);
	Wait(MS_1000);
	PlaySound(SOUND_CLICK);
}
```

Selgitus näitele: defineeritud main-ülesandes mängitakse kolm korda 1-sekundilise vahega süsteemi klõpsatuse heli. Juhin tähelepanu asjaolule, et 1-sekundilist ajavahemikku saab kirja panna nii sekundites kui millisekundites.

void StopAllTasks()

Kirjeldus: funktsioon seiskab kõik ülesanded. Funktsiooni väljakutsele järgnevat koodi ei käivitata ning sisuliselt peatatakse ka programmi täitmine (kuid ei väljuta täitmise režiimist).

Parameetrid: funktsioonil puuduvad parameetrid.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task yl1() {
	// Ülesande tegevused
}
task yl2() {
	// Ülesande tegevused
}
task main() {
	Precedes(yl1, yl2);
	Wait(SEC_5);
	StopAllTasks();
}
```

Selgitus näitele: näites kirjeldatakse ära kaks ülesannet - yl1 ja yl2, ülesannetes sooritatavad tegevused pole antud näite kontekstis olulised. Ülesandes main käivitatakse kirjeldatud

ülesanded funktsiooniga Precedes. Seejärel ootab ülesanne main 5 sekundit, lubades ülesannetel yl1 ja yl2 töötada, misjärel peatab kõigi ülesannete täitmise funktsiooniga StopAllTasks.

void Stop(bool bvalue)

Kirjeldus: seiskab hetkel täidetava programmi. Sarnane funktsiooniga StopAllTasks, kuid erinevalt viimasest on funktsioonile Stop võimalik ette anda tingimus, mille täitmisel programmi täitmine peatatakse.

Parameetrid:

bvalue - tõeväärtustüüpi parameeter, võimalikud väärtused:

- false - programmi täitmist ei peatata
- true - programmi täitmine peatatakse

Näide kasutamisest ja selgitus: antud funktsiooni võib kasutada erinevalt. Üheks võimaluseks on funktsioonile kohe ette anda tõeväärtus "tõene" või "väär":

```
Stop(true); // Programmi täitmine peatatakse
Stop(false);      // Programmi täitmist ei peatata (selline kasutamine ei oma mõtet, kuna programmi täitmine jätkub ka ilma funktsiooni sedalaadi väljakutsumiseta)
```

Teine võimalus funktsiooni kasutamiseks on ette anda programmi täitmise peatamise tingimus ning kontrollida, kas see on täidetud:

```
Stop(2 == 3);     // Kaks ei võrdu kolmega, seega tingimus ei ole täidetud ja programmi täitmist ei peatata
Stop(x == 9);     // Programmi täitmine peatatakse, kui x väärtuseks on 9, vastasel juhul programmi täitmine jätkub
```

void ExitTo(task newTask)

Kirjeldus: peatab koheselt käesoleva ülesande täitmiste ja alustab parameetrina etteantud ülesande täitmist.

Parameetrid:

newTask - ülesande nimi, mida soovitakse käivitada pärast käesoleva ülesande lõpetamist

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task yl2();
task yl1() {
      PlayTone(TONE_C4, MS_500);
      ExitTo(yl2);
}
task yl2() {
      PlayTone(TONE_C6, MS_500);
      ExitTo(yl1);
}
task main() {
      precedes(yl2);
}
```

Selgitus näitele: Näites kirjeldatakse ära kaks ülesannet - yl1 ja yl2 - ning peaülesanne main. Ülesanded yl1 ja yl2 mängivad kasutajale ette ühe helinoodi ning suunavad seejärel töö teisele ülesandele. Ülesannet yl2 on defineeritud kaks korda - alguses tühjana ja seejärel juba ka tegevustega. Põhjuseks on see, et ülesande yl1 definitsioonis viidatakse ülesandele yl2, mis peab varem defineeritud olema, vastasel juhul toimuks viitamine olematule ülesandele.

void Precedes(task task1, task task2, ..., task taskN)

Kirjeldus: funktsioon võimaldab koostada nimekirja ülesannetest, mis käivitatakse pärast käesoleva ülesande täitmise lõpetamist. Funktsiooni tuleks kasutada ühe ülesande sees mitte rohkem kui korra ning soovitavalt kohe ülesande definitsiooni alguses. Nimekirjas olevad ülesanded käivitatakse samaaegselt (kui pole takistavaid tegureid) ning ülesannete arv nimekirjas ei ole piiratud.

Parameetrid:

task1,2,..,n - ülesannete nimed, mida soovitakse käivitada

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task yl1() {
      // Ülesande tegevused
```

```
}
task yl2() {
	// Ülesande tegevused
}
task main() {
	Precedes(yl1, yl2);
	PlayTone(TONE_C4, SEC_5)
}
```

Selgitus näitele: näiteprogrammis kirjeldatakse kaks ülesannet - yl1 ja yl2 - ning lisaks peaülesanne main. Funktsiooni Precedes abil määratakse main-ülesandes sellele järgnema ülesanded yl1 ja yl2. Programmi käivitamisel täidetakse esmalt main-ülesanne - mängitakse helinoot C 5 sekundi jooksul, seejärel ülesanne lõppeb ning asutakse ülesannete yl1 ja yl2 täitmise juurde.

void Follows(task task1, task task2, ..., task taskN)

Kirjeldus: funktsioon võimaldab koostada nimekirja ülesannetest, millele käesolev ülesanne järgneb. Käesolev ülesanne käivitatakse iga kord, kui mõni nimekirjas olev ülesanne on oma töö lõpetanud. Funktsiooni tuleks kasutada ühe ülesande sees mitte rohkem kui korra ning soovitavalt kohe ülesande definitsiooni alguses.

Parameetrid:

task1,2,..,n - ülesannete nimed, mille järel soovitakse antud ülesannet välja kutsuda.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task yl1() {
	Follows(main);
	// Ülesande tegevused
}
task main() {
	PlayTone(TONE_C4, SEC_5)
}
```

Selgitus näitele: Näiteprogrammis defineeritakse ülesanne yl1 ja peaülesanne main. Ülesande yl1 definitsioonis kirjeldatakse Follows funktsiooni abil, et see ülesanne järgneb main-ülesandele. Programmi töö käigus täidetakse esmalt peaülesanne, milles mängitakse 5 sekundi jooksul kasutajale C helinooti, ning seejärel kutsutakse välja ülesanne yl1 ning sooritatakse selles kirjeldatud tegevused.

void StartTask(task t)

Kirjeldus: funktsioon üksiku ülesande käivitamiseks.

Parameetrid:

t - käivitatava ülesande nimi

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task yl1() {
      // Ülesande tegevused
}
StartTask(yl1);
```

Selgitus näitele: esmalt defineeritakse ülesanne yl1 ning seejärel kutsutakse see StartTask funktsiooni abil see välja.

2.2.4 Ekraani moodul

Soovitus: kui on soov kuvada programmi töö lõpus roboti ekraanile mingisugust infot, tuleks pärast ekraani mooduli funktsiooni väljakutsumist panna funktsiooniga Wait programm mõneks ajaks ootele - vastasel juhul kuvatakse info vaid hetkeks ekraanile ning programmi töö lõppeb ekraani tühjendamisega.

void ClearScreen()

Kirjeldus: Funktsioon roboti ekraani tühjendamiseks - kogu ekraanil olev info kustutatakse ja tulemuseks on tühi ekraan.

Parameetrid: funktsioonil puuduvad parameetrid.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk “tühi”, ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task main() {
	TextOut(0, LCD_LINE1, “jutt”);
	Wait(SEC_5);
	ClearScreen();
}
```

Selgitus näitele: programmis defineeritakse peaülesanne main, mille sees väljastatakse esmalt funktsiooniga TextOut roboti ekraani esimesele reale tekst “jutt”, seejärel paneb funktsioon Wait roboti 5 sekundiks ootele, misjärel ekraan tühjendatakse funktsiooniga ClearScreen.

void ClearLine(byte line)

Kirjeldus: tühjendab ekraani etteantud rea.

Parameetrid:

line - ekraani rea, mida soovitakse tühjendada, tähis. Valida on kaheksa rea vahel, mida saab märkida kas märksõna või sellele vastava numbri abil:

- LCD_LINE1 või 56 - kõige ülemine rida
- LCD_LINE2 või 48
- LCD_LINE3 või 40
- LCD_LINE4 või 32
- LCD_LINE5 või 24
- LCD_LINE6 või 16
- LCD_LINE7 või 8
- LCD_LINE8 või 0 - kõige alumine rida

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk “tühi”, ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task main() {
	TextOut(0, LCD_LINE1, “Esimese rea jutt.”);
	TextOut(0, LCD_LINE3, “Kolmanda rea jutt.”);
	Wait(SEC_5);
```

```
    ClearLine(LCD_LINE1);
}
```

Selgitus näitele: programmis defineeritakse peaülesanne main, mille sees väljastatakse esmalt funktsiooniga TextOut roboti ekraani esimesele reale tekst "Esimese rea jutt.", siis kolmandale reale tekst "Kolmanda rea jutt." ning seejärel paneb funktsioon Wait roboti 5 sekundiks ootele, misjärel ekraani esimene rida tühjendatakse funktsiooniga ClearLine. Ekraanile jääb alles kolmandale reale kirjutatud tekst.

char TextOut(int x, int y, string str, unsigned long options=DRAW_OPT_NORMAL)

Kirjeldus: funktsioon teksti roboti ekraanile väljastamiseks.

Parameetrid:

x - teksti alguskoht tekstireal. Võimalik valida väärtusi vahemikus 0...99 (roboti ekraani laius on 100 pikslit)

y - ekraani rida, kuhu soovitakse teksti kuvada. Valida on kaheksa rea vahel, mida saab märkida kas märksõna või sellele vastava numbri abil:

- LCD_LINE1 või 56 - kõige ülemine rida
- LCD_LINE2 või 48
- LCD_LINE3 või 40
- LCD_LINE4 või 32
- LCD_LINE5 või 24
- LCD_LINE6 või 16
- LCD_LINE7 või 8
- LCD_LINE8 või 0 - kõige alumine rida

str - tekst, mida soovitakse ekraanil kuvada. Selle parameetri väärtus tuleb kirjutada jutumärkide vahele.

options - täiendavad seadistused teksti kuvamiseks. Selle võimaluse kasutamiseks on vajalik robotile paigaldada NXC jaoks optimeeritud ja täiustatud püsivara, seega selle parameetri võib lihtsalt sisestamata jätta.

Tagastusväärtus: funktsiooni töö tulemuseks on kasutaja poolt etteantud tekst, mis väljastatakse ekraanile.

Näide kasutamisest:

```
task main() {
      TextOut(0, 56, “Tervitused!”);
      TextOut(10, 48, “- Tere!”);
}
```

Selgitus näitele: programmis defineeritud peaülesandes kirjutatakse roboti ekraani esimesele ja teisele reale vastavalt tervitustekstid “Tervitused!” ja “- Tere!”. Teise rea tekst kirjutatakse sealjuures taandega, kuna teksti alguskohaks on määratud ekraanirea 10. piksel.

char NumOut(int x, int y, variant value, unsigned long options=DRAW_OPT_NORMAL)

Kirjeldus: funktsioon võimaldab väljastada ekraanile erinevat tüüpi (täisarvud, kümnendmurrud) arve.

Parameetrid:

x - arvu asukoht tekstireal.

y - ekraani rida, kuhu soovitakse teksti kuvada. Valida on kaheksa rea vahel, mida saab märkida kas märksõna või sellele vastava numbri abil:

- LCD_LINE1 või 56 - kõige ülemine rida
- LCD_LINE2 või 48
- LCD_LINE3 või 40
- LCD_LINE4 või 32
- LCD_LINE5 või 24
- LCD_LINE6 või 16
- LCD_LINE7 või 8
- LCD_LINE8 või 0 - kõige alumine rida

value - arv, mida soovitakse ekraanil kuvada. Kõnealune funktsioon toetab erinevat tüüpi arve.

options - täiendavad seadistused teksti kuvamiseks. Selle võimaluse kasutamiseks on vajalik robotile paigaldada NXC jaoks optimeeritud ja täiustatud püsivara, seega selle parameetri võib lihtsalt sisestamata jätta.

Tagastusväärtus: funktsiooni töö tulemuseks on kasutaja poolt etteantud arv, mis väljastatakse ekraanile.

Näide kasutamisest:

```
x = 4;
NumOut(0, LCD_LINE3, x);
```

Selgitus näitele: antud programmilõigus defineeritakse esmalt täisarvtüüpi muutuja x ja antakse talle väärtuseks 4. Seejärel funktsiooni NumOut abil kirjutatakse muutuja x väärtus (4) ekraani kolmanda rea algusesse.

char LineOut(int x_1, int y_1, int x_2, int y_2, unsigned long options=DRAW_OPT_NORMAL)

Kirjeldus: funktsioon väljastab roboti ekraanile sirge joone punktidest x_1,y_1 kuni x_2,y_2.

Parameetrid:

x_1 - esimese punkti koordinaat ekraani x-teljel. Väärtused vahemikus 0..99

y_1 - esimese punkti koordinaat ekraani y-teljel. Väärtused vahemikus 0..63

x_2 - teise punkti koordinaat ekraani x-teljel. Väärtused vahemikus 0..99

y_2 - teise punkti koordinaat ekraani y-teljel. Väärtused vahemikus 0..63

options - täiendavad seadistused teksti kuvamiseks. Selle võimaluse kasutamiseks on vajalik robotile paigaldada NXC jaoks optimeeritud ja täiustatud püsivara, seega selle parameetri võib lihtsalt sisestamata jätta.

Tagastusväärtus: funktsiooni töö tulemuseks on sirge joon, mis väljastatakse ekraanile vastavalt kasutaja etteantud koordinaatidele.

Näide kasutamisest:

```
LineOut(0, 32, 99, 32);
```

Selgitus näitele: funktsioon LineOut joonistab ekraani keskele ekraani vasakust äärest parema ääreni ulatuva horisontaalse joone.

char CircleOut(int x, int y, byte radius, unsigned long options=DRAW_OPT_NORMAL)

Kirjeldus: funktsioon roboti ekraanile ringi joonistamiseks.

Parameetrid:

x - ringi keskpunkti koordinaat ekraani x-teljel.

y - ringi keskpunkti koordinaat ekraani y-teljel.

radius - ringi raadius.

options - täiendavad seadistused teksti kuvamiseks. Selle võimaluse kasutamiseks on vajalik robotile paigaldada NXC jaoks optimeeritud ja täiustatud püsivara, seega selle parameetri võib lihtsalt sisestamata jätta.

Tagastusväärtus: funktsiooni töö tulemuseks on vastavalt kasutaja määratud parameetritele joonistatud ring, mis kuvatakse roboti ekraanile.

char PointOut(int x, int y, unsigned long options=DRAW_OPT_NORMAL)

Kirjeldus: funktsioon, mis joonistab ekraanile kasutaja etteantud kohas punkti.

Parameetrid:

x - punkti koordinaat ekraani x-teljel.

y - punkti koordinaat ekraani y-teljel.

options - täiendavad seadistused teksti kuvamiseks. Selle võimaluse kasutamiseks on vajalik robotile paigaldada NXC jaoks optimeeritud ja täiustatud püsivara, seega selle parameetri võib lihtsalt sisestamata jätta.

Tagastusväärtus: funktsiooni töö tulemuseks on punkt, mis kuvatakse roboti ekraanile vastavalt kasutaja etteantud koordinaatidele.

Näide kasutamisest:

```
PointOut(50, 50);
```

Selgitus näitele: funktsiooni PointOut abil joonistatakse roboti ekraanile punkt 50 piksli kaugusel ekraani vasakust ja alumisest äärest.

char RectOut(int x, int y, int width, int height, unsigned long options=DRAW_OPT_NORMAL)

Kirjeldus: funktsioon roboti ekraanile ristküliku joonistamiseks.

Parameetrid:

x - ristküliku ülemise vasakpoolse nurga koordinaat ekraani x-teljel.

y - ristküliku ülemise vasakpoolse nurga koordinaat ekraani y-teljel.

width - ristküliku laius.

height - ristküliku kõrgus.

options - täiendavad seadistused teksti kuvamiseks. Selle võimaluse kasutamiseks on vajalik robotile paigaldada NXC jaoks optimeeritud ja täiustatud püsivara, seega selle parameetri võib lihtsalt sisestamata jätta.

Tagastusväärtus: funktsiooni töö tulemuseks on ristküliku kujutis roboti ekraanil.

Näide kasutamisest:

```
RectOut(0, 0, 60, 40);
```

Selgitus näitele: funktsiooni RectOut abil joonistatakse roboti ekraani alumisse vasakpoolsesse nurka ristkülik laiusega 60 ja kõrgusega 40 pikslit.

2.2.5 Protsessori moodul

void PowerDown()

Kirjeldus: lülitab roboti NXT põhiploki välja, lõpetades ühtlasi ka käesoleva programmi täitmise.

Parameetrid: funktsioonil puuduvad parameetrid.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
task main() {
      TextOut(0, LCD_LINE1, "Sulgemiseni 10 sekundit! ");
      Wait(SEC_10);
      PowerDown();
}
```

Selgitus näitele: programmis kirjeldatakse peaülesanne main, mille sees kuvatakse siis funktsiooni TextOut abil ekraani esimesele reale tekst "Sulgemiseni on 10 sekundit!". Seejärel paneb Wait funktsioon programmi 10 sekundiks ootele, misjärel funktsiooniga PowerDown töö lõpetatakse ja lülitatakse robot välja.

2.2.6 Sisendi moodul

void SetSensor(const byte & port, const unsigned int config)

Kirjeldus: funktsioon anduri tüübi ja asukoha (NXT põhiploki pesa, kuhu andur on ühendatud) süsteemis registreerimiseks. Funktsioon seab anduri töövalmis ja nii saab võimalikuks teha pöördumisi anduri näidu saamiseks.

Parameetrid:

port - NXT põhiploki pesa, kuhu andur, mida soovitakse registreerida, on ühendatud. Andureid saab ühendada ainult anduritele mõeldud pesadesse (4 pesa) ning vastavaid pesasid tähistavad konstandid on järgmised:

- S1 - põhiploki anduri pesa number 1.
- S2 - põhiploki anduri pesa number 2.
- S3 - põhiploki anduri pesa number 3.
- S4 - põhiploki anduri pesa number 4.

config - anduri konfiguratsiooniparameetrid. Lisaks anduritüübile võimalik ka, olenevalt andurist, valida täiendavaid seadistusi. Käesoleva töö raames piirdume andurite vaikeseadistustega ning määrame ära ainult anduri tüübi, mida saab valida järgnevate konstantide seast:

- SENSOR_SOUND - heliandur.
- SENSOR_TOUCH - puuteandur.
- SENSOR_LIGHT - valguse andur
- SENSOR_LOWSPEED - kasutamiseks ultraheli- ja kolmanda osapoole i2c anduri kirjeldamiseks.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
SetSensor(S1, SENSOR_TOUCH);
SetSensor(S2, SENSOR_SOUND);
SetSensor(S3, SENSOR_LIGHT);
SetSensor(S4, SENSOR_LOWSPEED);
```

Selgitus näitele: süsteemis registreeritakse 4 andurit: heli-, puute-, valguse ja ultraheli või muu i2c andur. Andurid registeeritakse vastavalt pesadele 1, 2, 3 ja 4.

NB! Andurite registreerimist on soovituslik teostada kohe main-ülesande alguses!

unsigned int Sensor(const byte & port)

Kirjeldus: funktsioon andurilt näidu küsimiseks.

Parameetrid:

port - NXT põhiploki pesa, kuhu andur, mille näitu soovitakse teada, on ühendatud. Pesade tähistused on järgmised:

- S1 - põhiploki sisendi pesa number 1.
- S2 - põhiploki sisendi pesa number 2.
- S3 - põhiploki sisendi pesa number 3.
- S4 - põhiploki sisendi pesa number 4.

Tagastusväärtus: funktsioon tagastab anduri hetkenäidu märgita täisarvu kujul.

Näide kasutamisest:

```
task main() {
SetSensor(S1, SENSOR_TOUCH);
while (Sensor(S1) != 1) {
            PlayTone(TONE_C5, SEC_1);
            Wait(SEC_1);
      }
      TextOut(0, LCD_LINE1, "Puuteandur aktiveeritud!");
}
```

Selgitus näitele: näites kirjeldatakse esmalt peaülesanne main. Ülesandes main registreeritakse funktsiooniga SetSensor puuteandur, mis on ühendatud sisendi pesa 1 külge. Seejärel käivitub while-tsükkel, mille käigus mängitakse kasutajale helinooti C 1-sekundilise kestvusega, intervalliga 1 sekund. While-tsükli tegevusi korratakse seni, kuni kasutaja vajutab puuteanduri nuppu, teisisõnu aktiveerib puuteanduri. Aktiveerimise korral tagastab funktsioon Sensor puuteandurilt väärtuse "1". Kuna kirjeldatud while-tsükli tingimuse kohaselt ei tohi puuteanduri näit ühega võrduda, katkestatakse tsükkel ning roboti ekraani esimesele reale kuvatakse kiri "Puuteandur aktiveeritud!".

2.2.7 Failihalduse moodul

Programmi täitmise jooksul muutujatesse salvestatud andmed kaovad sealt niipea kui programmi täitmine on lõpule viidud. Üks võimalus nende andmete (näiteks andurite näidud) salvestamiseks on kuvada need ekraanile ja kirjutada seejärel paberile üles. Samas on olemas ka elegantsem lahendus - salvestada andmed roboti mälus eraldi faili, mille saab arvutisse alla laadida ja siis juba vastavalt vajadusele edasi kasutada. Selline lähenemine võib aga algaja programmeerija jaoks osutuda liiga keeruliseks, seega detailidesse me süüvima ei hakka, kuid järgnevalt tooksin ära valiku tegevustest, mida kasutajal on võimalik failidega teostada.

Failioperatsioonidega tegelevad failihalduse mooduli funktsioonid. Kõnealuste funktsioonide abil on võimalik teha harjumuspäraseid failitoiminguid, nagu faili loomine, andmete salvestamine faili või sealt lugemine, failide ostimine ja kustutamine, faili nime muutmine jms.

Selguse mõttes olgu öeldud, et failihalduse mooduli funktsioonide kasutamine ei ole hädavajalik - NXC keeles saab väga edukalt kirjutada programme ka ilma andmeid failidesse salvestamata.

2.2.8 Lowspeed protokolli moodul

Selle mooduli funktsioone kasutatakse kolmanda osapoole anduritega suhtlemiseks. Kuna käesolev töö keskendub standardse LEGO Mindstorms NXT komplekti komponentidele, ei peatu me Lowspeed protokolli mooduli funktsioonidel pikemalt.

2.2.9 Mootorite moodul

void Off(byte outputs)

Kirjeldus: funktsioon blokeerib valitud mootorid piduriga ja lülitab need seejärel välja.

Parameetrid:

outputs - põhiploki väljundipesa(de), mille külge ühendatud mootoreid soovitakse välja lülitada, tähised. Võimalik valida järgmiste kombinatsioonide seast:

- OUT_A - juhtploki väljundipesa A mootor.
- OUT_B - juhtploki väljundipesa B mootor.
- OUT_C - juhtploki väljundipesa C mootor.
- OUT_AB - juhtploki väljundipesade A ja B mootorid.
- OUT_AC - juhtploki väljundipesade A ja C mootorid.
- OUT_BC - juhtploki väljundipesade B ja C mootorid.
- OUT_ABC - juhtploki kõigi väljundipesade mootorid.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
Off(OUT_A);
Off(OUT_AB);
Off(OUT_ABC);
```

Selgitus näitele: programmilõigu esimesel real lülitatakse välja pesa A külge ühendatud mootor. Teisel real lülitatakse sarnasel viisil korraga välja mootorid A ja B ning viimasel real kõik kolm mootorit.

void OnFwd(byte outputs, char pwr)

Kirjeldus: lülitab sisse valitud mootorid ning paneb nad pöörlema etteantud kiirusega edaspidises suunas.

outputs - põhiploki väljundipesa(de), mille külge ühendatud mootoreid soovitakse välja lülitada, tähised. Võimalik valida järgmiste kombinatsioonide seast:

- OUT_A - juhtploki väljundipesa A mootor.
- OUT_B - juhtploki väljundipesa B mootor.
- OUT_C - juhtploki väljundipesa C mootor.
- OUT_AB - juhtploki väljundipesade A ja B mootorid.
- OUT_AC - juhtploki väljundipesade A ja C mootorid.
- OUT_BC - juhtploki väljundipesade B ja C mootorid.
- OUT_ABC - juhtploki kõigi väljundipesade mootorid.

pwr - mootorite pöörlemise kiirus vahemikus 0 ...100. Võimalik määrata samas ulatuses väärtusi ka nullist allapoole - sellisel juhul toimub mootorite pöörlemine tagurpidi.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
OnFwd(OUT_AC, 50);
OnFwd(OUT_AC, -75);
```

Selgitus näitele: programmilõigu esimesel real pannakse funktsiooni OnFwd abil mootorid A ja C pöörlema edaspidi kiirusega 50. Teisel real pannakse samad mootorid pöörlema kiirusega -75, mis tähendab ühtlasi ka seda, et pöörlemine toimub tagurpidi.

NB! Liiga madalaks seatud pöörlemiskiirus võib osutuda ebapiisavaks roboti paigast liigutamisel! Siledatel põrandapindadel võib pöörlemiskiirus olla väiksem, vaiba või tekstiili peal tuleks kehvema haardumise tõttu pöörlemiskiirust tõsta.

void OnFwdSync(byte outputs, char pwr, char turnpct)

Kirjeldus: sarnane funktsiooniga on OnFwd, kuid erinevalt viimasest võimaldab sünkroniseerida kahe mootori pöörlemist ning seadistada roboti pööramist.

Parameetrid:

outputs - põhiploki väljundipesa(de), mille külge ühendatud mootoreid soovitakse välja lülitada, tähised. Võimalik valida järgmiste kombinatsioonide seast:

- OUT_A - juhtploki väljundipesa A mootor.
- OUT_B - juhtploki väljundipesa B mootor.

- OUT_C - juhtploki väljundipesa C mootor.
- OUT_AB - juhtploki väljundipesade A ja B mootorid.
- OUT_AC - juhtploki väljundipesade A ja C mootorid.
- OUT_BC - juhtploki väljundipesade B ja C mootorid.
- OUT_ABC - juhtploki kõigi väljundipesade mootorid.

pwr - mootorite pöörlemise kiirus vahemikus 0...100. Võimalik määrata samas ulatuses väärtusi ka nullist allapoole - sellisel juhul toimub mootorite pöörlemine tagurpidi.

turnpct - pööramistegur. Võimalik määrata väärtusi vahemikus -100 ... 100. Määrates väärtuseks 0 sõidab robot otse, väärtustel -100 ja 100 pöörab robot ühel kohal vastavalt kas vasakule või paremale. Väärtusel 50 üks mootoritest seisab. Muud numbrid antud arvude vahemikust panevad roboti pöörama suurema või väiksema kaarega.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
OnFwdSync(OUT_AC, 75, 0);
OnFwdSync(OUT_AC, 75, -100);
OnFwdSync(OUT_AC, 75, 50);
```

Selgitus näitele: programmilõigu esimesel real pannakse mootorid A ja C sünkroniseeritult pöörlema kiirusel 75. Kuna pööramisteguri väärtuseks on null, sõidab robot sirgjooneliselt ja pööramist ei toimu.

Teisel real pannakse mootorid A ja C sarnaselt eelnevale olukorrale pöörlema kiirusega 75, kuid seekord on pööramisteguri väärtuseks määratud -100, mis tähendab, et robot teeb ühel kohal pöörde kas paremale või vasakule (pööramissuund oleneb roboti ehitusest).

Kolmandal real sooritatakse samuti pööramine paremale või vasakule, kuid seekord on pöörlemisteguriks 50, mis tähendab, et robot pöörab ümber ühe ratta (üks ratastest seisab). Kuna pööramistegur on seekord positiivne, toimub pööramine teisel real toodud näitest vastupidises suunas.

void OnRev(byte outputs, char pwr)

Kirjeldus: analoogne funktsiooniga OnFwd, kuid roboti liikumine toimub tagurpidises suunas. Negatiivne pöörlemiskiirus muudab suuna vastupidiseks. Detailsem info funktsiooni OnFwd kirjeldusest.

void OnRevSync(byte outputs, char pwr, char turnpct)

Kirjeldus: analoogne funktsiooniga OnFwdSync, kuid roboti liikumine toimub tagurpidises suunas. Negatiivne pöörlemiskiirus muudab suuna vastupidiseks. Detailsem info funktsiooni OnFwdSync kirjeldusest.

void Coast(byte outputs)

Kirjeldus: funktsiooniga Off sarnane funktsioon mootorite seiskamiseks. Erinevalt funktsioonist Off jäetakse mootorid vabakäiguga pöörlema kuni roboti hoog raugeb. Detailsem info funktsiooni parameetritest ja kasutamisest funktsiooni Off kirjeldusest.

void RotateMotor(byte outputs, char pwr, long angle)

Kirjeldus: funktsioon mootori(te) pööramiseks teatud arvu kraadide võrra.

Parameetrid:

outputs - põhiploki väljundipesa(de), mille külge ühendatud mootoreid soovitakse välja lülitada, tähised. Võimalik valida järgmiste kombinatsioonide seast:

- OUT_A - juhtploki väljundipesa A mootor.
- OUT_B - juhtploki väljundipesa B mootor.
- OUT_C - juhtploki väljundipesa C mootor.
- OUT_AB - juhtploki väljundipesade A ja B mootorid.
- OUT_AC - juhtploki väljundipesade A ja C mootorid.
- OUT_BC - juhtploki väljundipesade B ja C mootorid.
- OUT_ABC - juhtploki kõigi väljundipesade mootorid.

pwr - mootorite pöörlemise kiirus vahemikus 0...100. Võimalik määrata samas ulatuses väärtusi ka nullist allapoole - sellisel juhul toimub mootorite pöörlemine tagurpidi.

angle - pöörlemisnurk kraadides. Võimaldab pöörata roboti rattaid täpselt ettemääratud pöörete arvu võrra. Üks roboti ratta pööre võrdub 360 kraadiga.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk “tühi”, ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
RotateMotor(OUT_AB, 75, 720);
RotateMotor(OUT_AB, -75, 720);
```

Selgitus näitele: esimesel real pööratakse mootoreid A ja B funktsiooni RotateMotor abil tugevusega 75 edaspidi 720 kraadi ehk kahe täispöörde jagu. Teisel real toimub samalaadne mootorite pööramine vastupidises suunas, kuna mootorite pöörlemiskiiruseks on määratud negatiivne arv.

void RotateMotorEx(byte outputs, char pwr, long angle, char turnpct, bool sync, bool stop)

Kirjeldus: funktsioon mootori(te) pööramiseks teatud arvu kraadide võrra.

Parameetrid:

outputs - põhiploki väljundipesa(de), mille külge ühendatud mootoreid soovitakse välja lülitada, tähised. Võimalik valida järgmiste kombinatsioonide seast:

- OUT_A - juhtploki väljundipesa A mootor.
- OUT_B - juhtploki väljundipesa B mootor.
- OUT_C - juhtploki väljundipesa C mootor.
- OUT_AB - juhtploki väljundipesade A ja B mootorid.
- OUT_AC - juhtploki väljundipesade A ja C mootorid.
- OUT_BC - juhtploki väljundipesade B ja C mootorid.
- OUT_ABC - juhtploki kõigi väljundipesade mootorid.

pwr - mootorite pöörlemise kiirus vahemikus 0...100. Võimalik määrata samas ulatuses väärtusi ka nullist allapoole - sellisel juhul toimub mootorite pöörlemine tagurpidi.

angle - pöörlemisnurk kraadides. Võimaldab pöörata roboti rattaid täpselt ettemääratud pöörete arvu võrra. Üks roboti ratta pööre võrdub 360 kraadiga.

turnpct - pööramistegur. Võimalik määrata väärtusi vahemikus -100 ... 100. Määrates väärtuseks 0 sõidab robot otse, väärtustel -100 ja 100 pöörab robot ühel kohal vastavalt kas vasakule või paremale. Väärtusel 50 üks mootoritest seisab. Muud numbrid antud arvude vahemikust panevad roboti pöörama suurema või väiksema kaarega.

sync - mootorite sünkroonis pööramine. Väärtus TRUE aktiveerib selle võimaluse, FALSE ignoreerib. Kui pööramistegur on nullist erinev, peab olema kindlasti aktiveeritud, vastasel juhul pööramist ei toimu!

stop - määrab ära, kas pöörlemise lõppedes mootorid seisatakse piduriga või mitte. Väärtus TRUE aktiveerib, FALSE ignoreerib.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
RotateMotorEx(OUT_AB, 75, 720, 0, FALSE, TRUE);
RotateMotorEx(OUT_AB, -75, 720, 25, TRUE, FALSE);
```

Selgitused näitele: esimesel real pööratakse mootoreid A ja B funktsiooni RotateMotorEx abil tugevusega 75 edaspidi 720 kraadi ehk kahe täispöörde jagu, sealjuures robot sõidab sirgjooneliselt. Mootoreid ei hoita sünkroonis ja mootorite pööramise lõppedes seisatakse need piduri abil.

Teisel real toimub samalaadne mootorite pööramine vastupidises suunas, kuna mootorite pöörlemiskiiruseks on määratud negatiivne arv. Erinevalt esimesest reast toimub siin ka roboti pööramine, kuna pööramisteguriks on määratud 25. Mootoreid hoitakse sünkroonis, kuid seekord ei pidurdata pärast mootorite pööramise lõppemist.

2.2.10 Heli moodul

char PlayTone(unsigned int frequency, unsigned int duration)

Kirjeldus: kindla sageduse ja kestvusega helinoodi esitamine.

Parameetrid:

frequency - helinoodi võnkesagedus. Helisagedust on võimalik ette anda numbriliselt (nt. 400) või kasutada süsteemi helikonstante (süsteemis registreeritud kindla sagedusega noodid). Saadaval on üsna suures valikus erinevaid helikonstante, järgnevalt tuuakse ära mõned näited:

- TONE_C3 - kolmanda oktaavi C noot (sagedus 131)
- TONE_D3 - kolmanda oktaavi D noot (sagedus 147)
- TONE_E3 - kolmanda oktaavi E noot (sagedus 165)
- TONE_F3 - kolmanda oktaavi F noot (sagedus 175)
- TONE_G3 - kolmanda oktaavi G noot (sagedus 196)
- TONE_A3 - kolmanda oktaavi A noot (sagedus 220)
- TONE_B3 - kolmanda oktaavi B noot (sagedus 247)
- ja palju muud.

duration - noodi esitamise kestvus (sekundites või millisekundites) ühes järgnevatest vormingutest:

- SEC_”number” - aeg sekundites
- MS_”number” - aeg millisekundites

Tagastusväärtus: funktsiooni väljundiks on kasutajale esitatav heli.

Näide kasutamisest:

```
PlayTone(TONE_A3, MS_500);
PlayTone(220, MS_500);
```

Selgitus näitele: mõlemad read kujutavad sama tegevust - esitatakse 3 oktaavi noot A kestvusega 500 millisekundit. Esimesel juhul on noot märgitud kasutades konstanti, teisel juhul on noodi sagedus kirjutatud numbrina. Numbri asemel saab edukalt kasutada ka täisarvulise muutuja väärtust (näiteks: PlayTone(x, MS_500);). Sellisel juhul peab muutuja x olema eelnevalt defineeritud ja talle peab olema omistatud mõistlik sageduse väärtus.

NB! Funktsiooni PlayTone väljakutsumisel ei oota süsteem heli esitamise lõppemist, vaid läheb muude programmi tegevuste täitmisega edasi. Kui programmis asuvad järjestikku mitu funktsiooni PlayTone väljakutset, pannakse neis kirjeldatud noodid väikese viivitusega samaaegselt kõlama. Sellise olukorra vältimiseks on soovituslik funktsiooni PlayTone väljakutse järel kasutada ootamise funktsiooni Wait, mille parameetriks määrata noodi esitamise kestvusest pisut pikem aeg.

void PlaySound(const int & aCode)

Kirjeldus: funktsioon süsteemihelide esitamiseks.

Parameetrid:

aCode - süsteemiheli tähis, valikus järgmised helid:

- SOUND_CLICK - klõpsatuse heli
- SOUND_DOUBLE_BEEP - topelt piiks
- SOUND_DOWN - “langev” heli (muutub madalamaks)
- SOUND_UP - “tõusev” heli (muutub kõrgemaks)
- SOUND_LOW_BEEP - veateate heli
- SOUND_FAST_UP - kiiresti “tõusev” heli (muutub kiiresti kõrgemaks)

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk “tühi”, ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
PlaySound(SOUND_UP);
PlaySound(SOUND_DOWN);
```

Selgitus näitele: funktsiooni PlaySound abil esitatakse kõige pealt "tõused" ning seejärel "langev" heli.

byte SoundVolume()

Kirjeldus: funktsioon tagastab roboti helitugevuse (kui valjult hetkel helisid esitatakse).

Parameetrid: funktsioonil puuduvad parameetrid.

Tagastusväärtus: funktsioon tagastab süsteemi helitugevuse skaalal 0 ... 100.

Näide kasutamisest:

```
byte x;
x = SoundVolume();
```

Selgitus näitele: programmilõigus defineeritakse esmalt muutuja x ning seejärel omistatakse talle väärtus, mille funktsioon SoundVolume tagastab.

void SetSoundVolume(byte volume)

Kirjeldus: funktsioon süsteemi helitugevuse määramiseks.

Parameetrid:

volume - helitugevuse uus väärtus vahemikus 0 ... 100.

Tagastusväärtus: kuna funktsiooni tüübiks on void ehk "tühi", ei tagasta see funktsioon pärast töö lõpetamist midagi.

Näide kasutamisest:

```
SetSoundVolume(10);
```

Selgitus näitele: funktsiooni SetSoundVolume abil määratakse süsteemi helitugevuseks 10.

2.2.11 Kasutajaliidese moodul

unsigned int BatteryLevel()

Kirjeldus: tagastab NXT põhiploki aku laetuse taseme millivoltides.

Parameetrid: funktsioonil puuduvad parameetrid.

Tagastusväärtus: aku laetuse tase millivoltides.

Näide kasutamisest:

```
unsigned int akuTase;
akuTase = BatteryLevel();
```

Selgitus näitele: näites defineeritakse esmalt märgita täisarvu tüüpi muutuja akuTase ning seejärel omistatakse sellele funktsiooni BatteryLevel tagastatud aku laetuse väärtus.

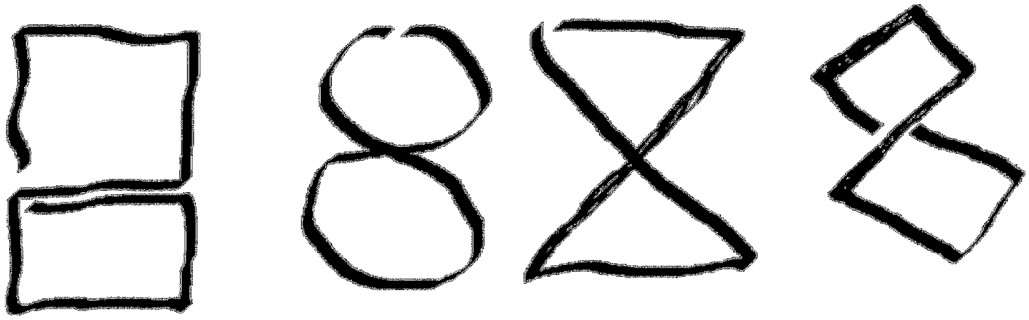
Eespool kirjeldatud funktsioonidest peaks algajale programmeerijale NXC keelega tutvumiseks ja lihtsamate ülesannete lahendamiseks piisama. Nüüd, kui teooria osa on seljataga, oleks õige aeg õpitut ka praktikas rakendada ning järgnevas peatükis saabki lugeja võimaluse panna end proovile roboti programmeerimises NXC keeles.

3. Ülesandeid harjutamiseks

Järgnev töö osa pakub lugejale ülesandeid iseseisvaks harjutamiseks. Ülesannete juures on ära toodud ülesande raskusaste, ülesande püstitus, pseudokood (programmi eeldatava ülesehituse kirjeldus) ning nimekiri soovituslikest funktsioonidest, mida ülesande lahendamisel kasutada võiks. Ülesande valmislahendusi lisatud ei ole, kuna reeglina on ühe probleemi lahendamiseks mitu erinevat moodust ning üksainus korrektne lahendus puudub. Siiski on lahendajat abistamas pseudokood, mis annab aimu programmi sisust ja tegevuste järgnevusest. Pseudokood on märgitud sinise värviga.
Lisaks pseudokoodile on soovitatav tutvuda ka alapeatükiga 3.4, kus on ära toodud mõningad näpunäited ülesannete lahendamiseks.

3.1 Ülesanne 1: Kaheksa sõitmine

Raskusaste: kerge
Kirjeldus: tegemist on klassikalise vigursõidu ülesandega, kus sõidetakse kahe takistuse ümber 8-kujulisi tiire. Harjutus sobib väga hästi roboti mootorite tundmaõppimiseks.
Enne harjutuse läbiviimist tuleb ette valmistada piisavalt suur pind roboti manööverdamiseks ning asetada sellele kaks kitsat takistust (nt. WC-paberi rullide südamikud). Takistused peavad paiknema nii, et robotil oleks võimalik nende vahelt läbi sõita ka teravama nurga alt. Mida lähemal takistused üksteisele paiknevad, seda raskem on nende vahelt slaalomit sõita. Sõidu ajal ei tohi robot takistusi riivata.
Märkused: Alustuseks on lihtsam läbida seda rada “digitaalse” kaheksa kujul. Seejärel võib proovida sõita traditsioonilist, ümmargust kaheksat ning modifitseerida sõidutrajektori veelgi. Ideid katsetamiseks:



Roboti koostamine: ülesande lahendamiseks on vaja robotiga ühendada kaks mootorit, mis veaksid roboti rattaid ja võimaldaks tal ruumis ringi liikuda. Mootorid võib ühendada näiteks roboti väljundipesade B ja C külge.

Programmi ülesehitus: kuna selle ülesande raames on lahendajal võimalik katsetada erinevate trajektooridega, on universaalset lahenduse pseudokoodi kirja panna raske. Ülesannet lahendama asudes on soovituslik eeldatav liikumistrajektoor üles joonistada ning juurde märkida ka roboti liikumise sammud (sõit otse, pööramine jms). Seejärel tuleks need sammud koos konkreetsete suurustega (teepikkus, roboti pööramisnurk) programmis funktsioonide abil kirja panna.

Soovitatavad funktsioonid: OnFwdSync, RotateMotor, RotateMotorEx

3.2 Ülesanne 2: Robot-moosekant

Raskusaste: keskmine. Ülesanne sobib hästi muusikatunnis lahendamiseks.

Kirjeldus: koostada esineja robot, mis sõidab publiku ette, mängib ette lühikese meloodia, ootab ära publiku aplausi ja sõidab seejärel oma kohale tagasi. Robot peaks tegutsema järgmiselt:

1. Sõidab ~50 cm ettepoole ja jääb seisma.
2. Esitab lühikese viisijupikese.
3. Jääb ootama aplausi.
4. Aplausi kõlades sõidab tagasi oma kohale ja pöörab end "näoga" samasse suunda, kus robot asus enne programmi täitmise alustamist.

Märkused

- Ideaalis võiks roboti töö alg- ja lõppasend olla samad, s.t. et robot lõpetab oma töö samas kohas ja "näoga" samas suunas, kus ta programmi täitmise alguses asus.
- Viisijupike tuleks kirja panna järjestikku asuvate helinootide esitamise funktsioonidega (näiteks PlayTone(noot1), PlayTone(noot2) jne). Helinootide esitamisel tuleks pöörata tähelepanu ka esituse kestvusele - liiga pikad noodid võivad meloodia ära rikkuda, seetõttu tasub enne katsetada paari esimese noodiga, leidmaks optimaalne esitamise kestvus.
- Viisijupikese aluseks on soovituslik valida mõni lihtne lastelaul, näiteks "Kus on minu koduke". Meloodia kirjeldamisel programmis tulevad kasuks ka põhiteadmised noodikirjast (oskus noodikirja lugeda).
- Kuna roboti andur ei suuda teha vahet plaksutamisel ja mõnel muul helil, siis tuleks lihtsalt seadistada robot reageerima valjule helile (anduri näit > 60%-80%). Helitugevus, millele

robot reageerima seadistatakse, peab olema piisavalt kõrge, et robot ei asuks tegutsema mingeid kõrvalisi taustahelisid kuuldes.

Roboti koostamine: ülesande lahendamiseks on vaja robotiga ühendada kaks mootorit, mis veaksid roboti rattaid ja võimaldaks tal ruumis ringi liikuda. Mootorid võib ühendada näiteks roboti väljundipesade B ja C külge. Lisaks läheb vaja ka heliandurit, mille võiks ühendada sisendi pesa number 2 külge.

Programmi ülesehitus

```
// Töö algus
pööra_mootoreid x kraadi;     // x - 50 cm-le vastav pöörete arv
esita noot1;
esita noot2;
…
esita nootN;
kuni helianduri_näit < 60 {
      ära tee midagi;
}
pööra robotit 180 kraadi;
pööra_mootoreid x kraadi;
pööra robotit 180 kraadi;
// Töö lõpp
```

Soovitatavad funktsioonid: OnFwdSync, RotateMotor, RotateMotorEx, PlayTone, Sensor

3.3 Ülesanne 3: Pimesikk

Raskusaste: kerge/kesmine

Kirjeldus: programmeerida robot ruumis “kinnisilmi” liikuma. Liikumisel juhindub robot oma puuteanduri näidust: kui puuteandur tuvastab kontakti takistusega, peab robot oma liikumissuunda muutma. Tegevuste järgnevus on selline:

1. Robot alustab programmi käivitudes otsesuunas liikumist.
2. Kui puuteandur fikseerib kontakti, lõpetab robot otsesuunas liikumise ja tagurdab veidi.
3. Tagurdamise lõppedes pöörab robot kohapeal vabalt valitud suunas 60 kraadi.

4. Pööramise lõppedes jätkub töö tegevuste loetelu esimesest punktist.

Roboti koostamine: ülesande lahendamiseks on vaja robotiga ühendada kaks mootorit, mis veaksid roboti rattaid ja võimaldaks tal ruumis ringi liikuda. Mootorid võib ühendada näiteks roboti väljundipesade B ja C külge. Lisaks läheb vaja ka puuteandurit, mis peaks asuma roboti eesotsas ja ulatuma muudest roboti osadest ettepoole.

Programmi ülesehitus

```
// Töö algus
kuni lõpmatuseni {
      pööra_mootoreid;
      kui (puuteandur!=0) {
            tagurda;
            pööra robotit 60 kraadi;
      }
// Töö lõpp
```

Märkus: ülesande raskusastme tõstmiseks võib lisada järgmise tingimuse: roboti töö peab lõppema siis, kui heliandur fikseerib valju heli. Sellisel juhul tuleb loomulikult ühendada roboti külge ka heliandur ning muuta vastavalt ka programmi ülesehitust.

Soovitatavad funktsioonid: OnFwdSync, RotateMotor, RotateMotorEx, Sensor

3.4 Abiks lahendamisel

Ülesannete lahendamisel võib ette tulla olukordi, kus väheste kogemustega lahendaja ei pruugi roboti mõne konkreetse tegevuse kirjeldamisega hakkama saada. Selliste olukordade ennetamiseks toon ära mõningad näpunäited, mis võiksid lahendajale programmide koostamisel kasulikud olla.

3.4.1 Kuidas panna robot läbima kindlaksmääratud vahemaad? Kuidas pöörata robotit teatud arvu kraadide võrra? [10]

Kindla vahemaa läbimine

Sageli võib tekkida vajadus sõidutada robotit teatud vahemaa jagu ette- või tahapoole. Kuna NXC-s puudub funktsioon, mis võtaks sisendiks soovitud vahemaa ja teostaks siis vastava mootorite pööramise, peame kasutama funktsiooni RotateMotorEx (rohkem infot funktsioonide

peatükis). Kõnealune funktsioon küsib muude sisendite seas kasutajalt ka kraadide arvu, mille võrra mootoreid pöörata tuleks, ning just kraadideks tulebki soovitud vahemaa teisendada. Teisenduse läbiviimiseks on meil tarvis välja selgitada roboti rataste mõõtmed. LEGO Mindstorms robotite ehitamisel kasutatakse üldjuhul kolmes eri suuruses rattaid, rataste mõõtmeid tutvustab järgnev tabel:

Ratas	**Läbimõõt (mm)**	**Ümbermõõt (mm)**
Väike	43,2	135,7
Keskmine	56	175,9
Suur	82	257,6

Tabel3: LEGO Mindstorms robotite rataste mõõtmed

Vahemaa teisendamine kraadideks

Vahemaa teisendamiseks vastavaks mootorite pöörete arvuks kraadides kasutame järgmist valemit:

KRAADIDE ARV = VAHEMAA/ÜMBERMÕÕT*360

Vahemaa - teepikkus, mida robot peaks läbima. Esitatakse millimeetrites.

Ümbermõõt - roboti ratta ümbermõõt millimeetrites.

Saadud tulemus tuleks seejärel lisada funktsiooni RotateMotorEx väljakutsesse, näide:

```
RotateMotorEx(OUT_AB, 75, KRAADIDE ARV, 0, TRUE, TRUE);
```

Näites pööratakse mootoreid A ja B kiirusega 75 kasutaja poolt leitud kraadide arvu võrra, roboti pööramist ei toimu (parameetri turn väärtus 0), mootorite pöörlemist hoitakse sünkroonis ning pöörlemise lõppedes pidurdatakse.

Roboti pööramine teatud arvu kraadide võrra

Roboti pööramiseks soovitud nurga võrra tuleb toimida sarnaselt eelmises punktis antud juhistele - esmalt teisendame soovitud roboti pööramisnurga mootori pööreteks kraadides ning seejärel kasutame funktsiooni RotateMotorEx pööramise teostamiseks.

Pööramine kohapeal mõlema ratta abil

Roboti pööramisnurga teisendamiseks mootori pööreteks kasutame järgmist valemit:

KRAADIDE ARV = (ROBOTI PÖÖRAMISE NURK*RATASTE VAHE)/LÄBIMÕÕT

Roboti pööramise nurk - nurk, mille võrra robotit soovitakse pöörata oma telje ümber.

Rataste vahe - roboti rataste omavaheline kaugus, mõõdetakse rataste keskkohast. Mõõdud millimeetrites.

Läbimõõt - roboti ratta läbimõõt millimeetrites.

Saadud tulemus tuleks seejärel lisada funktsiooni RotateMotorEx väljakutsesse, näide:

```
RotateMotorEx(OUT_AB, 75, KRAADIDE ARV, 100, TRUE, TRUE);
```

Näites pööratakse mootoreid A ja B kiirusega 75 kasutaja poolt leitud kraadide arvu võrra, kusjuures mootorid pöörlevad eri suundades (parameetri turn väärtus 100), mootorite pöörlemist hoitakse sünkroonis ning pöörlemise lõppedes pidurdatakse.

NB! Pärast pööramise lõpetamist on soovituslik lisada lühike ootepaus Wait funktsiooni kujul. See aitab seisata roboti mootorid (vastasel juhul võib pöörlemine jätkuda ka pärast kindlaksmääratud pöörete arvu täitumist).

Roboti pööramine ümber ühe ratta

Paljuski sarnanev eelmise pöörlemise juhuga (kohapeal pöörlemine kahe ratta osalusel). Toimime samamoodi: teisendame roboti pööramisnurga mootorite pööreteks kraadides järgmise valemi abil:

KRAADIDE ARV = (ROBOTI PÖÖRAMISE NURK*RATASTE VAHE*2)/LÄBIMÕÕT

seejärel lisame saadud tulemuse funktsiooni RotateMotor väljakutsesse:

```
RotateMotor(OUT_A, 75, KRAADIDE ARV);
```

Kasutame siinkohal funktsiooni RotateMotor, kuna soovime pöörata vaid ühte mootorit. Antud näites pööratakse mootorit A kiirusega 75 kasutaja poolt leitud kraadide arvu võrra. Roboti pööramine toimub seega ümber mootori B külge ühendatud ratta.

3.4.2 Kuidas panna robotit reageerima anduri näidu muutustele? [11]

Programmi koostades võib sageli tekkida vajadus siduda teatud roboti toimingud anduri näidu muutumisega: sooritada mingeid tegevusi või, vastupidi, panna robot ootele kuni andur mõõdab välja mingi kindla väärtuse. Selleks sobivad hästi while- ja until-tsüklid.

While-tsükkel

While-tsüklit saab kasutada tegevuste kordamiseks. Tegevusi korratakse seni, kuni täidetakse tsükli katkestamise tingimus. While-tsükli kasutamine on väga levinud ning selline koodikonstruktsioon on esindatud enamikes prorgammeerimiskeeltes. Näide while-tsüklist:

```
x = 0;
while (x < 10) {
	x = x + 1;
}
```

Näites on tsükli katkestamise tingimuseks muutuja x väärtus 10. See tähendab, et kui muutuja väärtuseks saab mingil hetkel 10, katkestatakse tsükli täitmine ja liigutakse järgmise programmi osa juurde. Tsükli katkestamise tingimus kirjutatakse märksõna while järele ümarsulgudesse. Tsükli katkestamise tingimuse kirjeldamisel peab kindlasti jälgima, et tingimus oleks täidetav. Antud juhul tähendab see, et kui tsükkel alustab x väärtusega 0, peab tsükli sees toimuma x väärtuse suurendamine. Vastasel juhul ei jõua muutuja x väärtus kunagi 10-ni ning programm jääb tsüklisse "lõksu". Sama põhimõte laieneb ka anduritele: kui tsükli katkestamise tingimuseks on seatud selline anduri näit, mida ei ole võimalik mõõta, jääb programm tsüklit lõpmatult kordama. Näide anduri näidu kasutamisest while-tsüklis:

```
while (SensorUS(S4) >= 10) {
	OnFwd(OUT_AC, 50);
}
```

Ülaltoodud näites on tsükli katkestamise tingimuseks ultraheli sensori näidu väärtus 10 (ehk siis robot on 10 cm kaugusel takistusest). Anduri näitu küsitakse antud juhul funktsiooniga SensorUS, mis on spetsiaalne funktsioon ultraheli anduri jaoks. Samas saab seda teha ka üldise funktsiooni Sensor abil, mis töötab ka teiste andurite puhul. Tsükli kehas (looksulgude vahel asuvad tegevused) toimub roboti sõidutamine otsesuunas (funktsiooniga OnFwd pööratakse mootoreid A ja C kiirusega 50). Kui robot jõuab takistusele 10 või vähema sentimeetri kaugusele, katkestatakse tsükli täitmine ja lõpetatakse mootorite pööramine.

Until-tsükkel

Juhul, kui soovitakse panna robot ootama mingit kindlat anduri näitu, saab kasutada until-tsüklit. Until-tsüklini jõudmisel pannakse programm ootele ja mingeid muid tegevusi tsükli läbiviimisel

ei sooritata. Nagu ka while-tsükli puhul on ka siin oht tsükli lõpetamise tingimusega eksida, lastes programmil lõputult oodata. Näide until-tsükli kasutamisest:

```
until (Sensor(S1));
OnFwd(OUT_AC, 50);
...
```

Antud näites oodatakse until-tsüklis signaali puuteandurilt. Kuna puuteanduri näit on vaikimisi FALSE (ehk siis anduri nupp pole alla vajutatud), pole ka tsükli katkestamise tingimus (puuteanduri väärtus TRUE) täidetud ning programm jääb ootele. Puuteanduri nupu allavajutamisel muutub anduri näit tõeseks ehk tagastatakse TRUE väärtus ning until-tsükkel katkestatakse. Seejärel asutakse täitma järgmisi programmi tegevusi, milleks antud juhul on mootorite pööramine funktsiooni OnFwd abil.

3.4.3 Programm ei käivitu?

Põhjuseid võib olla mitmeid, kuid alati tasub üle kontrollida, kas programmis on ikka kirjeldatud peaülesanne main. Kõik muud ülesanded tuleb välja kutsuda main ülesande sees.

Kokkuvõte

Käesoleva bakalaureusetöö töö raames võeti eesmärgiks luua eestikeelsed õppematerjalid NXC keeles programmeerimisega alustamiseks. Vastloodud õppematerjalid peaksid oluliselt täiendama seniseid eestikeelseid NXC-teemalisi materjale ning loodetavasti aitab see omakorda tuua rohkem üldhariduskoolide õpilasi LEGO Mindstorms NXT robotite programmeerimise ja ka programmeerimise juurde üldiselt. Käesolev töö on üles ehitatud kolmele üksteisega tihedalt seotud osale.

Töö esimeses osas tutvustatakse LEGO Mindstorms NXT robotite komplekti. Jutuks tuleb ka käesoleva aasta sügisel müüki saabuv uus EV3 robotite põlvkond. Robotitelt liigutakse sujuvalt edasi nende programmeerimise vahendite juurde, kus tutvustatakse erinevaid programmeerimiskeskkondi, teiste seas ka NXC keeles programmide kirjutamiseks kasutatavat BricxCC keskkonda. Töö esimene osa on ka sissejuhatuseks järgmisele, funktsioonide dokumenteerimisele keskenduvale töö osale: lugejale antakse põhjalik ülevaade NXC keelest.

Teises osas ootab lugejat põhjalik NXC enimkasutatavate funktsioonide dokumentatsioon: iga funktsiooni juures on äratoodud lühikirjeldus, sisend- ja väljundparameetrid ning kasutusnäide. Funktsioonide dokumentatsioon pärineb NXC keele ametlikust dokumentatsioonist.

Töö kolmas osa keskendub teooria rakendamisele praktikas - lugeja saab võimaluse lahendada kolm ülesannet roboti programmeerimise peale. Lisaks ülesannetele pakutakse algajale programmeerijale ka kasulikke näpunäiteid ning soovitusi selle jaoks eraldatud alapeatükis.

Käesolevas töös pakutavad juhendid ja õpetused peaksid olema õpilaste jaoks piisavad õpetaja juhendamisel NXC keeles programmeerimisega alustamiseks.

LEGO Mindstorms NXT robot programming using NXC language

Bachelor Thesis (6EAP)

Nikolai Konovalov

Summary

The aim of this bachelor thesis was to create educational materials for introducing students of primary and high school to LEGO Mindstorms NXT robot programming using NXC language. The texts are composed in a way that should be easily understandable by students and teachers in order to avoid confusion and ease the studying process.

This thesis provides a thorough overview of LEGO Mindstorms NXT robot programming using variety of tools, including BricxCC environment for writing programs in NXC. The technical features of LEGO Mindstorms NXT robot sets are also discussed.

In addition to general overview of NXC language, this thesis paper also provides an in-depth documentation of most commonly used NXC functions along with usage recommendations for each function. This part is largely based on official NXC documentation created by John Hansen.

In addition to theoretical part, this document also contains three practical exercises, which can be solved using function documentation and instructions from previous chapters. Exercises provide the reader with opportunity to write his/her first NXC programs following well-documented descriptions and guidelines. There is also a section with useful tips and tricks provided to assist the reader in solving exercises.

In conclusion it should be safe to say that materials provided in this document are sufficient for student to start studying programming in NXC language under teacher's supervision.

Viited

1. http://www.irobot.ee/PublishedService?pageID=18&freePage=271 (05.12.2012)
2. http://www.robootika.ee/lego/projekt/index.php/projektist/ (09.12.2012)
3. http://www.tiigrihype.ee/et/tehnoloogia-kaasfinantseerimine (12.12.2012)
4. http://mindstorms.lego.com/en-us/News/ReadMore/Default.aspx?id=476243 (01.05.2013)
5. http://www.legoeducation.us/eng/misc/comparingEV3andNXT.cfm (01.05.2013)
6. http://mindstorms.lego.com/en-us/News/ReadMore/Default.aspx?id=476781 (02.05.2013)
7. http://educationnews.legoeducation.us/News/133/LEGO-Education-Evolves-STEM-Learning-with-the-Next-Generation-LEGO-MINDSTORMS-Education-EV3-Platform (02.05.2013)
8. http://bricxcc.sourceforge.net/ (04.03.2013)
9. http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/group___n_x_t_firmware_modules.html (08.05.2013)
10. http://www.robootika.com/post/Lego-rattad-ja-matemaatika.aspx (08.05.2013)
11. http://www.robootika.com/post/Juhend-while-until-tsuklite-kasutamine.aspx (08.05.2013)

Illustratsioonid

12. http://upload.wikimedia.org/wikipedia/commons/thumb/c/cc/Nxt-brique.jpg/462px-Nxt-brique.jpg
13. http://cache.lego.com/upload/contentTemplating/Mindstorms2History/images/pic8FF18E3A1305C6E760132AEE82906D22.jpg
14.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Nikolai Konovalov
(*autori nimi*)
(sünnikuupäev: 15. detsember 1990)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
NXC
,
(*lõputöö pealkiri*)

mille juhendaja on Anne Villems,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus **13.05.2013**