TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Priit Rand

Platvormiülene *NXC* keskkond

Magistritöö (30 EAP)

Juhendaja: MSc Anne Villems
Kaasjuhendaja: MSc Taavi Duvin

Autor:..... "...." mai 2012
Juhendaja:..... "...." mai 2012
Kaasjuhendaja:..... "...." mai 2012

Lubada kaitsmisele
Professor..... "...." mai 2012

TARTU 2012

Sisukord

Sissejuhatus

Paljudes Eesti koolides on tänaseks traditsiooniliste ainete kõrvale tekkinud robootika suunitlusega õppeained ja huviringid. Nende eesmärgiks on arendada õpilaste loovust, loogilist mõtlemist, tehnilist taipu ja programmeerimisoskust. Peamiselt kasutatakse õpetamisel LEGO MINDSTORMS NXT robootikakomplekte. Tegemist on õppevahendiga [1], mis koosneb NXT programmeeritavast põhiplokist, mootoritest, anduritest, vajalikest juhtmetest ning LEGO klotsidest.

NXT programmerimiseks on olemas mitmeid vahendeid. Üheks võimaluseks on lihtsalt kasutatav graafiline *NXT-G* keel. Lihtsus on ka selle süsteemi puuduseks, kuna *NXT-G* võib edasijõudnud programmeerija jaoks muutuda kiiresti igavaks ja keerukamate programmide puhul ebamugavaks kasutada. Näiteks keerukamat programmiosa ei ole võimalik korraga ühele ekraanitäiele kuvada, erinevate programmiosade ühest kohast teise kopeerimine on tülikas ja robotis käivitatavasse baitkoodi teisendatud programm on võrreldes alternatiivsete keeltega suure mälumahuga [2]. Võimalus on kasutada ka spetsiaalselt LEGO MINDSTORMS programmeerimiseks mõeldud tekstipõhist *NXC* (*Not eXactly C*) keelt, millega saab luua keerukamaid, kuid samas vähem mälu nõudvaid programme.

NXC keeles programmide kirjutamiseks on Windows operatsioonisüsteemile parimaks arenduskeskkonnaks Bricx Command Center (BricxCC) [2]. Mac OS süsteemile on mõningate puudustega rakendus NeXT Command Center [3]. Näiteks ei tööta selles mõned menüü valikud ning programmi koodi värvimine võiks mitmekesisem olla. Linux platvormile puudub spetsiaalselt *NXC* kasutamiseks mõeldud vahend.

NXC-ga programmeerimisel võivad saada takistuseks erinevate operatsioonisüsteemidega arvutite olemasolu kodus, koolis, õpetajal, õpilasel. Samuti võivad erineda ka erinevate koolide arvutiklassides kasutatavad operatsioonisüsteemid. *NXC* programmeerimisega alustamiseks peab muretsema, missugune programm konkreetsele arvutile sobib. Erinevatest rakendustest tingituna on universaalset *NXC* kasutamise juhendmaterjali keerukas koostada.

NXC võib olla küll hea keel LEGO MINDSTORMS robotite programmeerimiseks aga selles kasutatavad sõnad ja funktsioonid on algaja eestikeelse programmeerija jaoks raskesti mõistetavad. Programmeerimise lihtsustamiseks võiks juhendmaterjalide loojatel olla võimalus *NXC* keelt tõlkida ja luua standardseid algajat programmeerijat abistavaid funktsioone.

Antud magistritöö eesmärgiks on luua *NXC* kasutamiseks keskkond, mis oleks arvutisse paigaldatuna ühtmoodi kasutatav kõigil kolmel enamlevinud operatsioonisüsteemil. Uue vahendi loomiseks uuritakse *NXC* keelt ja erinevatele platvormidele leiduvaid rakendusi. Vaadeldud programmide positiivseid omadusi arvestatakse ka uue arendusvahendi loomisel. Algaja programmeerija abistamiseks luuakse *NXC* keelele eestikeelsete nimedega standardfunktsioonide avatud teek. Igal süsteemi kasutajal saab olema võimalus antud töö autori poolt loodud funktsioone muuta ning uusi lisada.

Käesolev magistritöö koosneb kolmest osast. Töö esimeses osas antakse ülevaade *NXC* keelest ning selgitatakse, kuidas erinevatel platvormidel seada üles töökeskkond *NXC* keeles programmeerimiseks suvalise tekstiredaktoriga. Töö teises osas kirjeldatakse erinevatel platvormidel kasutatavaid vahendeid NXT robotite programmeerimiseks *NXC* keeles. Viimases peatükis kirjeldatakse loodud rakendust NXCEesti ja antakse ülevaade selle kasutamisest. Antud tööga on kaaasas CD (Lisa 1), mis sisaldab loodud tarkvara, kasutusjuhendit ning programmi lähtekoodi ja dokumentatsiooni. Lisas 2 on toodud antud töö raames loodud eestikeelsete nimedega funktsioonide teek.

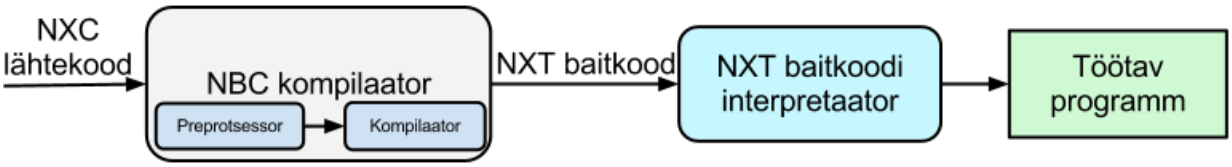
1. Programmeerimiskeel *NXC* ja selle kasutamine erinevatel platvormidel

NXC on John Hanseni poolt [4], LEGO MINDSTORMS NXT robotite programmeerimiseks loodud keel. Süntaksi poolest sarnaneb see *C* keelega [5, lk.1-2]. Sarnasustest hoolimata ei ole *NXC* üldkasutatav programmeerimiskeel, vaid on piiratud NXT baitkoodi interpretaatori võimalustega.

NXC programmide loomist ja käivitamis iseloomustavad järgmised sammud (joonis 1):

1. Programmi lähtekoodi loomine.
2. Lähtekoodi kompileerimine *NBC* (*Next Byte Codes* - mada taseme keel NXT robotite programmeerimiseks, millel baseerub ka *NXC*) kompilaatoriga, mille tulemuseks on NXT baitkood. NBC kompilaator sisaldab preprotsessorit, mis töötleb lähtekoodi enne reaalset kompileerimist.
3. NXT baitkoodi käivitamine robotis, mille tulemuseks on töötav programm.

Programmide käivitamiseks kasutatav baitkoodi interpretaator on tehase seadetega roboti mällu paigaldatud. See tähendab, et *NXC* programmeerimisega alustades ei ole vaja robotisse paigaldada uut baastarkvara ja sama roboti juhtplokis saab endiselt kasutada ka graafilist *NXT-G* keelt. Sobiva NBC kompilaatori leiab *NXC* kodulehelt [6].



Joonis 1. *NXC* programmi käivitamine

Järgnevas punktis antakse ülevaade, milline on *NXC* leksikaalne ehitus ja millised on selle programmeerimiskeele põhilised elemendid. Teises punktis antakse juhised, kuidas erinevatel operatsioonisüsteemidel seadistada arvutit nii, et sellega saaks *NXC* programmeerimisega alustada.

1.1 *NXC* keel

NXC on *C* sarnane keel LEGO MINDSTORMS NXT robotite programmeerimiseks. Antud punktis antakse ülevaade põhilistest *NXC* keele reeglitest ja elementidest.

1.1.1 *NXC* keele leksikaalne ehitus

Programmi leksikaalne ehitus määrab reeglid, kuidas kirjutada programmi koodi konkreetses keeles. Näiteks, mil viisil tuleb kirja panna muutujaid, missugused on võimalused kommentaaride lisamiseks või kuidas on üks programmi avaldis või lause teistest eraldatud. Järgnevalt kirjeldatakse, missuguste põhiliste reeglitega peaks *NXC* kirjutamisel arvestama [5, lk. 2-36]:

1. *NXC* on tõstutundlik, mis tähendab, et märksõnade, muutujate ja funktsioonide nimede kirjutamisel on oluline suur ja väike täht. Näiteks sõna `task` (tegum) peab olema alati väikeste tähtedega, kui sellega tahetakse tähistada tegumit.
2. Avaldiste lõpus on kohustuslik semikoolon (`;`).
3. Kommentaare saab kirjutada kahel viisil: ridahaaval ja üle mitme rea. Ridahaaval kommenteerimiseks kasutatakse kommenteeritava rea alguses kaht kaldkriipsu (`//`). Üle mitme rea kommenteerimiseks kasutatakse kommentaari alguses märke `/*` ja lõpus `*/`. Üle mitme rea kommenteerimisega tuleb arvestada sellega, et faili viimane rida ei tohi lõppeda kommentaari lõpuga, kuna kompilaator annab selle peale veateate.
4. Tühikute lisamine ja eemaldamine ei mõjuta programmi, kuni sõnad on üksteisest eraldatud. Näiteks avaldis `a=b;` on samaväärne avaldisega `a = b ;`.
5. Numbrilisi konstante võib kirjutada nii kümnend- kui ka kuueteistkümnendsüsteemis. Kümnendsüsteemi arv koosneb ühest või rohkemast kümnendsüsteemi numbrist. Kuueteistkümnendsüsteemi arv algab sümbolitega `0x` või `0X`, millele järgneb üks või mitu kuueteistkümnendsüsteemi numbrit. Näiteks `a = 10;` ja `a = 0x10;`.
6. Erinevatele elementidele, nagu funktsioonid ja muutujad, antavad nimed peavad algama suure või väikese algustähega. Nimes on lubatud kasutada ka alakriipsu (_).
7. Muutujate ja funktsioonide nimed ei tohi sisaldada täpitähti (õ,Õ,ä,Ä,ö,Ö,ü ja Ü).

8. Muutujad ja funktsioonid tuleb kirjeldada enne nende kasutamist. Ei ole lubatud olukord, kus programmi alguses olev tegum püüab kasutada programmi lõpus olevat funktsiooni.
9. *NXC* keel sisaldab kindla hulga märksõnu, mida ei saa kasutada muutujate ja funktsioonide nimedes. Need märksõnad on: `asm`, `bool`, `break`, `byte`, `case`, `char`, `const`, `continue`, `default`, `do`, `else`, `enum`, `false`, `float`, `for`, `goto`, `if`, `inline`, `int`, `long`, `mutex`, `priority`, `repeat`, `return`, `safecall`, `short`, `start`, `static`, `stop`, `string`, `struct`, `sub`, `switch`, `tasks`, `true`, `typedef`, `unsigned`, `until`, `void` ja `while`.

Käesolevas punktis anti ülevaade tähsamatest reeglitest, millega tuleks arvestada *NXC* programmi loomisel. Järgmises punktis antakse ülevaade *NXC* programmi struktuurist.

1.1.2 *NXC* programmi struktuur

NXC-keelne programm koosneb erinevatest plokkidest ja muutujatest. Plokkidest eristatakse tegumeid (`task`) ja funktsioone (`function`) [5, lk 9-15].

Tegumid on programmi osad, mis saavad robotis töötada paralleelsete lõimedena. Tegumeid kirjeldatakse järgneva struktuuriga:

```
task teguminimi(){
     //Tegumi sisu
}
```

Igas programmis peab olema üks peategum nimega *main*, mille täitmist alustatakse programmi käivitamisel. Ülejäänud tegumid kutsutakse välja töötava lõime sisust käsuga `start teguminimi;` ning peatatakse vastavalt käsuga `stop teguminimi`.

Funktsioon on instruktsioonide kogum, mis täidab mingit kindlat programmi osa ja seda saab vajadusel erinevatest programmi osadest välja kutsuda. Erinevalt tegumitest saab funktsioonile ette anda parameetreid. *NXC* funktsioonidel on järgmine süntaks:

```
[safecall] [inline] tagastustüüp nimi(argumendid)
{
     //Funktsiooni sisu
     [return tagastatav;]
}
```

Safecall on valikuline märksõna, mille eesmärgiks on antud funktsiooni kasutamist sünkroniseerida. Kui kaks lõime tahavad üht funktsiooni samaaegselt kasutada, siis peab teine väljakutsuja ootama, kuni funktsioon on esimese väljakutse täitnud.

Inline märksõna kasutamise korral tehakse kompilaatoris funktsiooni kasutamisel sellest koopia. Antud lahenduse puuduseks on see, et koopiate tegemisel suureneb baitkoodi maht. Märksõna kasutamata luuakse funktsioonist lõim, nagu tegumi puhul. Ühes programmis saab olla korraga kuni 265 lõime.

Tagastustüüp määrab funktsiooni poolt tagastatava väärtuse andmetüübi. Funktsioonid, mis ei tagasta midagi, on tüübiga *void*.

Argumentide list sisaldab komaga eraldatult argumente kujul `tüüp nimi`. Kasutatavateks argumentide tüüpideks on kõik *NXC* muutujate tüübid: *bool, char, byte, int, short, long, unsigned int, unsigned long, string, struct* tüübid ja erinevat tüüpi massiivid.

Muutujad on nimelised mälupesad, millele saab omistada väärtuseid. Muutujate abil saab programmis andmeid säilitada ja nendega manipuleerida. *NXC* keeles deklareeritakse muutujad andmetüübi ja selle järele lisatud, komaga eraldatud, muutujate nimedega. Soovi korral saab muutuja deklareerimisel sellele ka kohe väärtuse omistada. Järgnevalt esitatakse näiteid muutujate loomisest:

- *int x;* - deklareeritakse 16-bitine täisarvu tüüpi muutuja.
- *string y, z ;* - deklareeritakse sõne tüüpi muutujad *y* ja z.
- *long a, b=1, c;* - deklreeritakse 32-bitised täisarvu tüüpi muutujad *a*, *b* ja *c*, kus muutujale `b` omistatakse väärtus `1`.

- `bool d = test();` - deklareeritakse loogilise väärtuse tüüpi muutuja `d`, millele omistatakse funktsiooni test tagastatav väärtus.

Muutujaid saab jaotada lokaalseteks ja globaalseteks. Lokaalsed deklareeritakse koodi ploki sees. Need on kasutatavad alates deklareerimise kohast kuni konkreetse ploki lõppemiseni. Globaalsed muutujad luuakse väljaspool koodiplokke. Globaalsed muutujad on kõigi plokkide poolt kasutatavad deklareerimise kohast kuni programmi lõpuni.

1.1.3 Operaatorid

Operaatorid on vajalikud muutujate väärtuste muutmiseks ja võrdlemiseks. Peamiseks omistamismärgiks on võrdusmärk (=), mida kasutatakse kujul `muutuja = avaldis;`. *NXC* toetab ka nn täiendatud omistamist (*augmented assignment*), kus omistusmärgile lisatud tehtemärk näitab muutust sama muutuja väärtuses. Näiteks `x += y;` tähendab seda, et muutuja `x` väärtusele liidetakse muutuja `y` väärtus.

NXC keeles enimkasutatavad aritmeetilised operaatorid ja täiendatud omistustehted on järgmised [5, lk 22-23]:

1. = omistab muutujale väärtuse. Näiteks `x = y;`
2. `+, -, *, /, %` - liitmine, lahutamine, korrutamine, jagamine ja jäägiga jagamine. Näiteks `x = 5 % 2;`
3. `++` suurendab muutujat ühiku võrra. Kasutatakse kujul `x++;`
4. `--` vähendab muutujat ühiku võrra. Kasutatakse kujul `x--;`
5. `!` annab loogilist (`bool`) tüüpi muutujale vastandväärtuse. Kui `x = true;` ja `y = !x;`, siis `y = false;`
6. `+=` liidab muutujale väärtuse. Näiteks `x += y;` See on samaväärne tehtega `x = x + y;`
7. `-=` lahutab muutujast väärtuse. Näiteks `x -= y;` See on samaväärne tehtega `x = x - y;`
8. `*=` korrutab muutujat väärtusega. Näiteks `x *= y;` See on samaväärne tehtega `x = x * y;`
9. `/=` jagab muutujat väärtusega. Näiteks `x /= y;` See on samaväärne tehtega

`x = x / y;`

10. `%=` mooduloperaator (jäägi leidmiseks). Näiteks `x %= y;` See on samaväärne tehtega `x = x % y;`
11. `||=` omistab muutujale absoluutväärtuse omistatavast. Näiteks `x ||= y;`

Täiendatud omistustehete puhul ei tohi erinevate tehtemärkide vahele tühikut lisada. Kui kasutada näiteks tehet `x + = y;`, siis see ei ole korrektne ja kompilaator annab veateate.

1.1.4 Juhtstruktuurid

Programm on arvutile arusaadavate käskude jada, mida täidetakse järgemööda. Selleks, et käskude täitmise järjekorda saaks muuta, kasutatakse erinevaid juhtstruktuure, mis juhivad tavaliselt mingitele tingimustele vastavalt programmi kulgu. Üldiselt juhitakse programmi tingimustega järgnevalt: juhul kui *tingimus*, siis *tegevus*.

NXC keeles kasutatakse tingimusteks tingimusavaldisi, mis esitatakse kujul *avaldis tingimusoperaator avaldis*. Avaldiseks võib olla ka muutuja või sulgudesse pandud tingimusavaldis. Erinevad kasutatavad tingimusoperaatorid on järgmised [5, lk 36-37]:

1. == Tõene kui avaldised on võrdsed.
2. > Tõene kui esimese avaldise väärtus on teisest suurem.
3. >= Tõene kui esimese avaldise väärtus on teisest suurem või võrdne.
4. < Tõene kui esimese avaldise väärtus on teisest väiksem.
5. <= Tõene kui esimese avaldise väärtus on teisest väiksem või võrdne.
6. != Tõene kui avaldiste väärtused ei ole võrdsed.
7. && Tõene kui mõlemad avaldised on tõesed.
8. || Tõene kui vähemalt üks avaldistest on tõene.

Selleks, et *NXC* keeles programmi kulgu juhtida, kasutatakse järgnevaid kontrollstruktuure [5, 23-33]:

1. *if("tingimus"){"tegevus"}* - kui tingimus on tõene, siis tehakse tegevus

2. *if(“tingimus”){“tegevus1”} else {“tegevus2”}* - kui tingimus on tõene, siis tehakse tegevus1, vastasel juhul 2
3. *switch(“muutuja”){*
   case x: “tegevus1”
   break;
   case y: “tegevus3”
   break;
   default: “tegevus4”
   break;
   } - Põhimõttelt sarnane struktuuriga b. Paljude võimaluste puhul on antud varianti mugavam kasutada. *x* ja y on väärtused, millega *muutuja*-t võrreldakse.
4. *while(“tingimus”){“tegevus”}* - Senikaua kuni tingimus kehtib korratakse tegevust.
5. *do{“tegevus”}while(“tingimus”);* - Kõigepealt tehakse tegevus ja pärast seda kontrollitakse, kas antud tegevust korrata.
6. *for(“avaldis1”;“tingimus”;“avaldis2”){“tegevus”}* - Enne tsükli alustamist täidetakse *“avaldis1”*. Seejärel täidetakse *“tegevus”* kuni *“tingimus”* on tõene. Peale igat *“tegevus”* täitmist arvutatakse *“avaldis2”.*
7. *goto sihtpunktinimi;* - Suunab tegevuseprogrammis kohta, kus on deklareering *sihtpunktinimi;*.
8. *until(“tingimus”){“tegevus”}* - Oodatakse tingimuse tõekssaamist ja siis tehakse tegevus.
9. *break;* - Katkestab tsükli.
10. *return “väärtus”;* - Katkestab funktsiooni töö ja tagastav väärtuse.

Selles punktis tutvustati erinevaid struktuure NXC programmi töö juhtimiseks. Järgnevalt kirjeldatakse, kuidas anda juhiseid NBC kompilaatori preprotsessorile, mis töötleb programmi enne kompileerimist.

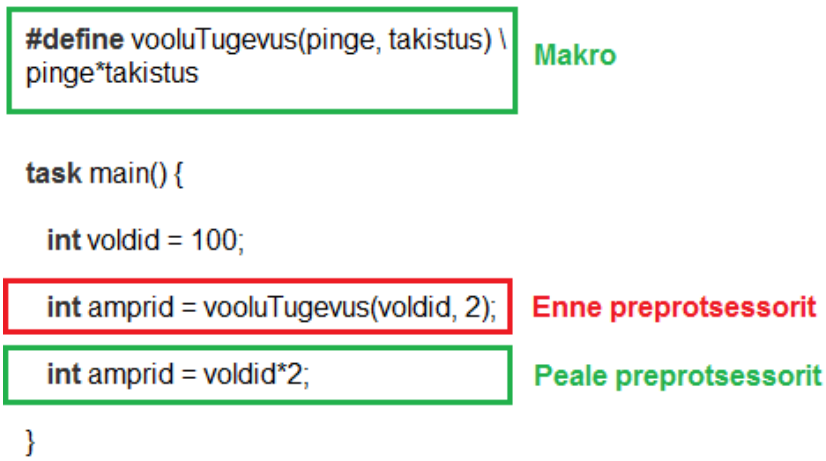
1.1.5 Preprotsessori käsud

NBC kompilaator sisaldab preprotsessorit, mis töötleb prgrammi lähteteksti faile enne, kui neid reaalselt kompileerima hakatakse. Preprotsessorile saab ette anda juhiseid lähtekoodi faili päises, märgiga `#` rea alguses. Kõige tähtsamateks *NXC* preprotsessori [5, lk 37-39] juhisteks on `#include`, `#define` ja `##`.

`#include` käsuga saab programmi laadida teisi *NXC* faile. See on kasulik siis, kui on vaja mingisuguseid funktsioone ja parameetreid mitmes programmis kasutada. Lisatava faili nimi pannakse jutumärkides `#include` käsu järele. Näiteks `#include "funktsioonid.nxc"`.

`#define` võimaldab luua makrosid. See tähendab, et programmifaili päisesse saab defineerida identifikaatoriga teksti osasid. Programmifailis sees asendatakse enne kompileerimist vastav identifikaator defineeritud teksti osaga.

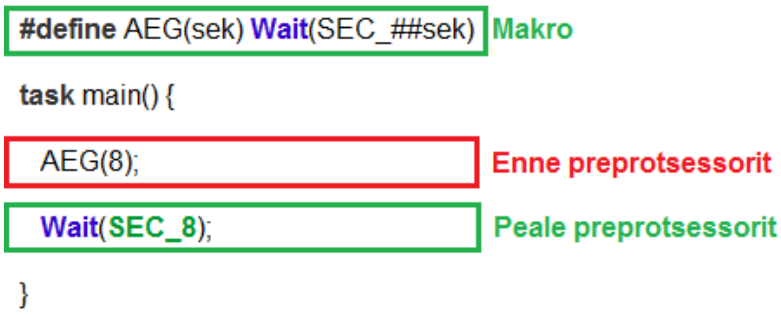
Makrosid luuakse järgneva struktuuriga: `#define identifikaator defineeritav`. Identifikaatoriks on nimi, mida kasutatakse makro asukoha määramiseks programmis. Defineeritavaks on *NXC* programmi osa, mis enne kompileerimist makro nime asemele pannakse. Makro lõpetab tavaliselt rea lõpp. Kui tegemist on mitmerealise makroga, siis pannakse rea lõppu \. Joonisel 2 on toodud näide makrost ja selle kasutamisest.



Joonis 2. Makro kasutamise näide

Makrosid on kasulik kasutada lähtekoodi selguse ja parema muudetavuse huvides. Näiteks on makrod head koht, kus defineerida globaalseid väärtusi, kuna nii on neid lihtne üles leida ja vajadusel ka muuta.

`##` on käsk, mida saab kasutada erinevate programmi teksti osade kokku liitmiseks, kuna preprotsessor asendab `##` mitte kui millegagi. Seda saab näiteks kasutada juhul kui makro funktsioonis on kahest parameetrist vaja luua kolmas. Joonisel 3 on toodud näide `##` kasutamisest makros.



Joonis 3. *##* kasutamine

`##` kasutamisel tuleb arvestada, mis juhtub makrosse parameetrite lisamisel. Kui kui kasutada näiteks joonisel 3 kujutatud makrot kujul `int x = 8; AEG(x);`, siis antud juhul on preprotsessori tulemuseks `Wait(SEC_x);`, mis annb kompileerimisel veateate kui enne makro kasutamist ei ole defineeritud täisarvtüüpi muutujat `SEC_x`.

1.1.6 *NXC* rakendusliides

NXC keelel on rakendusliides [5, lk 2] (*API - Application Programming Interface*), mis määrab ära, milliseid standardseid funktsioone ja konstante saab programmides kasutada. Rakendusliidese peamiseks kasutuskohaks on programmi töö juhtimine ning sisend- ja väljundseadmete haldamine.

Programmi kirjutamisel tuleks arvestada sellega, et rakendusliidese funktsioone ei kirjutataks üle või ei püütaks luua mõnda rakendusliidese konstandi nimega elementi. Kui püüda luua

juba reeglistikus olemasolevat elementi, siis tõenäoliselt tekib kompileerimisel viga. Näiteks ei saa defineerida muutujat nimega `NA` kuna antud tähepaarile vastab arv 65535.

Järgnevalt on toodud mõned näited *NXC* rakendusliidese funktsioonidest ja konstantidest:

- `Wait("aeg");` - Katkestab tegumis järgnevate käskude täitmise etteantud ajaks (millisekundit).
- `SEC_3` - Konstant, mis vastab 3000 millisekundile.
- `StartTask("tegum");` - Käivitab tegumi.
- `OnFwd("mootorid", "kiirus");` - Pöörab etteantud mootoreid etteantud kiirusega.
- `OUT_A` - Konstant, mis tähistab porti A.

Kõigist ülejäänud rakendusliidese elementidest saab ülevaate *NXC* manuaalist [5].

1.1.7 Näidisprogramm

Joonisel 4. on kujutatut *NXC* programmi näidis koos kommentaaridega. Tegemist on lihtsa maadeavastaja robotiga, mis keerab enne takistust kõrvale. Toodud näidise eesmärgiks ei ole anda antud probleemile parimat lahendust vaid näidata võimalikult palju *NXC* keele ja rakendusliidese erinevaid osi.

Joonisel 4 on kujutatud keele märksõnad rasvases ja mustas, rakendusliidese funktsioonid rasvases ja sinises ning konstandid rasvases ja rohelises kirjas. Kommentaarid on halli tooni.

```
1  #include "nxcEesti.nxc" // Välise faili lisamine
2  #define KIIRUS 20       // Makro defineerimine
3  /* Programmi eesmärk on see, et robot liigub otse ning
4  takistuse saabumisel keerab kõrvale ja liigub edasi*/
5  bool majakas = true;  // Globaalne loogika tüüpi muutuja paralleelsete protsesside juhtimiseks
6  int minkaugus = 50;   // Globaalne täisarv tüüpi muutuja.
7                        // Maksimaalne vahemaa takistuseni, mis sunnib robotit keerama
8  task liigu(){         // Alamülesanne otse liikumiseks
9    while(true){        // Lõputu tsükkel
10     if(majakas){      // Tingimuslause, mis kontrollib, kas majakas on tõene
11       OnFwdReg(OUT_BC, KIIRUS, TRUE); // Robot liigub otse edasi
12     } else {          // Tegevus, kui tingimus ei vastanud tõele
13       Off(OUT_B);     // Peatan mootori B, et robot keeraks
14     }
15   }
16 }
17
18 task m66da(){        // Alamülesanne
19   int takistus;       // Lokaalne muutuja mõõdetud takistuse kaugusele
20   SetSensorLowspeed(IN_4);  // sensori initsialiseerimine sisendpordil 4
21   while(true){                          // Tsükkel
22     takistus = SensorUS(IN_4);          // Loeb ultraheli sensorit
23     ClearScreen();                      // Tühjendab ekraani
24     NumOut(1, 1, takistus);             // Takistuse kaugus ekraanile
25     if(takistus > 1 && takistus <= minkaugus){ // Mitu tingimust
26       majakas = false;      // Majakas pannakse kinni, et ülesanne otse robotit ei liigutaks
27     } else {                // Vastasel juhul
28       majakas = true;       // Majaka vabastamine
29     }
30   }
31 }
32
33 task main(){                // Peaülesanne
34   Precedes(liigu, m66da);   // Alustan kaks paralleelset protsessi
35 }
```

Joonis 4. *NXC* keele näidisprogramm

Järgnevalt vaadeldakse, kuidas erinevatel operatsioonisüsteemidel seada üles töökeskkond *NXC* kasutamiseks.

1.2 *NXC* paigaldamine

Nii Windows, Linux kui ka Mac OS operatsioonisüsteemile on olemas oma kompilaator *NXC* keeles kirjutatud programmide kompileerimiseks baitkoodi. Olemasolevaid kompilaatoreid saab kasutada teiste programmide poolt või käsurearakendusena üldlevinud tekstiredaktoriga

tehtud programmiteksti kompileerimiseks. Kompilaatorite lisavõimalusteks on ka kompileeritud programmi robotisse toimetamine üle *USB* või sinihamba ühenduse.

Järgnevalt kirjeldatakse erinevatele operatsioonisüsteemidele *NXC* kasutamiseks vajalike lisaprogrammide paigaldamist ning roboti ja arvuti vahelise ühenduse loomist. Samuti selgitatakse, kuidas käsurealt esimene programm kompileerida ja robotisse saata. Tarkvara paigaldamiseks peavad kasutajal olema administraatori õigused.

1.2.1 Windows

Windows operatsioonisüsteemi korral tuleks NXT ühendamiseks arvutisse paigaldada vajalik draiver. Seda saab teha järgneval viisil:

1. LEGO kodulehelt [7] tuleb alla laadida sobiv draiver. Antud lehel tuleb valida *Downloads* ja *PC*.
2. Pärast allalaadimist tuleb fail lahti pakkida.
3. Tekkinud kaustast tuleb käivitada programm *setup.exe*. Ekraanijuhiseid järgides installeeritakse arvutisse vajalik lisaprogramm.

Peale draiveri installeerimist tuleb arvutisse laadida ka sobiv kompilaator:

1. *NXC* koduleheküljelt [6] tuleks leida uusima stabiilse versiooniga NBC kompilaator (antud töö kirjutamise ajal versioon NBC 1.2.1 r4).
2. Windows operatsioonisüsteemi jaoks tuleb alla laadida *.zip* laiendiga fail.
3. Allalaetud fail tuleb lahti pakkida.
4. Lahtipakitud kausta sisu võiks kopeerida kohta, kust kasutaja selle hiljem kergesti üles leiaks aga tähelepanematusest ära ei kustutaks. Näiteks kausta *Dokumendid*.

Pärast vajaliku draiveri paigaldamist ja kompilaatori salvestamist saab neid testida järgnevalt:

1. Tuleks käivitada NXT ja ühendada see *USB* kaabli abil arvutiga.
2. Tekstiredaktoriga (näiteks Notepad) võiks luua testfaili *test.nxc* ning lisada sellesse järgnev sisu:

```
task main(){
          TextOut(0, LCD_LINE3, "Tere NXT");
```

```
    Wait(10000);
}
```

3. Kui fail *test.nxc* ja kompilaator on paigutatud kausta *Dokumenid* ja terminalis ollakse asukohas *C:\Kasutajad\<kasutajanimi>\Dokumendid>*, siis tuleb kompileerimiseks käsurealt käivita järgnev käsk: `nbc.exe test.nxc -T=NXT -S=usb -r`.

Kui käsu käivitamisel toimib kõik õigesti, siis ilmub NXT ekraanile järgnev tekst: „Tere NXT“.

1.2.2 Linux (Ubuntu 11.10)

Sarnaselt operatsioonisüsteemile Windows, tuleb ka Linuxi puhul arvutisse laadid NBC kompilaator, mille leiab *NXC* kodulehelt [6]. Kompilaatori saab terminalis paigaldada järgnevalt [8]:

1. Kompilaatori salvestamine:
   ```
   $ wget http://downloads.sourceforge.net/bricxcc/nbc-1.2.1.r4.tgz
   ```
2. Allalaetud kompilaatori faili lahti pakkimine:
   ```
   $mkdir nbc
   $tar xzf nbc-1.2.1.r4.tgz -C nbc
   $cd nbc/NXT
   ```
3. Võiks testida, kas kompilaator töötab:
   ```
   $ ./nbc
   ```
   Kui kompilaator töötab, siis kuvatakse ekraanile järgnev tekst:
   Next Byte Codes Compiler version 1.2 (1.2.1.r4, built Tue Mar 15 16:10:49 CDT 2011)
   Copyright (c) 2006-2010, John Hansen
   Use "nbc -help" for more information.
4. Selleks, et lihtsustada kompilaatori hilisemat kasutamist süsteemi programmina tuleks see asetada kausta */usr/local/bin/*:
   ```
   $ sudo mv nbc/usr/local/bin/
   ```

Linux operatsioonisüsteemile ei ole draiverit NXT roboti *USB* abil arvutiga ühenduse saamiseks. Selleks, et kompilaator siiski saaks programmid robotisse saata, tuleb tegutseda järgnevalt:

1. Tuleb luua kasutajate grupp nimega legonxt. Sellesse hakkavad kuuluma kasutajad, kes saavad robotit arvutiga ühendada:
   $ sudo addgroup legonxt
2. Loodud gruppi võiks kasutaja enda kasutajanime lisada (kasutajanime leiab käsuga *echo ${USER}*):
   $sudo adduser <minukasutajanimi> legonxt
3. Tuleb luua fail nimega 45-legonxt.rules kausta */etc/udev/rules.d*. See kirjeldab arvuti seadmehaldurile ühendatavat seadet:
   $ sudo nano /etc/udev/rules.d/45-legonxt.rules
4. Antud faili tuleb lisada järgnev sisu:
   ```
   SUBSYSTEM=="usb_device",
   ACTION=="add",
   SYSFS{idVendor}=="0694",
   SYSFS{idProduct}=="0002",
   SYMLINK+="legonxt-%k",
   GROUP="legonxt",
   MODE="0660",
   RUN+="/etc/udev/legonxt.sh"
   ```
5. Kausta */etc/udev/* tuleb luua fail nimega *legonxt.sh* . Selles failis määratakse ära, et NXT roboti ühendamisel kuulub see legonxt gruppi.
   $ sudo nano /etc/udev/legonxt.sh
6. Loodud faili tuleb lisada järgnev sisu:
   ```
   #!/bin/bash # GROUP=legonxt
   if [ "${ACTION}" = "add" ] && [ -f "${DEVICE}" ]
   then
   chmod o-rwx "${DEVICE}"
   chgrp "${GROUP}" "${DEVICE}"
   chmod g+rw "${DEVICE}"
   ```

```
fi
```

7. Viimatiloodud failile tuleb anda käivitumisõigus:
   ```
   $ sudo chmod a+x /etc/udev/legonxt.sh
   ```
8. Loodud seadistuste rakendamiseks tuleks arvuti taaskäivitada.

Kompilaatori ja *USB* ühenduse testimiseks võiks tegutseda järgnevalt:

1. NXT peab *USB* kaabli abil arvutiga ühendama ja käivitama.
2. Tuleks luua testprogramm, näiteks *tere.nxc*:
   ```
   $ nano tere.nxc
   ```
3. Sellesse võiks lisada järgneva sisu:
   ```
   task main(){
        TextOut(0, LCD_LINE3, "Tere NXT");
        Wait(10000);
   }
   ```
4. Loodud programmis saab kompileerida ning robotisse käivitamiseks saata järgneva käsuga:
   ```
   $nbc tere.nxc -sm- -r -S=usb
   ```

Kui kõik õnnestub, siis ilmub roboti ekraanile järgnev kiri: „Tere NXT“.

1.2.3 Mac OS

NXT ühendamiseks Mac OS operatsioonisüsteemiga arvutiga peab arvutisse paigaldama vajaliku draiveri, mille leiab LEGO kodulehelt [7]. Sealt tuleb valida *Downloads* ja *MAC*. Pärast allalaadimis tuleb salvestatud faili lahti pakkida ning installeerida selles olev programm lähtudes ekraanijuhistest.

Peale draiveri installeerimist peab arvutisse laadima ka NBC kompilaatori, mille saab *NXC* kodulehelt [6]. Sealt leiab operatsioonisüsteemile Mac OS sobiva kompilaatori, millel on laiendiks *.osx.tgz*. Kompilaatori paigaldamiseks terminalis võiks tegutseda järgnevalt:

1. Võiks luua kausta, millesse lisatakse kompilaator:
   ```
   $mkdir nbc
   ```
2. Allalaetud kompilaator tuleks loodud kausta lahti pakkida:

```
$tar xzf nbc-1.2.1.r4.tgz -C nbc
```

3. Kompilaatori töötamist saab testida järgnevalt:

   ```
   $cd nbc/NXT
   $./nbc
   ```

   Kui kompilaator töötab, siis kuvatakse ektaanile järgnev tekst:

   „*Next Byte Codes Compiler version 1.2 (1.2.1.r4, built Tue Mar 15 16:10:49 CDT 2011)*

   Copyright (c) 2006-2010, John Hansen

   Use "nbc -help" for more information. „

4. Selleks, et lihtsustada kompilaatori hilisemat kasutamist, võiks selle kasutaja programmide kausta lisada:

   ```
   sudo mkdir -p /usr/local/bin
   sudo cp ./nbc /usr/local/bin
   ```

Kompilaatori ja *USB* ühenduse testimiseks võiks tegutseda järgnevalt:

1. NXT peaks *USB* kaabli abil arvutiga ühendama ja käivitama.
2. Tuleks luua testprogramm *tere.nxc*:

   ```
   $ pico tere.nxc
   ```

3. Sellesse võiks lisada järgneva sisu:

   ```
   task main(){
        TextOut(0, LCD_LINE3, "Tere NXT");
        Wait(10000);
   }
   ```

4. Loodud programmi saab kompileerida ning robotisse käivitamiseks saata järgneva käsuga:

   ```
   $nbc tere.nxc -sm- -r -S=usb
   ```

Kui kõik õnnestub, siis ilmub roboti ekraanile järgnev kiri: „Tere NXT“.

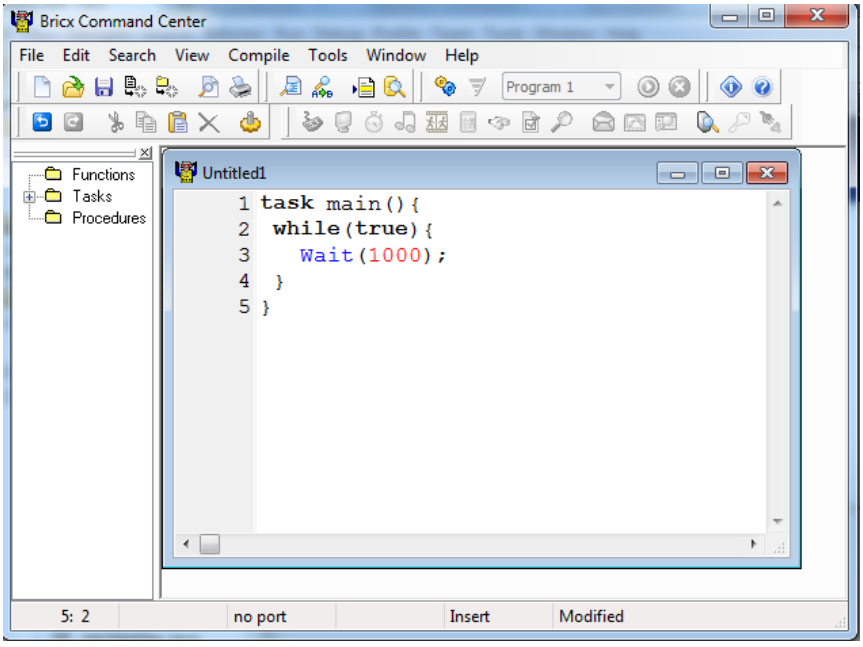
Käesoleva peatükiga kirjeldati, kuidas erinevate operatsioonisüsteemide korral teha vajalikud seadistused roboti ühendamiseks arvutiga. Samuti kirjeldati, kuidas ilma spetsiaalse arenduskeskkonnata *NXC* programme kirjutada, kompileerida ja robotisse saata.

2. Olemasolevad vahendid *NXC* kasutamiseks

Käesolevas peatükis tutvustatakse erinevatele platvormidel olemasolevaid rakendusi, mis võimaldavad *NXC* keeles programmide koostamist ja kompileerimist. Kirjeldatud vahendid asuvad kas BricxCC kodulehel [3] või on antud töö autori arvates veebist leitud lahendustest parimad.

2.1 Windows

Enimlevinud vahend Windows platvormil *NXC* keele kasutamiseks on Bricx Command Center (BricxCC) [3]. Joonisel 5 on kujutatud BricxCC peamine kasutajaliides.



Joonis 5. Bricx Command Center kasutajaliides.

BricxCC on integreeritud programmeerimiskeskkond (*IDE - Integrated Development Environment*) erinevate LEGO MINDSTORMS robotite programmeerimiseks. Lisaks *NXC* keelele on selles toetatud veel *NQC* (*Not Quite C*) ja *NBC*.

NQC on süntaksi poolest sarnane *NXC* keelega. See on mõeldud vaid LEGO RCX robotite programmeerimiseks [19].

Programmis BricxCC värvitakse programmi teksti vastavalt sellele, kas tegemist on arvu, sõne, *NXC* rakendusliidese funktsiooni, kommentaari, märksõna või konstandiga. Programmi koostamist lihtsustab mõnevõrra veel hulga valmis mallide olemasolu, mida saab lihtsalt olemasolevasse programmiteksti lisada. Nendeks mallideks on väiksemad lõigud, nagu tingimuslaused või keele rakendusliidese funktsioonid koos märksõnadega, mille peab kasutaja sobivalt muutma. Märksõnad viitavad tavaliselt sellele, millega see tuleks asendada. Kui mingis mallis on näiteks märksõna "port", siis tuleb see asendada roboti mingisugusele pordile vastava konstandiga. Valmis programmi on võimalik kompileerida, kohe robotisse saata ning käivitada.

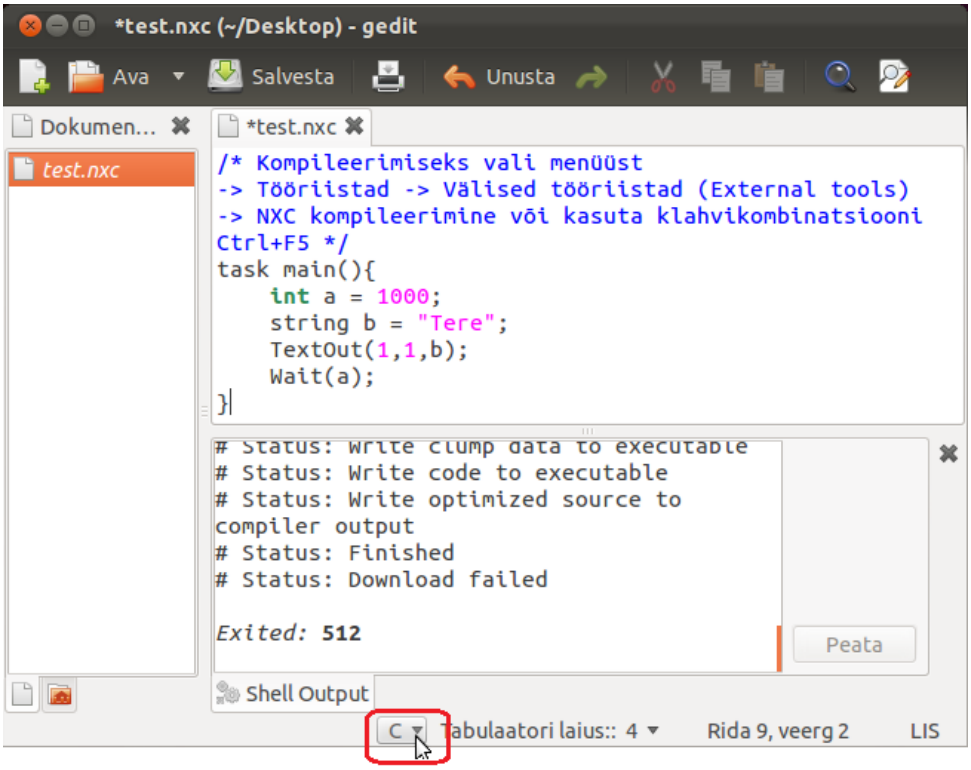
Lisaks programmiteksti redigeerimise ja kompileerimise võimalusele omab BricxCC veel hulga lisavõimalusi NXT robotiga töötamiseks. Nendeks võimalusteks on näiteks NXT mootorite kaugjuhtimine, robotis olevate failide kuvamine ja kustutamine ning seadme tarkvara uuendamine.

BricxCC paigaldamiseks tuleb see rakenduse kodulehelt [3] alla laadida ja installeerida lähtudes ekraanijuhistest. Selleks, et loodavaid *NXC* programme oleks võimalik ka edukalt robotisse saata, tuleb arvuti ja roboti vahelise ühenduse saamiseks LEGO MINDSTORMS kodulehelt [9] arvutisse laadida ja installeerida Fantom Driver [7].

Üheks vahendiks, millega saaks Windows keskkonnas *NXC* keelt kasutada ilma, et peaks kompileerimiseks käsurida kasutama on Notepad++ [12]. Antud rakendust saab seadistada nii, et sellega on võimalik kirjutatud programme kompileerida ja robotis käivitada ilma käsurida kasutamata [13]. Lisaks on võimalus sellele lisada *NXC API* tugi ja osalist lähtekoodi värvimist [14].

2.2 Linux (Ubuntu 11.10)

Linux platvormil võimaldab *NXC* kasutamist näiteks tekstiredaktor Gedit (joonis 6), mis tuleb antud operatsioonisüsteemi installeerimisega kaasa.



Joonis 6. Gedit peale seadistamist

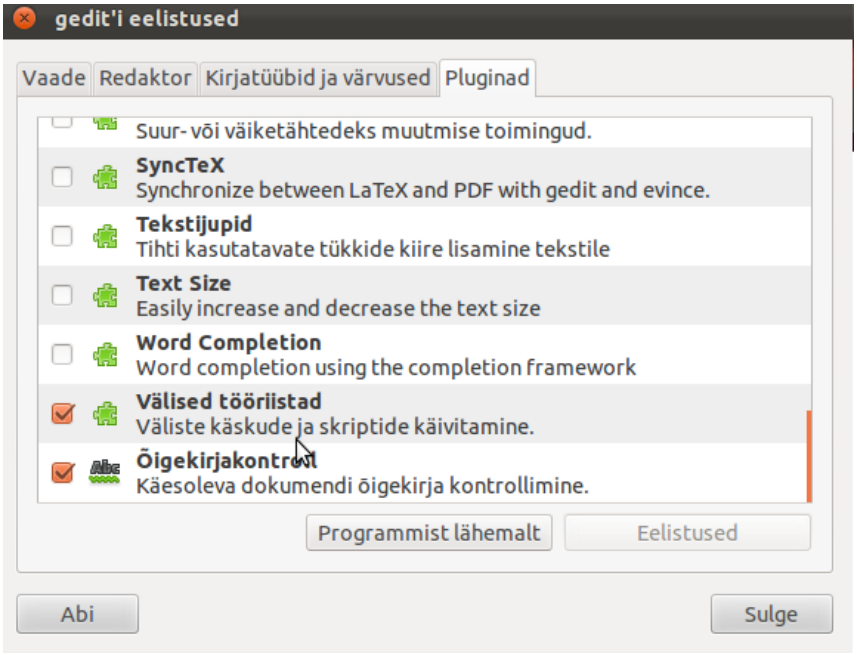
Enne programmeerimisega alustamist on vajalik kompilaatori paigaldamine ja arvuti tuleks seadistada nii, et see saaks ühendust robotiga (pt 1.1.2).

NXC ei kuulu programmi Gedit toetatud keelte hulka. Kuna *NXC* on *C* keele sarnane, siis aitab kasutajat ka *C* koodi värvimise stiil. Seda saab muuta joonisel 6 punase kastiga märgitud kohast.

Selleks, et Gedit võimaldaks otse redigeeritavat faili kompileerida ja robotisse saata, tuleks tegutseda järgnevalt (vajab administraatori õiguseid) [15]:

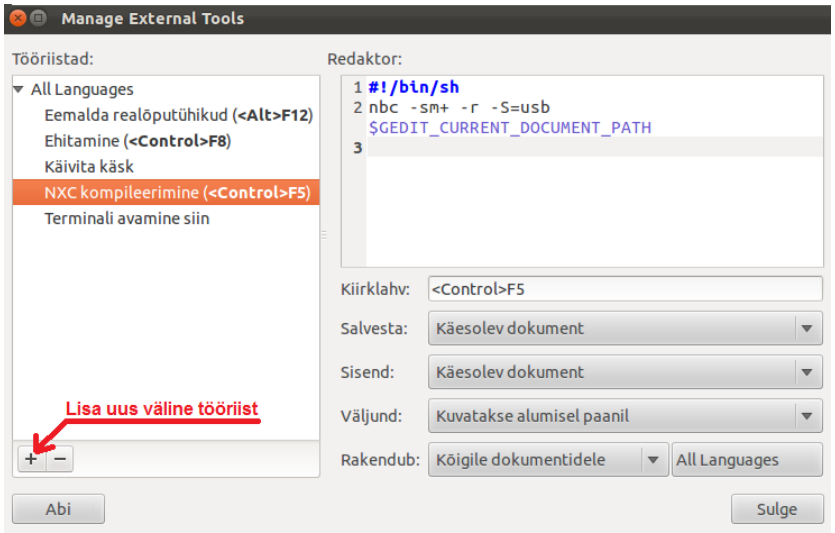
1. Kui seda ei ole varem tehtud, siis tuleb terminalis instatalleeri Gedit lisaprogrammid:
   `sudo apt-get install gedit-plugins`

2. Tuleb käivitada Gedit ning valida menüüst *Muuda* ja *Eelistused* ning aken *Pluginad* (joonis 7).



Joonis 7. Gedit Lisaprogrammid.

3. Avanenud aknast tuleb märkida *Välised tööriistad*.
4. Programmi menüüst tuleb valida *Tööriistad* ja *Halda väliseid tööriistu* (*Manage external tools*).
5. Avanenud aknas tuleb lisada uus tööriist (joonis 8).



Joonis 8. Välise tööriista määramine.

6. *Redaktor* aknasse võiks lisada järgneva sisu:

```
#!/bin/sh
 nbc -sm- -r -S=usb $GEDIT_CURRENT_DOCUMENT_PATH
```

See määrab ara, et nbc kompilaator kompileerib redigeeritava faili ja saadab selle üle *USB* ühenduse arvutisse, kus see käivitatakse.

7. *Kiirklahv* väljale võiks lisada kasutajale sobiva otsetee klahvikombinatsiooni *NXC* programmi kompileerimiseks ja robotisse saatmiseks. Näiteks <*Control*>*F5*.
8. Ülejäänud valikud võiks teha vastavalt joonisel 8 kujutatule. Need määravad, et enne kompileerimist fail salvestatakse, kompileeritakse hetkel aktiivne redigeeritav fail, tööriista rakendamise tulemus kuvatakse eraldi paneelil redigeeritava teksti all ning tööriista saab rakendada igat tüüpi failidele.

Tehtud seadistuste testimiseks võiks tegutseda järgnevalt:

1. Võiks luua faili *Test.nxc*, millesse tuleks lisada järgneva sisu:

```
task main(){
      TextOut(0, LCD_LINE3, "Tere NXT");
      Wait(10000);
      }
```

2. Tuleks ühendada NXT robot *USB* kaabli abil arvutiga ning vajutage Gedit redigeerimise vaates varemsisestatud klahvikombinatsiooni või valige menüüst *Tööriistad*, *Välised tööriistad* ning *NXC kompileerimine*.

Kui kõik õnnestus, siis ilmus roboti ekraanile kiri “Tere NXT”.

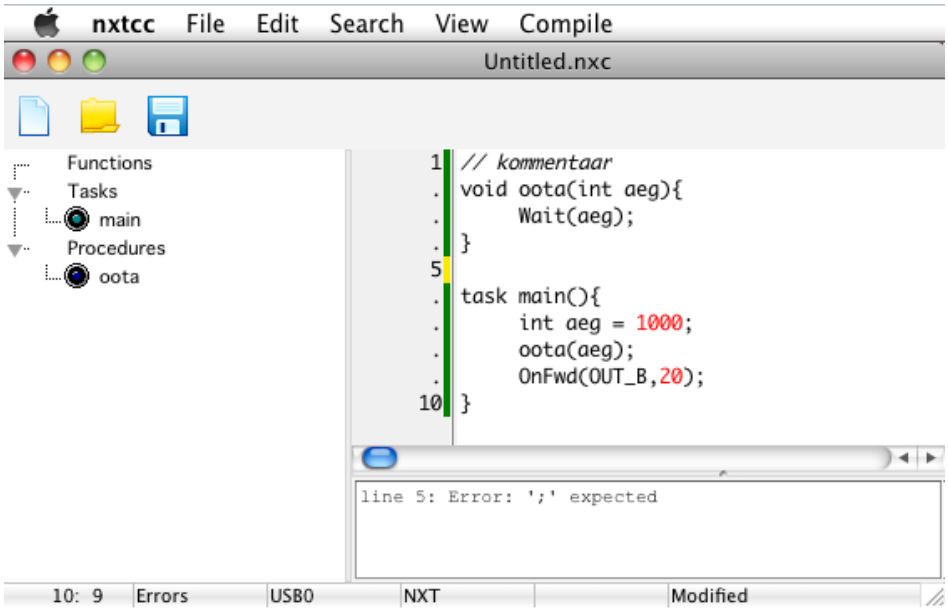
2.2 Mac OS

Mac OS operatsioonisüsteemile on sobivaks rakenduseks *NXC* kasutamiseks NeXT Command Center (NXTCC) [16] (joonis 9). Antud programmi kasutamiseks tuleb eelnevalt arvutisse installeerida Fantom Driver. Keskkonna käivitamiseks ei ole vaja programmi eelnevalt installeerida.

NXTCC omab suurt osa funktsionaalsusest [16], mis on ka operatsioonisüsteemi Windows *NXC* programmil Bricx Command Center. Erinevateks tööriisatadeks on näiteks roboti

mootorite otsene juhtimine, diagnostika, ekraanipildi kuvamine, programmide haldamine, mälu tühjendamine, baastarkvara uuendamine, välja lülitamine ning *NXC* koodiredaktor (joonis 10) koos kompileerimise ja loodud programmi robotisse saatmise võimalusega.

Joonis 9. NeXT Tools peamine kasutajaliides



Joonis 10. NeXT Tools koodiredaktor.

Koodiredaktor on mõningate vigade ja puudustega [16]. Rakendusel on näiteks järgnevad kitsaskohad:

1. Teksti värvimine on puudulik. Eristada saab vaid numbreid ja kommentaare.

2. Menüüst saab valida küll mallide vaate kuvamist, aga see on tühi ja täiendamise võimalus puudub.
3. Mõned nupud ja menüü valikud ei tööta iga kord.

Kirjeldatud programmist on versioon ka Linux platvormile. NXTCC Linux versioonis ei õnnestunud antud töö autoril loodud *NXC* programmi salvestada ega kompileerida. Seetõttu ei ole antud rakendust ka eelmises peatükis mainitud.

Käesolevas peatükis anti ülevaade erinevatele operatsioonisüsteemidele olemasolevatest rakendustest, millega saab *NXC* keelt kasutada. Järgmises peatükis kirjeldatakse antud töö raames loodud platvormiülesest keskkonda *NXC* kasutamiseks.

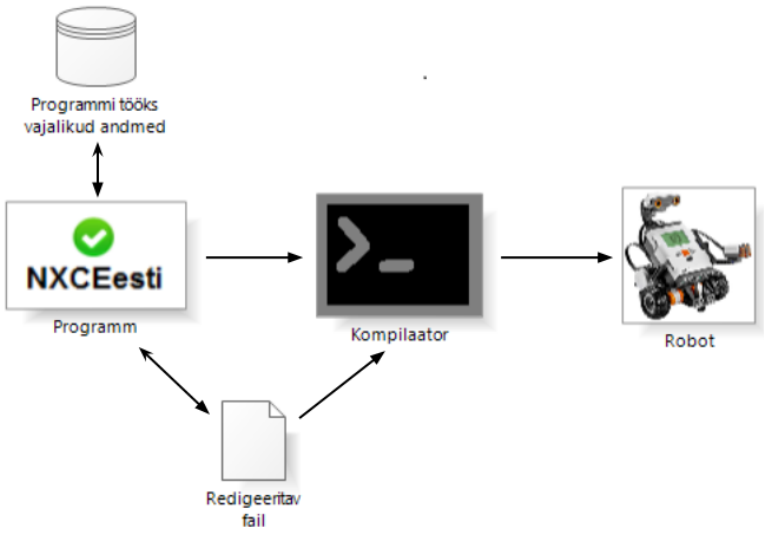
3. NXCEesti

Käesolevas peatükis antakse ülevaade antud töö raames loodud *NXC* kasutamise keskkonnast NXCEesti.

3.1 Ülesande püstitus

Eesmärgiks on luua nii Windows, Linux kui ka Mac OS operatsioonisüsteemil töötav eestikeelne keskkond *NXC* kasutamiseks. Tarkvaraga peab saama kasutada põhilisi tekstiredaktori võimalusi ning igale platvormile sobivat kompilaatorit redigeeritava programmi kompileerimiseks ja robotisse saatmiseks.

NXCEesti üldist tööpõhimõtet kirjeldab joonis 11. Programm kasutab oma tööks andmeid, mis on salvestatud tekstifailidesse, mida on kasutajal võimalik ka iseseisvalt muuta. Rakendusega saab redigeerida *NXC*-keelseid tekstifaile ning anda eraldiseisvale kompilaatorile juhiseid ja korraldusi neid faile kompileerida ning robotisse saata.



Joonis 11. Programmi tööpõhimõte.

Erinevatele platvormidele on oma kompilaator. Õige kompilaatori kasutamise eest vastutab programm. NXCEesti tuvastab operatsioonisüsteemi ning valib selle järgi sobiva kompilaatori.

3.2 Programmile esitatud nõuded

Koostöös juhendajaga, kes on ka robotiringi juhendajaga, ning lähtudes peatükis 2 vaadeldud programmidest, esitati uuele *NXC* kasutamise keskkonnale järgnevad nõuded:

Funktsionaalsed nõuded

1. Programmiga peab olema võimalus *NXC* faile luua, avada, salvestada, redigeerida ja kompileerida.
2. Kasutajal peab olema võimalus uusi värvitavaid sõnu lisada.
3. Programmiga peab olema võimalus kasutada eestikeelseid standardfunktsioone.
4. Kasutajal peab olema võimalus standardfunktsioone lisada ja muuta.
5. Programmis peab olema võimalus kasutada malle.
6. Kasutajal peab olema võimalus malle lisada ja muuta.
7. Mallidele peab olema võimalus lisada täiendavat lisainfot, mida kuvatakse eraldi paneelil.
8. Koodi redigeerimisel peab olema võimalus viimati tehtud tegevused unustada ja taastada.
9. Kasutajal peab olema võimalus kompileerimise seadeid muuta. See tähendab, et ta peab saama määrata, kas kopilaatori väljundis kuvatakse hoiatusi ja staatuse teateid ning kuidas loodud programm robotisse toimetatakse.
10. Redaktor peab võimaldama kasutada funktsioone lõika, kopeeri ja aseta.
11. Programmi teksti redigeerimisel peab olema võimalus mõlemas suunas mitut rida korraga tabuleerida.
12. Kasutajal peab olema võimalus teksti suurust muuta.
13. Kasutajal peab olema võimalus kogu programmi algseaded taastada.
14. Kasutajal peab olema võimalus erinevat tüüpi sõnade värvi muuta.
15. Mac OS operatsioonisüsteemil peab saama klaviatuuri otsetee funktsioone kasutada käsu (*Command*) klahvi abil.

Mittefunktsionaalsed nõuded

1. Süsteem peab töötama sarnaselt nii Linux, Windows kui ka Mac OS platvormil.

2. Teksti redigeerimisel peab programm värvima sisestatud sõnad vastavalt nende tüübile (konstandid, rakendusliidese funktsioonid, kommentaarid, jutumärkides olevad sõnad ja märksõnad).
3. Kompileerimisel peab olema võimalus näha kompilaatori väljundit.
4. Kasutajaliides peab olema eestikeelne.

3.3 Kasutatud vahendid ja tehnoloogiad

Rakenduse loomisel seati eesmärgiks, et see peab olema platvormiülene. Seetõttu kasutati arenduseks *Java* programmeerimiskeelt. Võimalikeks alternatiivideks olid ka *Python* ja *C*. Platvormiülesteste rakenduste loomiseks sobivate keelte hulgast valiti *Java* järgnevatel põhjustel:

1. Üks programm töötab kõigil platvormidel.
2. Antud keel sobib tekstiredaktori loomiseks.
3. Võimalikest variantidest oli antud töö autor Java keelega kõige tuttavam.
4. Hilisema arenduse jaoks leiduks suure tõenäosusega ka teisi arendajaid.

Arendusvahendiks valiti Net Beans IDE 1.7.1 [17], mis lihtsustas kasutajaliidese loomist. Abimaterjalina on kasutatud peamiselt Java SE 6 rakendusliidese spetsifikatsiooni (*Java Standard Edition 6 API Specification*) [18] ja mõningaid veebiallikaid, mis on viidatud loodud programmi lähtekoodis.

Suur osa mallidest ja märksõnadest, mis ei ole antud töö raames loodud tõlked, on üle võetud programmist BricxCC. Antud töö autor eemaldas mõningad mallid, mis reaalsel testimisel ei töötanud.

3.4 Nõuded arvutile NXCEesti kasutamiseks

NXCEesti kasutamiseks peavad kasutaja arvutil olema täidetud järgnevad nõuded:

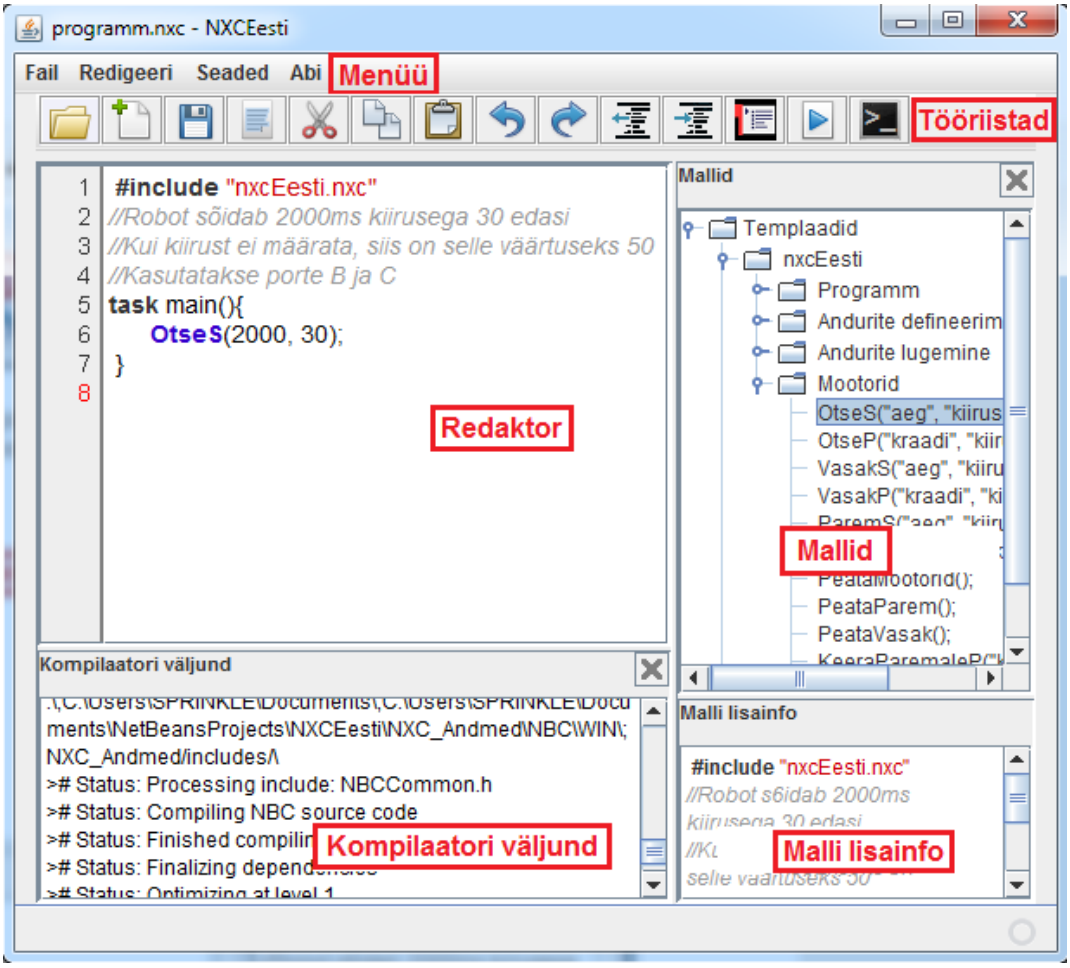
- Vähemalt 27MB vaba kõvakettaruumi.
- Vähemalt 70MB vaba operatiivmälu.
- Vähemalt üks vaba *USB* 2.0 port roboti ühendamiseks.

- Operatsioonisüsteem Mac OS, Linux või Windows.
- *JRE 6* (*Java Runtime Environment 6*) [19] või uuem.
- Windows ja Mac OS operatsioonisüsteemi puhul peab eelnevalt olema paigaldatud Fantom Driver roboti ühendamiseks arvutiga. Draiveri leiab LEGO kodulehelt [7].

Kui kasutaja arvuti vastab eelnimetatud nõuetele, siis võib antud magistritööga kaasas olevalt *CD* plaadilt arvutisse kopeerida kausta *Programm* alamkaust *NXCEesti*. Linux operatsioonisüsteemi puhul peab kopeeritud kaustast käivitama faili *LinuxInstall.sh*, mis teeb arvutis vajalikud muudatused NXT robotiga ühenduse saamiseks ja annab programmile ja kompilaatorile õigused käivitumiseks. Fail *NXCEesti.jar* on programm, mis tuleb käivitada, et loodud rakendust kasutada.

3.5 Kasutajaliides

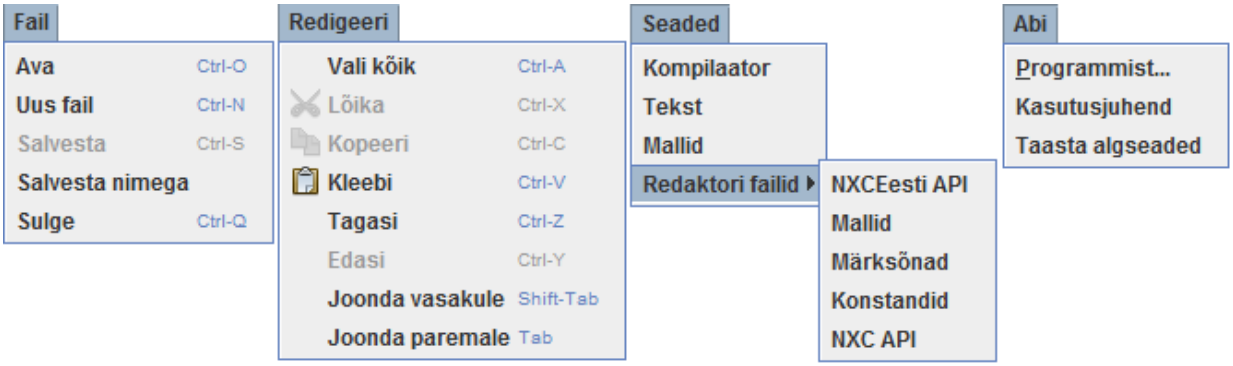
Joonisel 12 on kujutatud peamine NXCEesti kasutajaliidese vaade. Järgnevalt kirjeldatakse kasutajaliidese põhilisi osi.



Joonis 12. NXCEesti kasutajaliides

Menüü

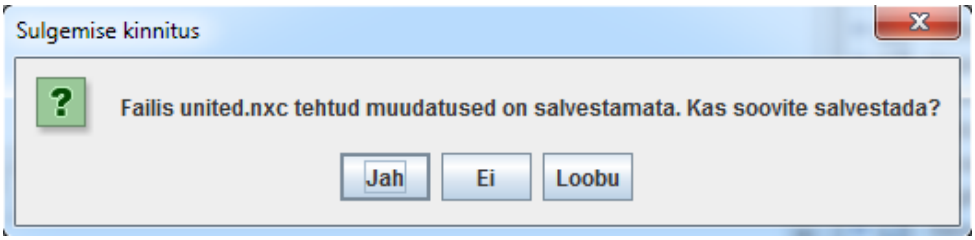
NXCEesti menüüd (joonis 13) sisaldavad valikuid, mis on vajalikud failide loomiseks, redigeerimiseks ning programmi seadete muutmiseks. Kui elemendil on klaviatuurilt otsetee, siis on see kuvatud antud valiku järel.



Joonis 13. Alammenüüd.

Järgnevalt kirjeldatakse menüüde valikuid:

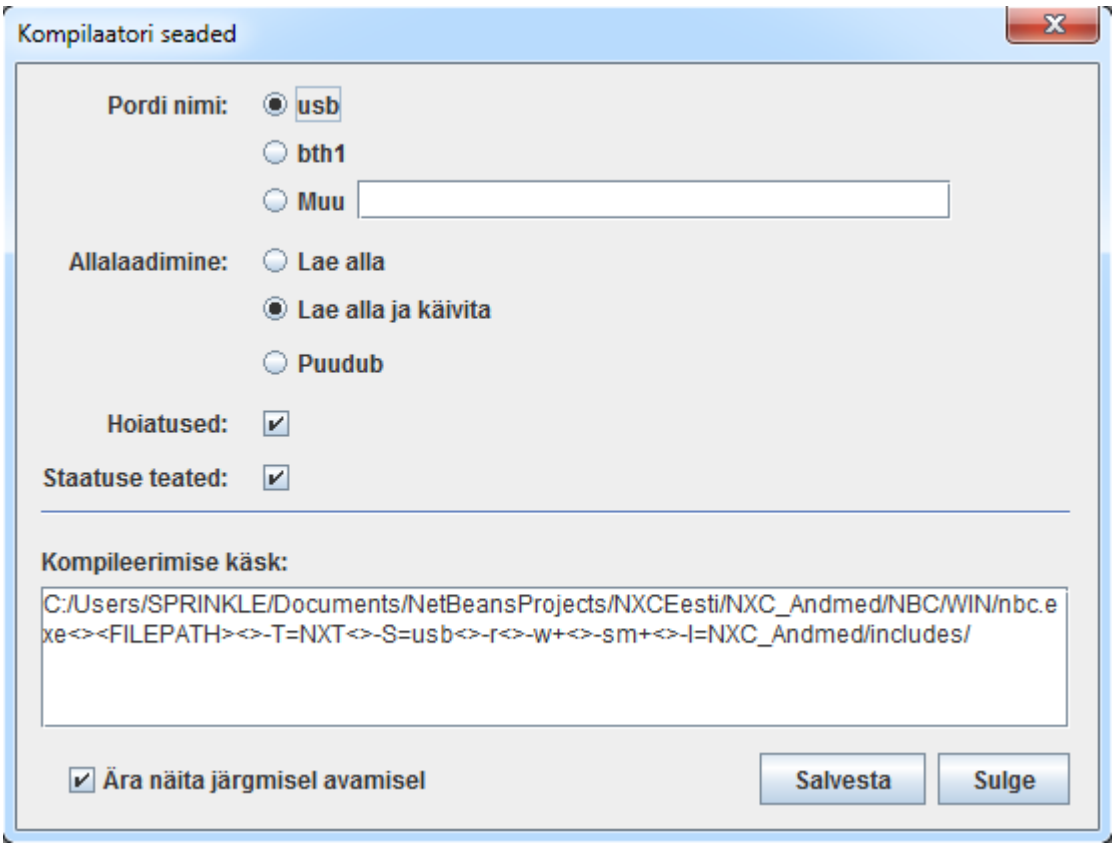
1. *Ava* - Valik olemasoleva *NXC* programmi avamiseks.
2. *Uus fail* - Loob uue *NXC* faili, millel on vaikimisi nimeks *programm.nxc*. Kui varem redigeeritud fail on salvestamata, siis küsitakse enne uue faili loomist kinnitust eelmise salvestamise kohta (joonis 14).



Joonis 14. Sulgemise kinnitus.

3. *Salvesta* - Salvestab hetkel muudetava faili. Kui failil on vaikimisi nimi, siis küsitakse sellele uus.
4. *Salvesta nimega* - Võimalus salvestada fail uue nimega.
5. *Sulge* - Sulgeb programmi. Kui muudetav tekst on salvestamata, siis küsitakse salvestamise ja sulgemise kohta kinnitust (joonis 14).
6. *Vali kõik* - Selekteerib kogu teksti.
7. *Lõika* - Lõikab selekteeritud teksti ja lisab selle operatsioonisüsteemi lõikepuhvrisse.
8. *Kopeeri* - Kopeerib selekteeritud teksti operatsioonisüsteemi lõikepuhvrisse.
9. *Kleebi* - Asetab operatsioonisüsteemi lõikepuhvris oleva teksti redaktoris kohta, kus asub sisestusmärk (*caret*).

10. *Tagasi* - Unustab viimati tehtud tegevuse. Kokku hoitakse meeles kuni 100 viimast tegevust.
11. *Edasi* - Taastab viimati unustatud tegevused. Seda ainult juhul kui peale unustamist ei ole ühtegi muudatust tehtud.
12. *Joonda vasakule* - Liigutab selekteeritud teksti tabulaatori võrra vasakule. Vasakule joondamine toimub olukorrani, kus ühegi valitud rea esimeseks tähemärgiks ei ole tabulaatorit ega tühikut.
13. *Joonda paremale* - Liigutab selekteeritud teksti tabulaatori võrra paremale.
14. *Kompilaator* - Avab kompilaatori seadete vaate (joonis 15).

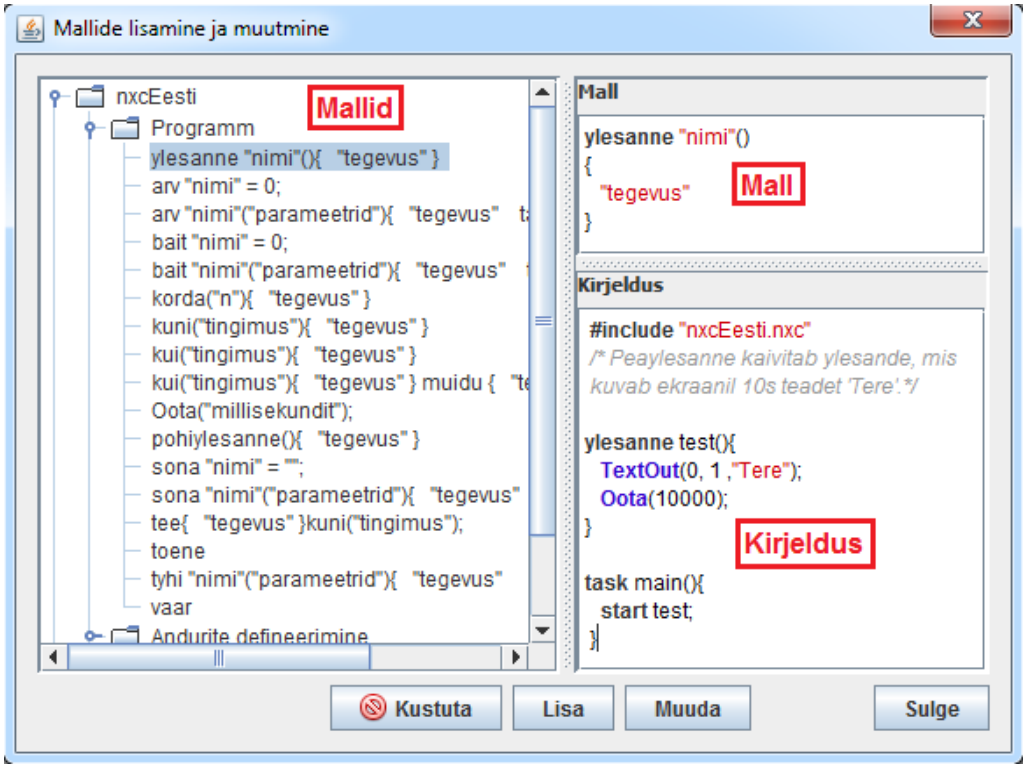


Joonis 15. Kompilaatori seaded.

15. *Tekst* - Avab vaate, kus saab redaktori teksti suurust ja erinevat tüüpi sõnade värvimist muuta (joonis 16).

Joonis 16. Teksti seaded.

16. *Mallid* - Avab vaate, kus saab malle muuta ja lisada (joonis 17).



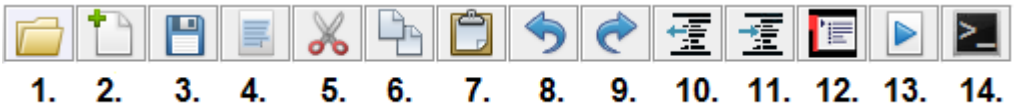
Joonis 17. Mallide lisamine ja muutmine.

17. *Redaktori failid* - Avab alammenüü, kust leiab programmi tööks vajalikud failid, mida saab kasutaja vajadusel ise muuta.
18. *Programmist* - Kuvab lühiinfo programmist.
19. *Kasutusjuhend* - Avab *pdf* faili, mis sisaldab NXCEesti kasutusjuhendit. Antud faili leiab ka programmi kaustast *NXC_Andmed* nimega *juhend.pdf*. Soovi korral võib kasutaja selle enda loodud juhendiga asendada.
20. *Taasta algseaded* - Taastab programmi algseaded. See sisaldab kõiki seadeid koos tekstifailide ja kompilaatoritega, mis on atud programmi tööks vajalikud. Programmi kaustast *NXC_Andmed* alamkaustast *VARU* saadakse algandmed, millega asendatakse programmi poolt kasutatavad failid.

Tööriistad

Programmi tööriistariba (joonis 18) sisaldab lisaks menüüdes olevatele valikutele ka võimalusi programmis kasutatavate paneelide näitamiseks ja redigeeritava koodi kompileerimiseks. Tööriistariba nupud on järgmised:

1. Olemasoleva faili avamine.
2. Loob uue faili.
3. Salvestab redigeeritava faili.
4. Selekteerib kogu aktiivse paneeli teksti.
5. Lõikab selekteeritud teksti.
6. Kopeerib valitud teksti.
7. Asetab sisestusmärgi kohale operatsioonisüsteemi lõikepuhvris oleva teksti.
8. Unustab viimati sisestatud teksti.
9. Taastab viimati unustatud teksti.
10. Liigutab valitud teksti tabulaatori võrra vasakule.
11. Liigutab valitud teksti tabulaatori võrra paremale.
12. Avab või sulgeb mallide vaate.
13. Kompileerib redigeeritava programmi.
14. Avab või sulgeb kompilaatori väljundi vaate.



Joonis18. Tööriistariba

Mallid

Mallid on NXCEesti osa, mis sisaldab programmi kirjutamiseks hulka erinevaid šabloone. Nendeks eeskujuvormideks on enamus *NXC* rakendusliidese funktsioonidest, põhilised programmi struktuurid ning antud töö käigus tehtud tõlked ja standardfunktsioonid.

Malli kasutamiseks tuleb see hiirega redaktori alale soovitud kohta lohistada. Kui mall sisaldab jutumärkides olevaid märksõnu, siis tuleb kasutajal asendada need sobiva muutuja või konstandiga.

Malli lisainfo

Kui mallide vaates valida mingi mall, millele on lisatud lisainfot, siis antud vaates kuvatakse seda.

Redaktor

Redaktor on ala, kus toimub reaalne uue koodi kirjutamine. Kirjutatavas tekstis kujutatakse märksõnad, rakendusliidese funktsioonid ja konstandid rasvases kirjas ning kasutaja poolt määratud värvitoonis.

Selleks, et kasutada antud töö käigus loodud tõlkeid ja standardfunktsioone peab iga programmi alguses olema järgnev rida: `#include "nxcEesti.nxc"`.

Kompilaatori väljund

Kui loodav programm kompileerida, siis selles vaates kuvatakse kompileerimise käigus tekkinud staatuse- ja veateated.

Kompilaatori seaded

Kompilaatori seadete vaates (joonis 15) saab valida, missugust ühendusviisi roboti ja arvuti vahel kasutatakse. *USB* ühenduse kasutamiseks tuleks valida *usb*. Sinihamba ühenduse loomisel tuleks valida *bth1* või valik *muu* koos sobiva nimega. Sinihamba ühendus nõuab sobiva riistvaraga arvutit ning arvuti ja roboti eelnevad seadistamist, mida antud töös ei käsitleta.

Punkti *Allalaadimine* juures saab valida, missugused toimingud tehakse peale kompileerimist. Nendeks on järgnevad võimalused:

a. Programm laetakse robotisse ja käivitatakse.
b. Programm laetakse robotisse. Käivitamiseks peab seda kasutaja robotis ise tegema.
c. Programm kompileeritakse aga robotisse seda ei saadeta. Antud juhul saab ainult testida, ega kompileerimisel ei teki veateateid.

Valikutega *Hoiatused* ja *Staatuse teated* saab määrata, missugust infot kompileerimisel väljastatakse. Need võivad olla vajalikud programmist vigade otsimisel või edukast kompileerimisest tagasiside saamiseks.

Kompileerimise käsk paneelil kuvatakse kompileerimiseks kasutatav käsk. Vajadusel saab kasutaja selle sisu muuta. Kui tehtud muudatused salvestada, siis programmi kompileerimisel kasutatakse just selle kasti sisu. Antud käsus asendatakse lõik <*FILEPATH*> redigeeritava faili asukohaga. Märgipaar <> tähistab tühikut, mis ei asu faili nimes.

Ära näita järgmisel avamisel valikuga määratakse, kas programmi käivitamisel näidatakse kompilaatori seadete muutmise akent või mitte. Selleks, et programm saaks töötava kompileerimise käsu luua on kindlasti vaja enne esimest kompileerimist seadeid muuta.

3.6 Programmi täiendamine

NXCEesti kasutab tekstis värvitavate märksõnade ja mallide hoiustamiseks tekstifaile. Järgnevalt kirjeldatakse, kuidas need failid üles leida ning, kuidas neid muuta.

Värvitavad sõnad

NXCEesti kasutaja saab muuta, milliseid sõnu koodis värvitakse. Menüüst *Seaded, Redaktori failid* ning *Märksõnad* valides avatakse fail, mis sisaldab rasvases kirjas värvitavaid märksõnu. Valikuga *Konstandid* avatakse konstandid ning valikuga *NXC API* saab muuta sõnu, mis värvitakse teksti seadetes (*Seaded, Tekst* ja valik *Funktsioonide värv*) määratud tooni. *NXC API* märksõnad värvitakse kuni esimese avava suluni. Failides tuleb erinevad sõnad üksteisest eraldada reavahetustega.

Standardfunktsioonid

Programmile saab kasutaja lisada ka uusi tõlkeid ja standardfunktsioone. Selleks tuleb täiendada faili, mille leiab menüüst *Seaded, Redaktori* failid ning *NXCEesti API*. Tegemist on tavalise *NXC* programmiga, millel puudub tegum `main`. Olemasolevad tõlked on loodud kasutades makrode defineerimist ning erinevat tüüpi funktsioone.

Mallid

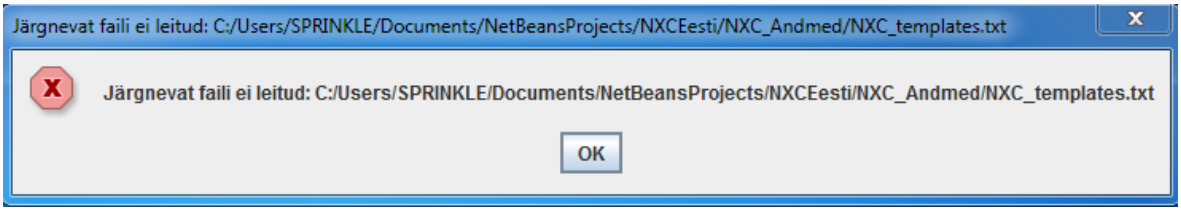
Mallide paneelil olevaid malle saab muuta menüü valikuga *Seaded* ning *Mallid* avanevas vaates (joonis 17). Mingi malli valimisel kuvatakse selle sisu paneelil *Mall* ning kirjeldus paneelil *Kirjeldus*. Elemendi muutmiseks tuleb teha tekstipaneelidel vajalikud muudatused ning vajutada nuppu *Muuda*. Nupuga *Lisa* lisatakse valitud elemendi järele uus sama astme element. Nupp *Kustuta* kustutab valitud elemendi ja selle alamelemendid.

Muudetavad failid leiab ka programmi alamkaustast *NXC_Andmed*. Linux operatsioonisüsteemi puhul tuleb arvestada, et programmi paigaldamisel kopeeritakse reaalselt kasutatav kaust kasutaja kodukataloogi.

3.7 Võimalikud veaolukorrad ja lahendused

NXCEesti kasutamisel võib ette tulla veaolukordi, mida saab kasutaja ise lahendada. Järgnevalt loetletakse tõenäolisemad vead koos lahendustega:

1. Programmi käivitamisel saab kasutaja sarnase teate joonisel 19. kujutatuga.
    a. Sellises olukorras peaks kasutaja veenduma, et kaust *NXC_Andmed* asuks veateates märgitud asukohas.



Joonis 19. Faili ei leitud.

2. Kompileerimisel saadakse teade, et allalaadimine ebaõnnestus (joonis 20)
    a. Tuleb kontrollida, kas robot on sisse lülitatud ja arvutiga ühendatud.
    b. Peaks veenduma, et kompilaatori seadetes on määratud õige ühendusviis. Mõningates seadetes võiks ka muudatusi teha, et uuendada kompileerimise käsku.
    c. Windows süsteemi korral tuleb kontrollida, et oleks paigaldatud Fantom Driver. Linux operatsioonisüsteemi puhul võiks käivita faili *LinuxInstall.sh* ning taaskäivita arvuti.

```
>#  Status: Write clump data to executable
>#  Status: Write code to executable
>#  Status: Write optimized source to compiler output
>#  Status: Finished
>#  Status: Download failed
null
```

Joonis 20. Allalaadimine ebaõnnestus.

3. Kui Mac OS operatsioonisüsteemiga arvutis saadakse peale kompileerimist joonisel 21 kujutatuga sarnanev kompilaatori väljund, siis tähendab see seda, et arvutisse ei ole paigaldatud Fantom Driver. Antud probleemi lahendamiseks tuleb LEGO MINDSTORMS koduleheküljelt [7] arvutisse salvestada ja installeerida Mac OS operatsioonisüsteemi Fantom Driver.

```
null
ERROR>dyld: Library not loaded: /Library/Frameworks/Fantom.framework/Versions/1/Fantom
ERROR>  Referenced from:
/Users/priitrand/Downloads/NXCEesti-2/dist/NXC_Andmed/NBC/MAC/nbc
ERROR>  Reason: image not found
```

Joonis 21. Mac OS operatsioonisüsteemil ei ole Fantom Driver installeeritud.

4. Kompileerimisel antakse kompilaatori väljundisse järgmine teade: “Palun määrake kompilaatori seaded”. Antud olukorras tuleks valida menüüst *Seaded*, *Kompilaator* ning muuta vajalikke seadeid.

Ülejäänud vead, mis tekivad kompileerimise käigus, on tõenäoliselt tingitud vigadest kirjutatavas programmis. Nende lahendamiseks tuleks juhinduda kompilaatori väljundi teadetest.

3.8 *NXC* eestikeelsete funktsioonide teek

Antud magistritöö raames loodi *NXC* kasutamise lihtsustamiseks keele märksõnade tõlkeid ja eestikeelseid standardfunktsioone. Loodud funktsioonid ja tõlked on kirjeldatud antud töö lisas Lisa 2. Joonisel 22 on toodud joonisel 4 kujutatud programm, kasutades loodud funktsioone ja tõlkeid.

Loodud funktsioonide kasutamiseks peab iga programmi päises olema järgnev rida: `#include "nxcEesti.nxc"`. Antud reaga kirjeldatakse kompilaatori preprotsessorile, et programmi kaasatakse ka fail *nxcEesti.nxc*, mille sisuks ongi eestikeelsed funktsioonid.

```
#include "nxcEesti.nxc" // Välise faili lisamine
#define KIIRUS 20       // Makro defineerimine
/* Programmi eesmärk on see, et robot liigub otse ning
takistuse saabumisel keerab kõrvale ja liigub edasi*/
bool majakas = toene;   // Globaalne loogilise väärtuse tüüpi muutuja
arv minkaugus = 50;     // Globaalne täisarv tüüpi muutuja.
                        // Maksimaalne vahemaa takistuseni, mis sunnib robotit keerama
ylesanne liigu(){       // Alamülesanne otse liikumiseks
  kuni(toene){          // Lõputu tsükkel
    kui(majakas){       // Tingimuslause, mis kontrollib, kas majakas on tõene
      OtseS(0, KIIRUS); // Robot liigub otse edasi
    } muidu {           // Tegevus, kui tingimus ei vastanud tõele
      PeataParem();     // Peatan mootori B, et robot keeraks
    }
  }
}

ylesanne m66da(){     // Alamülesanne
  arv takistus;       // Lokaalne muutuja mõõdetud takistuse kaugusele
  UltraheliSisse(4);  // Sensori initsialiseerimine sisendpordil 4
  kuni(toene){                          // Tsükkel
    takistus = Ultraheli(4);            // Loeb ultraheli sensorit
    PuhastaEkraan();                    // Tühjendab ekraani
    ArvEkraanile(1, 1, takistus);       // Takistuse kaugus ekraanile
    kui(takistus > 1 && takistus <= minkaugus){ // Mitu tingimust
      majakas = vaar;       // Majakas pannakse kinni, et ülesanne otse robotit ei liigutaks
    } muidu {               // Vastasel juhul
      majakas = toene;      // Majaka vabastamine
    }
  }
}

pohiylesanne(){     // Peaülesanne
  Alusta(m66da);    // Käivitan tegumi
  Alusta(liigu);    // Käivitan tegumi
}
```

Joonis 22. NXC näidisprogramm, kasutades antud töö raames loodud tõlkeid.

Käesoleva peatükiga kirjeldati *NXC* kasutamiseks loodud tarkvara NXCEesti.

Kokkuvõte

NXC keele kasutamiseks erinevatel operatsioonisüsteemidel puudus seni ühtne rakendus. Käesoleva töö peamiseks eesmärgiks oli luua *NXC*-keelsete rakenduste loomise keskkond, mida saaks ühtmoodi kasutada nii Windows, Linux kui ka Mac OS platvormil.

Enne uue keskkonna loomist uuriti *NXC* keelt üldiselt ning anti ülevaate juba loodud rakendustest *NXC* kasutamiseks. Lähtudes olemasolevatest programmidest ja LEGO robotite programmeerimist õpetava juhendaja soovitustest seati nõuded ka uuele süsteemile.

Antud töö raames loodi *NXC* kasutamiseks platvormiülene keskkond NXCEesti. Rakenduse põhilise funktsionaalsuse hulka kuulub integreeritud kompilaator, sõnade tüüpi eristav teksti värvimine, mallide kasutamine ning programmi töös kasutatavate andmete muutmise võimalus. Lisaks saab kasutada eestikeelseid *NXC* funktsioone. Arvestatakse ka kasutava operatsioonisüsteemi mõningate eripäradega.

Rakenduse edasiarenduse võimaluseks võiks näiteks olla roboti failihalduri loomine, mis lihtsustaks programmide robotist kustutamist. Samuti võiks lisada süntaksi kontrolli, mis märgiks ära avatud sulud või puuduvad funktsioonide parameetrid. Luua saaks ka lahenduse, kus programmis kasutatavaid andmeid saaks otse veebist või mõnelt andmekandjalt uuendada.

Püstitatud eesmärgid said täidetud ja loodetavasti leiab loodud rakendus ka reaalset kasutust *NXC* õpetamisel.

CROSS-PLATFORM ENVIROMENT FOR *NXC*

Master thesis (30 EAP)

Priit Rand

SUMMARY

The purpose of this Master thesis was to develop as cross-platform environment for programming LEGO MINDSTORMS NXT robots using *NXC* language. There are robotics classes in many Estonian schools. Pupils and teachers could have computers with different operating systems but there is no cross-platform programming environment for *NXC*.

This paper consists of three parts. The first part describes the *NXC* language and how to use it with no special development environment. The second parts gives overview from existing programming solutions for different operating systems. The third chapter describes created cross-platform environment for *NXC* programming.

The main points that describes the created application are:

- Works on Windows, Mac OS and Linux.
- Integrated compiler for *NXC*.
- *NXC* code highlighting.
- Possibility to create and translate standard *NXC API* functions.
- Possibility to use and modify *NXC* code templates.
- Designed for Estonian speakers.

It is hoped that this application find use in *NXC* programming.

Viited

[1] Anton Adamenkov (2010), bakalaureusetöö „Lego MINDSTORMS NXT'ga ühilduv õhurõhuandur ja pneumokomplekt“

[2] „NXT programmeerimiskeeled“

http://www.teamhassenplug.org/NXT/NXTSoftware.html

[3] „Bricx Command Center kodulehekülg“

http://bricxcc.sourceforge.net/, 26. aprill 2012

[4] Daniele Benedettelli „*NXC* õpetus“ 2007

http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf, 26. aprill 2012

[5] „*NXC* keele manuaal“

http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf, 26. aprill 2012

[6] „*NBC*, *NXC* ja *SPC* keel“

http://bricxcc.sourceforge.net/nbc/, 26. aprill 2012

[7] „Fantom Driver allalaadimine“

http://mindstorms.lego.com/en-us/support/files/Driver.aspx, 26. aprill 2012

[8] Jørgen Christian Larsen „*NXC* kasutamine linux operatsioonisüsteemil“ 2008

http://www.jclarsen.dk/index.php/guides/21-programming/52-using-nxc-on-linux, 26. aprill 2012

[9] „LEGO MINDSTORM kodulehekülg“

http://mindstorms.lego.com/en-us/Default.aspx, 26. aprill 2012

[10] „*NQC* keel“

http://bricxcc.sourceforge.net/nqc/, 26. aprill 2012

[11] John Hansen „*NBC* programmeerimise juhend“

http://bricxcc.sourceforge.net/nbc/doc/nbcapi/, 26. aprill 2012

[12] „Notepad++“

http://notepad-plus-plus.org/, 26. aprill 2012

[13] „*NXC* kasutamine programmiga Notepad++“

http://spillerrec.wordpress.com/2011/03/05/writing-nxc-code-in-notepad/, 26. aprill 2012

[14] „*NXC* programmi koodi värvimine programmis Notepad++“

http://muntoo.wordpress.com/2011/07/13/nxc-syntax-highlighting-in-notepad/, 26. aprill 2012

[15] „Programmi Gedit kasutamine suvalise programmeerimiskeele arendusvahendina”
http://techmalt.com/?p=254, 26. aprill 2012
[16] „Lego programmeerimise vahendid“
http://bricxcc.sourceforge.net/utilities.html, 26. aprill 2012
[17] „Net Beans“
http://netbeans.org/, 26. aprill 2012
[18] „Java 6 rakendusliidese spetsifikatsioon“
http://docs.oracle.com/javase/6/docs/api/, 26. aprill 2012
[19] „Java allalaadimine“
http://www.oracle.com/technetwork/java/javase/downloads/index.html, 26. aprill 2012

Lisad

Lisa 1. Loodud tarkvara

Antud tööga on kaasas CD, millel on järgnev sisu:

1. Dokumentatsioon - Sisaldab lähtekoodist genereeritud dokumentatsiooni (javadoc) ja klassidiagrammi.
2. Net Beans projekt - Kasutatud arendusvahendi projekt, mis sisaldab lähtekoodi.
3. Programm - Valminud rakendus, mille saab kasutaja oma arvutisse kopeerida ja käivitada. Eelnevalt võiks arvutis teha ka seadistused roboti arvutiga ühendamiseks (kirjeldatud alamkausta NXCEesti failis *LOE_MIND*).
4. juhend.pdf - NXCEesti kasutusjuhend.

Lisa 2. *NXC* eestikeelsete funktsioonide teek

Käesolevas lisas on toodud antud töö raames loodud tõlked *NXC* märksõnadele ning eestikeelsed funktsioonid.

Tõlked

NXC keele märksõnadest on tõlgitud järgnevad:

1. **korda** - *repeat*
2. **kuni** - *while*
3. **arv** - *int*
4. **kui** - *if*
5. **muidu** - *else*
6. **tee** - *do*
7. **ootaKuni** - *until*
8. **pohiylesanne** - *task main*
9. **ylesanne** - *task*
10. **toene** - *true*
11. **vaar** - *false*
12. **sona** - *string*
13. **bait** - *byte*

14. **tagasta** - *return*
15. **tyhi** - *void*

Muutujate loomisel tuleb arvestada, et nende nimedeks ei saa olla ka tõlked.

Funktsioonid

Järgnevalt kirjeldatakse antud töö käigus loodud funktsioone.
Kiiruse parameetrid saavad olla vahemikus -100 kuni 100. Kui kiirus on negatiivse märgiga, siis liigub mootor vastassuunas.
Sensorite ja mootorite portide ning ekraani rea väärtused ei saa olla muutujates. Väärtused tuleb otse funktsiooni kirjutada. Sensorite portide võimalikud väärtused on 1, 2, 3 ja 4. Mootorite portide võimalikud väärtused on A, B, C, AB, AC, BC ja ABC. Ekraani rida saab määrata numbritega 1 kuni 8.

Alusta("tegum");
Käivitab tegumi.

ArvEkraanile("x", "rida", takistus);
Kuvab ekraanile arvu. Parameeter *x* näitab pikslites arvu algust ekraani vasakust servast. Parameeter *rida* Näitab rea numbrit.

HeliSisse("port");
Defineerib etteantud pordil heli anduri.

Heli("port");
Tagastab etteantud pordilt heli anduri tulemuse.

HetkeTakt();
Tagastab süsteemi aja.

JoonEkraanile("x", "y", "x2","y2");
Joonistab roboti ekraanile joone.

Juhuarv("x");

Tahastab täisarv tüüpi juhuarvu vahemikus 0 kuni etteantud arv *x*.

KeeraParemaleP("kraadi", "kiirus");
Robot keerab paremale, pöörates mootoreid C ja B vastassuunas etteantud kraadide võrra ja kiirusega.

KeeraVasakuleP("kraadi", "kiirus");
Robot keerab vasakule, pöörates mootoreid C ja B vastassuunas etteantud kraadide võrra ja kiirusega.

KeskmineNuppVajutatud();
Kontrollib, kas roboti keskmist nuppu on vajutatud.

NelinurkEkraanile("x", "y", "laius", "kõrgus");
Joonistab roboti ekraanile nelinurga.

OtseS("pordid", "aeg", "kiirus");
Robot mootorid pöörlevad etteantud aja ja kiirusega.

OtseP("pordid", "kraadi", "kiirus");
Robot mootorid pööravad etteantud kraadide võrra etteantud kiirusega.

Oota("millisekundit");
Katkestab tegumi järgnevate käskude täitmise etteantud ajaks.

ParemNuppVajutatud();
Kontrollib, kas roboti parempoolset nuppu on vajutatud.

ParemS("aeg", "kiirus");
Keerab roboti parempoolset mootorit (B) etteantud aja ja kiirusega.

ParemP("kraadi", "kiirus");
Keerab roboti parempoolset mootorit etteantud kraadide võrra ja kiirusega.

PuudeSisse("port");

Defineerib etteantud pordil puute sensori.

Puude("port");

Tagastab puute sensori tulemuse.

PeataMootorid();

Peatab kõik mootorid.

PeataMootor("port");

Peatab mootori etteantud pordil.

PeataVasak();

Peatab mootori pordis C

PeataParem();

Peatab mootori pordis B

PuhastaEkraan();

Tühjendab roboti ekraani.

PuhastaRida("rida");

Tühjendab rea (1 kuni 8) roboti ekraanil.

PunktEkraanile(x, y);

Joonistab roboti ekraanil punkti.

RingEkraanile("x", "y", "raadius");

Joonistab roboti ekraanile ringi. *x* ja *y* tähistavad ringi keskpunkti.

SulgeNuppVajutatud();

Kontrollib, kas roboti sulgemise nuppu on vajutatud.

TekitaHeli("sagedus","kestvus");

Tekitab heli etteantud sagedusega ja kestvusega (ms)

TekstEkraanile("x","rida","tekst");

Kuvab ekraanile teksti. Parameeter x näitab pikslites teksti algust ekraani vasakust servast. Parameeter *rida* näitab rea numbrit(1 kuni 8).

UltraheliSisse("port");

Defineerib etteantud pordil ultraheli sensori.

Ultraheli("port");

Tagastab ultraheli sensori tulemuse (0 kuni 255 cm);

VaigistaHeli();

Kui programm tekitab heli, siis see vaigistatakse.

ValgusSisse("port");

Defineerib etteantud pordile valguse sensori.

Valgus("port");

Tagastab valguse sensori tulemuse (0 kuni 100).

ValgusSisseValgusega("port");

Defineerib antud pordile valguse sensori, millel on valgusallikas sisse lülitatud.

VasakNuppVajutatud();

Kontrollib, kas vasakpoolset nuppu on vajutatud.

VasakP("kraadi","kiirus");

Vasakpoolne mootor (mootor pordis C) keerab etteantud kraadide võrra ja kiirusega.

VasakS("aeg","kiirus");

Vasakpoolne mootor (mootor pordis C) keerleb etteantud aja ja kiirusega.