Open your mind. LUT.
Lappeenranta University of Technology

VAIHTOEHTOISIA TEKSTIPOHJAISIA OHJELMOINTIKIELIÄ LEGO MINDSTORMS® NXT –ALUSTALLE KORKEAMMAN ASTEEN OPETUSKÄYTÖSSÄ

Alternative text-based programming languages for Lego Mindstorms® NXT platform for tertiary level educational purposes

Iiro Pelli

Kandidaatintyö
LUT Energia
Sähkötekniikka

16.9.2016

TIIVISTELMÄ

Iiro Pelli
0371499
LUT School of Energy Systems
Sähkötekniikka
Tero Ahonen

Vaihtoehtoisia tekstipohjaisia ohjelmointikieliä Lego Mindstorms® NXT –alustalle korkeamman asteen opetuskäytössä

2016

Kandidaatintyö.
31 sivua, 1 kuva, 9 esimerkkikoodia, 3 taulukkoa

Työn tarkastaja: Tutkijatohtori Tero Ahonen

Hakusanat: sulautetut järjestelmät, robotti, ohjelmointi, opettaminen

Tässä kandidaatintyössä selvitetään eri vaihtoehtoisten tekstipohjaisten ohjelmointikielien saatavuus opetuskäytössä olevalle Lego Mindstorms NXT –alustalle. Soveltuvia ohjelmointikieliä havainnollistetaan esimerkkiohjelmien avulla sekä esitellään näiden käyttöönottomenetelmät. Myös kielien välisiä vertailuja tehtiin verraten niiden nopeutta, kokoa, ohjelmistoympäristöjen ominaisuuksia ja ohjelmointirajapintojen selkeyttä. Tiedon haku suoritettiin kirjallisuusselvityksenä.

Taustaselvityksessä havaittiin, että opetuskäyttöön soveltuvia ohjelmointikieliä on useita: leJOS nxtOSEK (C/C++), RobotC (C), Not eXactly C (C), leJOS NXJ (Java), Next Byte Codes (C /Assembly), pyNXC (Python/C) ja pbLua (Lua).

Työn keskeisimpien tuloksien mukaan RobotC on suunnattu hieman kokeneemmille C-ohjelmoijille tarjoten monipuolisen ohjelmistoympäristön sekä tehokkaan suorituskyvyn. NxtOSEK tukee reaaliaikakäyttöjärjestelmiä ja se on suunnattu edistyneille C/C++ -ohjelmoijille. Not eXactly C on sen sijaan suunnattu hieman vähemmän ohjelmointia harrastaneille, C-kielestä kiinnostuneille henkilöille. LeJOS NXJ edistyneille Java-ohjelmoijille tarjoten erittäin monipuolisen APIn. Assemblystä kiinnostuville opiskelijoille soveltuu Next Byte Codes ja vastaavasti Pythonille pyNXC ja Lualle pbLua.

ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
Electrical Engineering

Iiro Pelli

Alternative text-based programming languages for Lego Mindstorms® NXT –platform for tertiary level educational purposes

2016

Bachelor's Thesis.
31 pages, 1 figure, 9 code examples, 3 tables

Examiner: Post-Doctoral Researcher Tero Ahonen

Keywords: embedded systems, robot, programming, teaching

This bachelor's thesis is a study on text-based programming languages for Lego Mindstorms NXT. The applicable languages are demonstrated lightly with code examples and the procedures for installing required applications are given. Some comparison between languages are done by comparing written applications' speeds, sizes, given IDEs' features and APIs' plainnesses. The information was gathered via internet.

Several applicable programming languages were discovered: leJOS nxtOSEK (C/C++), RobotC (C), Not eXactly C (C), leJOS NXJ (Java), Next Byte Codes (C /Assembly), pyNXC (Python/C) and pbLua (Lua).

The main results of this thesis are that RobotC is oriented for a little advanced C programmers offering them a versatile IDE and good performance. LeJOS nxtOSEK has a real-time operating system support and it is targeted for advanced C/C++ programmers. Not eXactly C is for little less experienced programmers. LeJOS NXJ is for advanced Java programmers and supports a very complex API. Next Byte Codes is for Assembly rookies, pyNXC for Python and pbLua for Lua.

SISÄLLYSLUETTELO

KÄYTETYT MERKINNÄT JA LYHENTEET

ANSI	American National Standards Institute,
API	Application Programming Interface, ohjelmointirajapinta
IDE	Integrated Development Environment, ohjelmistoympäristö
NBC	NeXT Byte Codes
NXC	Not eXactly C
RAM	Random-Access Memory, keskusmuisti
ROM	Read-Only Memory, lukumuisti

1 JOHDANTO

Lego® Mindstorms® NXT on vuonna 2006 julkaistu RCX-sarjan korvaava ulkoisesti ohjelmoitava robottipaketti, joka sisältää mikropiirillä varustetun ohjelmoitavan NXT keskuspalikan, johon voidaan liittää neljä ulkoista sensoria sekä kolme ulkoista ohjattavaa servomoottoria. Keskuspalikkaan voidaan liittää Legon Technic-sarjan rakennuspalikoita, minkä avulla voidaan rakentaa eri tarkoitukseen soveltuvia älykkäitä robotteja. NXT:n opetustuskäyttöön tarkoitettua alustaa on käytetty opetustarkoituksessa sulautetuilla järjestelmillä tai robotiikkaan johdattelevilla kursseilla maailmanlaajuisesti, sekä muun muassa Helsingissä, Jyväskylässä sekä Lappeenrannassa [1][2][3].

Mindstorms-sarjalla on vahva jalansija maailmanlaajuisissa robottikilpailuissa, joissa tarkoituksena usein on kilpailla roboteilla erilaisissa ympäristöissä, kuten sumopainissa tai sudokun ratkaisemisessa. Nykyään Mindstorms-sarjan uudempi versio, vuonna 2013 julkaistu EV3, on syrjäyttämässä vanhaa NXT-sarjaa.

1.1 Tausta

Legon NXT-paketti ei vakiona tarjoa mukanaan muuta kuin Legon oman erittäin korkean tason graafisen LabVIEWiin pohjautuvan NXT-G ohjelmointikielen, joka on suunnattu aloittelijoille tarkoituksenaan saada rakennetut robotit suorittamaan haluttuja toimenpiteitä helposti ilman aiempaa ohjelmointikokemusta. Laite ei tue alkuperäisessä tilassaan ulkopuolisia kieliä, minkä takia kolmansien osapuolien on tarvittaessa täytynyt itse kustomoida laitteen omaa firmwarea tukemaan kielen käyttämää käskykantaa.

Tekstipohjaiset ohjelmointikielet, kuten C/C++/Java, ovat suuressa suosiossa teollisuuden parissa [5], ja täten myös eri koulut ja oppilaitokset tarjoavat erilaisia ohjelmointikursseja vastaaville kielille. Ohjelmointikielien eroavaisuudet koostuvat usein ohjelmien koosta, nopeudesta sekä opittavuudesta. LUT:n ”Johdanto sulautettuihin järjestelmiin” –kurssilla ohjelmointikielenä on tähän mennessä käytetty vain C-kieleen pohjautuvaa leJOS nxtOSEK-kieltä. Eri ohjelmointikielien kartoitus on tarpeen, koska eri taustaisia opiskelijoita on paljon sekä saatavilla olevien kielien määrä on moninainen. Työn aihe on valittu alunperin tarkoituksena selventää, mitä erilaisia tekstipohjaisia

kolmansien osapuolien ohjelmointikieliä NXT-alustalle on vuoden 2006 julkaisuajankohdan jälkeen saatavilla.

1.2 Työn tavoitteet ja rajaus

Työn tavoitteena on esittää erilaisia opetukseen soveltuvia ohjelmointikieliä NXT-alustalle. Työssä tarkastellaan tekstipohjaisia kieliä, joiden suorittaminen on mahdollista tehdä omalla NXT-alustan suorittimella. Työssä pyritään myös esittämään vertailua kielien kesken esimerkkikoodien, kielien opittavuuden, ohjelmien nopeuden sekä kokojen avulla.

2 SULAUTETTUJEN JÄRJESTELMIEN OHJELMOINTI

Kaikkia elektronisia laitteita, joihin on asennettu dataa käsittelevä mikroprosessori poislukien tietokoneet, kutsutaan sulautetuiksi järjestelmiksi. Sulautettuja järjestelmiä ovat esimerkiksi autot, mikroaaltouunit sekä uudemmat jääkaapit. Sulautetut järjestelmät kykenevät yleensä reagoimaan ulkopuolisiin herätteisiin niihin liitettyjen sensoreiden avulla reaaliaikaisesti [6].

Sulautettuja järjestelmiä ohjelmoidaan usein niille tehtyjen ohjelmointiympäristöjen (IDE) avulla. Ohjelmointiympäristöt sisältävät yleensä tekstieditorin sekä kääntäjän ohjelmointia helpottamiseksi. Esimerkkinä mainittakoon mikroprosessoreiden johtavan valmistajan Atmelin kehittynyt ohjelmistoympäristö, Atmel Studio, joka sisältää muun muassa tekstieditorin sekä kääntäjän, joka kykenee kääntämään tehdyn koodin erilaisille mikroprosessoreille. Kyseistä ohjelmistoa käytetään myös muun muassa LUT:n Embedded System Programming -kurssilla. IDE tarjoaa myös valmiita kirjastoja sekä kykenee tarjoamaan käyttäjälle valmiin koodirungon alustasta rippuen, mikä helpottaa oman ohjelmointiprojektin aloittamista. Ohjelmistoympäristön vahva hallinta nopeuttaa ohjelmointiprojektien edistymistä, sekä edistää debuggaamista erinäisten työkalujen avulla.

2.1 Lego MINDSTROMS

Mindstorms NXT –alusta tukee alkuperäisessä tilassaan Legon omaa NXT-G ohjelmointikieltä. Kieli on graafinen ja pohjautuu LabVIEWiin sekä se on suunnattu lapsille ja robotiikkaan kiinnostuneille ihmisille, jotka eivät ole aiemmin olleet tekemisisissä ohjelmoinnin kanssa. Graafinen ohjelmointi on rajoittunutta, sillä ohjelmoinnissa käytettyjä valmiita moduuleita ei voida muokata toiminnaltaan läheskään yhtä paljoa kuin mitä tekstipohjaisissa toteutuksissa.

2.2 LUT:n johdanto sulautettuihin järjestelmiin –kurssi

Joka vuosi LUTissa keväisin järjestettävällä ”Johdanto sulautettuihin järjestelmiin” –kurssilla käytetään oppimisalustana Mindstorms NXT 2.0:aa. Kurssin käytännön harjoituksissa opiskelijat opettelevat ohjelmoimaan sekä testaamaan sovellusohjelmia NXT:llä C-kielellä. Tähän mennessä kurssilla on käytetty C-kieleen pohjautuvaa avoimeen lähdekoodiin perustuvaa leJOS nxtOSEK –ohjelmointikieltä.

3 KOLMANSIEN OSAPUOLIEN TEKSTIPOHJAISIA OHJELMOINTIKIELIÄ

NXT:lle tuotuja tekstipohjaisia ohjelmointikieliä on kymmeniä. Kielet poikkeavat toisistaan nopeuden, ohjelmien koon, kielen opittavuuden ja tuettujen ohjelmistoympäristöjen mukaan. Tässä kappaleessa listataan eri kieliin pohjautuvia ohjelmointikieliä sekä tutustutaan niiden ominaisuuksiin asennustapojen ja koodiesimerkkien avulla. Asennuksessa keskitytään pääosin Windows-käyttöjärjestelmään, sillä se on yleisin käyttöjärjestelmä LUT:n tietokoneissa.

3.1 LeJOS nxtOSEK

LeJOS nxtOSEK on ilmainen, avoimeen lähdekoodiin perustuva alusta. Alusta tukee ANSI C- sekä C++- ohjelmointikieliä. NxtOSEK on suunnattu sulautettujen järjestelmien ohjelmoijille sekä soveltuu C/C++ kieliin tutustuneille koodaajille [7]. Alusta tukee reaaliaikakäyttöjärjestelmää, joka pohjautuu TOPPERS/ATK ja TOPPERS/JSP kerneliin µITRON4.0 –standardien mukaisesti [8]. Reaaliaikakäyttöjärjestelmällä saadaan myös mahdolliseksi multitaskaus, minkä avulla voidaan luoda monimutkaisia sovelluksia. Sovelluksia voidaan asentaa suoraan NXT-blokkiin joita suoritin pystyy ajamaan itsenäisesti.

NxtOSEKin käyttöönotto vaatii muutaman työkalun asentamisen. Ensimmäinen nxtOSEKin asennuksen vaihe Windowsille koostuu Cygwinin asennuksesta. Cygwin mahdollistaa monien Linux ohjelmien käytön Windows-ympäristössä. Cygwinin kotisivuilta saa ladattua asennusohjelman, joka sisältää suoraviivaiset asennusohjeet.

Toinen vaihe sisältää GCC 4.0.2 GNU ARM:n asennuksen, eli GNU kääntäjäpaketin ARM prosessoreille, ja se tukee NXT:ssä olevaa ARM7 prosessoria. NxtOSEKin kotisivut eivät enää tarjoa toimivaa linkkiä GCC:n asennukseen, mutta toimivan paketin voi löytää GNU:n sivustolta (gnu.org/software/gcc/releases.html).

Viimeisessä vaiheessa valitaan NXT:n firmwaren asennus. Asennuksessa valitaan joko John Hansen’s ”Enhanced NXT firmware”, ”nxtOSEK NXT BIOS” tai ”no firmware” asennus. Opetuskäyttöön on suositeltavaa asentaa Enhanced NXT firmware, sillä muistiin voi lähettää

useamman nxtOSEK sovelluksen, ja muut kielet, kuten NXT-G/NXC/NBC toimivat sillä suoraan ilman firmwaren vaihtoa. Enhanced NXT Firmwaren asennusta varten ladataan John Hansen's NeXTTool, jonka avulla voidaan lähettää .rxe- ja .rfw –tiedostoja NXT:hen. Lisäksi mikäli USB-ajureita ei ole entuudestaan asennettu, niin Legon NXT ajurit, nimeltään "Fantom Driver", täytyy asentaa yhteyttä varten. Firmwaren lähetys NXT:hen on kuuden vaiheen työ, ja ohjeet löytyvät nxtOSEKin kotisivuilta.

Enhanced NXT firmwarella kyetään ajamaan sovelluksia, joiden koko on korkeintaan 64 kB (ROM+RAM). NXT BIOSilla voidaan ajaa sovelluksia, joiden koot ovat korkeintaan 224 kB (ROM) ja 50 kB (RAM), mutta ainoastaan yksi sovellus voi olla tallennettuna NXT:hen. "No firmware" asentaa sovelluksen suoraan NXT:n RAMiin ja se voidaan suorittaa suoraan RAM:sta. Hyötypuolena flash-muistia ei käytetä. Haittapuolena sovellus häviää kokonaan laitteen sammuessa, sekä sovelluksen koko on korkeintaan 64 kB [7].

NxtOSEKilla ohjelmointia varten voidaan käyttää esimerkiksi Eclipse CDT –ohjelmistoympäristöä C/C++:aa varten. Esimerkkiohjelmia, joita voidaan käyttää mallipohjana, löytyy nxtOSEKin mukana. Esimerkiksi valmiiksi ohjelmoitu *Helloworld.c* on mukana paketissa. Sensoreiden sekä servomoottoreiden käyttöä varten kotisivuilta löytyy yhden klikkauksen alta lista erinäisistä ohjelmointirajapinnoista. Esimerkiksi etäisyysanturin käynnistys, lopetus sekä luku voidaan suorittaa komennoilla *ecrobot_init_sonar_sensor(portti),* *ecrobot_term_sonar_sensor(portti)* ja *ecrobot_get_sonar_sensor(portti).*

Multitaskaamista varten taskit täytyy erikseen määritellä projektiin kuuluvassa .oil-tiedostossa. Tiedostossa taskeille täytyy määritellä interruptien tärkeysjärjestykset sekä aika, jonka välein taskia kutsutaan erikseen. Itse .c-tiedostossa taskien määrittelyt tehdään *DeclareTask(task1name, task2name)*-komennolla.

Ohjelman ollessa valmis, Cygwinin nykyhakemisto vaihdetaan projektikansioon ja ohjelma käännetään syöttämällä komentoriville komento *make all*. Tämän jälkeen ohjelma voidaan lähettää NXT:hen kirjoittamalla Cygwiniin joko *./rxeflash.sh*, *./appflash.sh* tai *./ramboot.sh*, riippuen asennetusta firmwaresta. Ohjelman lähettäminen onnistuu myös Eclipse CDT:n sisäänrakennetun

funktion avulla. Yläpalkista valitaan *external tools*, johon voidaan syöttää tarvittavat argumentit rxeflashia, appflashia tai ramboottia varten. Tarkemmat ohjeet löytyvät kotisivustolta kohdasta *Eclipse & nxtOSEK*.

Seuraavaksi esitetyssä koodiesimerkissä (3.1.1) käydään läpi nxtOSEKilla tehty toteus, jossa pääasiassa käynnistetään etäisyysanturi ja tulostetaan sen antamaa lukuarvoa LCD-näytölle.

```
1   /* example.c */
2   #include "kernel.h"
3   #include "kernel_id.h"
4   #include "ecrobot_interface.h"
5   #include <stdio.h>
6
7   /* OSEK declarations */
8   DeclareTask(LCDUpdate, ReadSensors);
9
10  /* Definitions */
11  #define SONAR SENSOR NXT PORT S1
12  float sonar = 0;
13
14  /* Device initialization and termination hooks */
15  void ecrobot_device_initialize()
16  {
17      /* Initialize sonar sensor */
18      ecrobot_init_sonar_sensor(SONAR_SENSOR);
19  }
20
21  void ecrobot_device_terminate()
22  {
23      /* Terminate sonar sensor */
24      ecrobot_term_sonar_sensor(SONAR_SENSOR);
25  }
26
27  /* nxtOSEK hook to be invoked from an ISR in category 2 */
28  void user_1ms_isr_type2(void){ /* do nothing */ }
29
30  /* Read for example every 10 ms */
31  TASK(ReadSensors)
32  {
33      /* display sonar sensor data on LCD */
34      sonar = ecrobot get sonar sensor(SONAR SENSOR);
35  }
36
37  /* LCDUpdate is continuously executed, every 5 ms */
38  TASK(LCDUpdate)
39  {
40      /* display sonar sensor data on LCD */
41      display_goto_xy(0,0);
42      display_string("Distance:");
43      display_int(sonar,3);
44      display_update();
45      }
46      TerminateTask();
47  }
```

Esimerkkikoodi 3.1.1. LeJOS nxtOSEKilla toteutettu esimerkkikoodi.

Etäisyysanturin käyttö vaatii ohjelmointirajapinnan käyttöönottoa, joka saadaa määrittelemällä kirjastosta *ecrobot_interface.h*. Sovelluksen rinnakkaisajossa käytetyt taskit määritellään

DeclareTask()-funktiolla. Etäisyysanturin käyttöönotto vaatii ohjelman käynnistyessä ajettavan käynnistyskutsun ”*ecrobot_init_sonar_sensor(sensor_port)*, sekä ohjelman sammuessa lopetuskutsun ”*ecrobot_term_sonar_sensor(sensor_port)*. *ReadSensors()*-taskin sisällä luetaan etäisyysanturin lukema, joka tulostetaan LCD-näytölle toisessa taskissa.

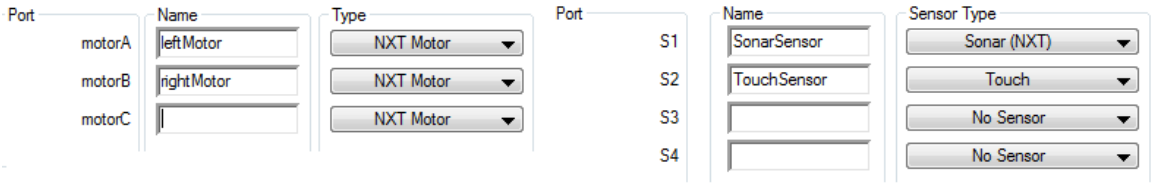
3.2 RobotC

RobotC on Windowsille tehty C-kieleen pohjautuva maksullinen ohjelmointikieli. Ohjelmistosta on saatavilla 10 päivän pituinen ilmaislisenssi, joka kattaa lähes saman toiminnallisuuden maksullisen version kanssa. 30 tietokoneen ”luokkahuonepaketti” on lisenssiltään noin $300. RobotC on suunnattu kehittyneempään opetuskäyttöön sekä ammattimaisempiin sovelluksiin. RobotC toimii ympäristönä Mindstorms NXT ja EV3 alustoille, VEX Roboticsin alustoille sekä eri mallisille Arduinoille. RobotC tarjoaa helppokäyttöisen ohjelmistoympäristön, joka sisältää reaaliaikaisen debuggerin sekä modernin tekstieditorin [9].

RobotC:n asennusohjelma voidaan ladata RobotC:n kotisivuilta. Asennusohjelma asentaa kaiken tarvittavan käyttäjän tietokoneelle eikä erillistä muokkausta jälkikäteen tarvita. Asennusohjelman jälkeen käyttäjä voi asentaa RobotC:n muokatun NXT tai EV3 –firmwaren laitteeseensa. Firmwaren asentaminen tapahtuu ohjelmistoympäristöstä löytyvän *Robot*-tabin alta. Valikosta voidaan valita joko NXT tai EV3, minkä jälkeen käyttäjä voi lähettää firmwaren suoraan *download firmware* vaihtoehdosta. Asennuksen tarkemmat ohjeet ovat löydettävissä RobotC:n kotisivuilta PDF-muodossa [9].

Ohjelmointi on verrattain helppoa tarjotussa ohjelmistoympäristössä. IDE kattaa suuren määrän valmiita funktioita, ja osa niistä on listattuna suoraan käyttäjälle tekstieditorin vasemmalla puolella, josta niitä voi raahata suoraan tekstikenttään. IDE:n mukaan on pakattu erittäin suuri määrä valmiita esimerkkiprojekteja, joita voi käyttää koodin pohjana tai muuten opiskelutarkoituksessa. Esimerkkiprojekteista löytyy muun muassa yksinkertainen koodi alustalla olevien nappien painalluksien lukemiseen sekä valmis, edistyneempi projekti tasapainottelevalle robotille.

Anturien ja servomoottorien käynnistystoimenpiteet on mahdollista tehdä IDE:n yläriviltä löytyvällä erillisellä *Motor and Sensor Setup* –painikkeella. Painiketta napsauttamalla avautuu kuvan 3.2.1 kaltainen ikkuna, johon voidaan syöttää moottorin/anturin nimi sekä tyyppi.



Kuva 3.2.1. RobotC-ohjelmistoympäristöstä löytyvä servomoottorien ja antureiden asetukseen käytettävissä oleva paneeli.

IDE luo automaattisesti paneeliin syötettyjen tietojen pohjalta tekstieditoriin komennot, jotka tekevät tarvittavat toimenpiteet oheislaitteiden käyttöönottoa varten. Valmiin ohjelman kääntäminen tapahtuu IDE:stä löytyvän *Compile Program* –painikkeen avulla, ja Mindstorms-alustalle lähetys onnistuu klikkaamalla *Download to Robot* –painiketta. Seuraavassa koodiesimerkissä (3.2.2) käydään läpi samanlainen toiminta kuin aikaisemmin.

```
1   #pragma config(Sensor, S1,     sonarsensor)
2   // automatically generated code
3
4   int distance = 0;
5   task ReadSensor()
6   {
7     while(true)
8     {
9       wait1Msec(10); // Allow for a short wait, freeing up the CPU for other tasks.
10      distance = SensorValue[sonarsensor];
11    }
12    return;
13  }
14  task main()
15  {
16    startTask(ReadSensor);    // Start Task ReadSensor.
17
18    while(true)
19    {
20          string s1 = "Distance: ";
21          nxtDisplayString(3, "%s", s1);  // display string s1
22          nxtDisplayString(3, "%d", distance);  // display the string, 's1' on line 3
23          wait1Msec(100); // Wait a small amount to allow the NXT Display to display correctly.
24    }
25    return;
26  }
```

Esimerkkikoodi 3.2.2. RobotC:llä toteutettu esimerkkikoodi.

Taskia *ReadSensor()* käytetään etäisyysanturin lukuun, ja se ajetaan 10 millisekunnin välein. Pääohjelma käynnistää taskin sekä tulostaa etäisyyden LCD-näytölle.

3.3 Not eXactly C

NXC, eli Not eXactly C, on John Hansenin kehittämä, nimensä mukaisesti C-kieleen pohjautuva ilmainen ohjelmointikieli NXT-alustalle. NXC:n ohjelmointiin suositellaan käyttämään pääosin Windowsille tehtyä ”BricxCC” ohjelmistoympäristöä, jonka voi ladata BricxCC:n omilta kotisivuilta. IDE ei ole yhtä kattava kuin Eclipse tai RobotC:n IDE, mutta yksinkertaisuudensa vuoksi se on helposti opittava.

BricxCC:n asennus onnistuu asennusohjelman kautta, joka asentaa kaiken tarvittavan käyttäjälle. NXC toimii vakiona Legon alkuperäisellä firmwarella, mutta mikäli käyttäjä haluaa multidimensionaalisten listojen sekä natiivien bit-shift –operaatioiden tuen, niin NXC:n kotisivuilta on löydettävissä paranneltu NBC/NXC firmware. Firmwaren päivitys on mahdollista suorittaa BricxCC:llä funktiolla, jonka voi löytää sen yläpalkista kohdasta *Download Firmware*.

Ohjelmointia helpottamiseksi NXC:stä löytyy noin 50 sivun pituinen opas sekä 2000 sivun mittainen yksityiskohtaisempi opaskirja, jotka ovat saatavilla IDEn *help*–kohdasta, sekä NXC:n omilta kotisivuilta. NXC ei syntaksinsa osalta poikkea C-kielestä, mutta toiminnallisuudeltaansa se ei kata kaikkea C:n tarjoamia ominaisuuksia. Kuten RobotC, BricxCC tarjoaa käyttäjälle myös valmiita funktioita suoraan tekstikenttään raahattaviksi. NXC:n tarjoamia ohjelmarajapintoja käytettäessä käyttäjän ei itse tarvitse lisätä tarvittavia kirjastoja mukaan koodiin, vaan kääntäjä lisää ne itse.

Esimerkiksi rinnakkaisajoa varten NXC tarjoaa valmiin *mutex*-muuttujan, jonka avulla voidaan esimerkiksi estää useamman taskin yhtäaikainen servomoottorien nopeuksien muutos. Eston voi myös toteuttaa semaforien käytöllä. Seuraava esimerkkikoodi (3.3.1) kattaa taskien, etäisyysanturien, moottorien sekä näytölle tulostamisen periaatteellisen toiminnan.

```
1   /* example_NXC.c */
2
3   task ReadSensors()
4   {
```

```
5         /* Initialize ultrasonic sensor */
6         SetSensorUltrasonic(IN_1);
7         /* Read distance from sensor */
8         dist = SensorUS(IN_1);
9     }
10    task Motors()
11    {
12         /* Go forward, 75% power */
13         OnFwd(OUT_AC, 75);
14              /* Shut down motors if distance is equal or less than 50 */
15         if (dist <= 50) {
16              Off(OUT_AC);
17              stop(Motors);
18        }
19        Wait(500);
20    }
21    task LCDUpdate()
22    {
23         ClearScreen();
24         /* Print distance on LCD */
25         NumOut(0, 0, dist);
26    }
27    task main() {
28         int dist = 0;
29         Precedes(ReadSensors, LCDUpdate, Motors);
30    }
```

Esimerkkikoodi 3.3.1. NXC:llä toteutettu esimerkkikoodi.

Ultraäänisensorin käynnistys voidaan toteuttaa pääohjelmassa helposti yhden rivin komennolla, argumentiksi vaaditaan vain portti, johon sensori on kytketty. Servomoottoreita voidaan ohjata *OnFwd(motors, pwr)*, *OnRev(motors, pwr)* ja *Off(motors, pwr)* –komennoilla. Taskia *main()* käytetään vain rinnakkaistaskien käynnistämiseen, ja niitä voidaan sammuttaa erikseen funktiolla *stop()*. Laskuoperaatiot ja vertailut tapahtuvat normaalisti C-kielen syntaksin mukaan.

3.4 leJOS NXJ

LeJOS on ilmainen, avoimen lähdekoodin Javaan pohjautuva ohjelmointikieli NXT- ja EV3 –alustoille. Se on suunnattu jo entuudestaan Javaan tutustuneille ohjelmoijille. LeJOS on alunperin RCX:lle tehdyn ”TinyVM” –firmwaren uudempi NXT:lle luotu versio, joka mahdollistaa laitteiston Java tuen. LeJOS NXJ sisältää kaikki NXJ:n tukemat API:t sekä työkalut koodin lähettämiseen NXT:lle [10].

LeJOS NXJ käyttää teollisuuden standardien mukaista Java-kieltä. NXJ:llä voidaan ohjelmoida Windows-, Linux- sekä Mac OS X –alustoilla. Ohjelmoinnin helpottamista varten

ohjelmistoympäristöt, kuten Eclipse sekä Netbeans, tukevat kieltä NXJ-pluginien asennusten jälkeen. NXJ:tä on käytetty maailmanlaajuisesti opetustarkoituksessa yliopistoissa sekä muissa oppilaitoksissa [10].

NXJ:n asennus vaatii ensimmäiseksi tietokoneelta tuen 32-bittiselle Java Development Kitille (JDK). USB-yhteyttä varten Legon Fantom usb-ajurit on syytä olla asennettuna. NXJ:n kotisivuilta lödettävissä oleva asennusohje sisältää tarvittavat linkit JDK:n sekä Fantom-ajureiden lataamiseen. LeJOS NXJ:n asennusohjelman kulku on suoraviivainen, ja sen päätteeksi NXT:n firmwaren päivitystä varten avautuu erillinen ”NXJ Flash” –työkalu. Asennus on myös mahdollista suorittaa manuaalisesti.

LeJOS NXJ –ohjelmia on mahdollista kirjoittaa, kääntää ja lähettää NXT:hen ilman erillistä IDE:ä. Valmiin Java-ohjelman jälkeen käyttäjä aukaisee komentorivin, menee haluttuun projektikansioon ja kirjoittaa komentoriville esimerkiksi *nxjc Helloworld.java*. Linkkerille sekä NXT:lle lähettämistä varten on omat komentonsa, mutta työtä on helpotettu yhdistämällä nämä yhdeksi *nxj –r <objekti>* –komennoksi. Ohjelmia voidaan myös ajaa tietokoneella PC API:a käyttämällä, jota varten on myös luotu tarvittavat komentorivin komennot.

LeJOS NXJ:n käyttäminen Eclipsellä vaatii erillisen pluginin asennuksen. Asennus on helppo tehdä sille tarkoitetulla funktiolla, joka vaatii vain pluginin internet-osoitteen. Asennuksen jälkeen Eclipsellä on mahdollista päivittää NXT:n firmware. Eclipsellä voidaan kääntää kirjoitettu sovellus sekä lähettää että käynnistää ohjelma NXT:ssä tietokoneen kautta. Oppimista helpottamiseksi esimerkkiprojekteja tulee asennuksen mukana kymmeniä, sekä lisää on tarjolla leJOSin kotisivuilla. LeJOS:n foorumeilla on myös aktiivinen käyttäjäyhteisö josta voi tarvittaessa löytää apua.

Koodiesimerkissä 3.4.1 on esitetty yksinkertainen leJOS NXJ esimerkkikoodi. Moottorit käynnistetään liikkumaan eteenpäin ja ultraäänisensorilla mitataan etäisyyttä eteenpäin. Etäisyyden ollessa 50 tai alle, moottorit pysäytetään.

```
1   import lejos.nxt.*;
2   public class DistanceTask {
3           public static void main (String[] args)
4           throws Exception
5           {
```

```
6                   UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);
7                   Motor.A.SetSpeed(360); Motor.A.forward();
8                   Motor.B.SetSpeed(360); Motor.B.forward();
9                   while (true) {
10                          if (sonic.getDistance() <= 50) {
11                                  Motor.A.stop();
12                                  Motor.B.stop();
13                          }
14                          Thread.sleep(2000);
15                  }
16          }
17  }
```

Esimerkkikoodi 3.4.1. LeJOS NXJ:llä toteutettu esimerkkikoodi.

Multitaskaamista, tai Javassa kutsuttuna multithreadaamista, ei pätevyyden puutteessa onnistuttu esimerkkikoodiin sisältämään. Multithreadaamisen nopeahko toteutus ei onnistunut ilman aiempaa Java-taustaa.

3.5 Next Byte Codes

Next Byte Codes (NBC) on John Hansensin kirjoittama kieli, joka käyttää pohjanaan matalan tason assembly-kieltä. NBC käyttää ohjelmarajapintanaan samaa API:a kuin NXC, minkä myötä robotin liikkeelle saanti ei vaadi assembly-käskyjen käyttöä.

NBC:tä voidaan ohjelmoida millä tahansa tekstieditorilla tai vaihtoehtoisesti BricxCC:tä apuna käyttäen. NBC:n asennus ei vaadi erillisten asennusohjelmien ajoa. Ohjelmat on syytä kirjoittaa samaan kansioon NBC-ohjelman kanssa. Ohjelman kääntö voidaan toteuttaa komentorivillä asettamalla nykypoluksi NBC:n kansion ja kirjoittamalla *nbc <objekti.nbc>*. Ohjelman lähetys onnistuu komennolla *nbc –S=usb –d <objekti.nbc>*.

NBC:n kotisivuilta on ladattavissa oppaita, joita assemblyyn tutustumattomien on helppo seurata. Opaskirjojen avulla kyettiin kirjoittamaan koodiesimerkin 3.5.1 kaltainen ohjelma.

```
1   dseg segment
2           dist byte
3           mindist byte
4   dseg ends
5
6   thread main
7           SetSensorUltrasonic(IN_1)
8           OnFWD(OUT_BC,75)
9           set dist, 0
10          set mindist, 50
11          precedes ReadSensors, LCDUpdate
12  endt
13  thread ReadSensors
14  readloop:
```

```
15          ReadSensor(IN_1,dist)
16          brcmp GTEQ, readloop, dist, mindist
17          Off(OUT_BC);
18          jmp readloop
19  endt
20
21  thread LCDUpdate:
22  lcdloop:
23          ClearScreen();
24          /* Print distance on LCD */
25          NumOut(0, 0, dist);
26          jmp lcdloop
27  end
```

Esimerkkikoodi 3.5.1. NBC:llä toteutettu esimerkkikoodi.

Ohjelman kulku koostuu ensin muuttujien määrittelystä segment-osiossa. Main-threadissa alustetaan etäisyysanturi sekä käytetään NXC API:n funktiota *OnFwd()* asettamaan B- ja C –moottorit päälle. Seuraavaksi muuttujiin alustetaan luvut 0 ja 50, minkä jälkeen taskit *ReadSensors* ja *LCDUpdate* käynnistetään ajamaan yhtäaikaisesti. Threadin *ReadSensor* silmukassa luetaan etäisyysanturia ja sijoitetaan anturin antama lukuarvo muuttujaan *dist*. Komento *brcmp* vertaa muuttujia *dist* ja *mindist*. GTEQ (*"greater or equal than"*) mahdollistaa vertailun muuttujien välillä. Jos *dist* on suurempi kuin *mindist*, palataan takaisin lukemaan etäisyysanturin lukuarvoa. Muussa tapauksessa moottorit kytketään pois päältä. Silmukka aloittaa suorituksensa alusta komenolla *jmp*. Tämän threadin taustalla toimii LCD-näytön päivitys.

3.6 pyNXC

PyNXC on kieli, jonka avulla voidaan ohjelmoida NXT-alustaa Pythonilla. PyNXC kääntää Pythonilla kirjoitetun koodin NXC-kielelle, minkä jälkeen se voidaan kääntää NXC-kääntäjillä NXT:lle ladattavaksi. PyNXC käyttää APIna samoja kirjastoja kuin NXC, minkä myötä samoja funktioita on sovellettavissa pythonilla.

PyNXC:n asennus vaatii PyNXC-paketin lataamisen, joka sisältää python ohjelman kyseisten tiedostojen aukaisemista varten sekä NBC-kääntäjän näiden kääntämiselle. PyNXC toimii NXC:n tavoin suoraan NXT:n mukana tulevalla Legon firmwarella, mutta lisäominaisuuksia varten sen voi päivittää paranneltuun NXC/NBC firmwaren. Näiden asennustoimenpiteet on esitetty aiemmin. PyNXC:n on todettu toimivan Pythonin versioilla 2.5 ja 2.6 tai wxPythonin versiolla 2.8.7.1 [19].

Python ohjelman lataaminen, kääntäminen ja NXT:lle lähetys on mahdollista suorittaa PyNXC:n omassa GUI:ssa interaktiivisesti tai vaihtoehtoisesti komentorivin kautta kirjoittamalla riville *./pynxc –download program_name.py*. PyNXC:tä on mahdollista käyttää Windows, Linux sekä MAC OS X –käyttöjärjestelmillä. PyNXC:n paketti sisältää mukanaan esimerkkikoodeja, joissa käydään läpi funktiot muun muassa sensorien, moottorien ja taskien käytöille [19].

Python-koodia voidaan kirjoittaa millä tahansa tekstieditorilla, mutta suositeltavaa on käyttää Pythonin omaa editoria. Seuraava esimerkkikoodi 3.6.1 toteuttaa saman toiminnallisuuden kuin aiemmissa esimerkeissä. Robotti sammuttaa moottorit etäisyyden ollessa alle 50.

```
1    def task_readsensors():
2        while True:
3            if SensorVal(4)>NEAR:
4                Wait(800)
5            else:
6                Off(OUT_AC)
7
8    def task_lcdupdate():
9        while True:
10           ClearScreen();
11           /* Print distance on LCD */
12           NumOut(0, 0, dist);
13
14   def main():
15       OnFwd(OUT_AC,50)
16       DefineSensors(None,None,None,EYES)
17       Precededs(task_readsensors, task_lcdupdate)
```

Esimerkkikoodi 3.6.1. NXC:llä toteutettu esimerkkikoodi.

Taskit määritellään erikseen Pythonin aliohjelmien komennolla *def*. Pääohjelmä *main()* käynnistää moottorit B ja C sekä alustaa ultraäänisensorin. Ultraäänisensori on kytkettynä laitteen porttiin S4, minkä takia *EYES* on neljäntenä argumenttina. *Precedes* käynnistää anturin lukemiseen sekä näytön päivitykseen tehdyt taskit. Vertailukomennot tapahtuvat normaalisti Pythonin syntaksin mukaisesti.

3.7 pbLua

pbLua on NXT:lle asennettava käyttöjärjestelmä, joka pystyy kääntämään sekä ajamaan Lualla kirjoitettua koodia suoraan itse NXT-alustalla. pbLuaa voidaan ohjelmoida millä tahansa tekstieditorilla ja ohjelmien lähetys tapahtuu Hyperterminalin tai minkä tahansa

komentorivisovelluksen kautta kopioimalla kirjoitettu koodi suoraan komentoriville. pbLuaa voidaan käyttää käyttöjärjestelmästä riippumatta [11].

pbLuan asennus vaatii firmwaren sekä USB-ajureiden päivittämisen. Firmwaren voi lähettää NXT:lle käyttämällä Legon omaa NXT IDE:ä, ja valitsemalla *Update Firmware* johon valitaan pbLuan verkkosivuilta ladattavissa oleva tiedosto. USB-ajurit löytyvät samasta paketista, joka sisältää firmwaren [11]. pbLua käyttää itsetehtyä ohjelmointirajapintaa. Tiedostoja on myös mahdollista tallentaa NXT:n FLASH-muistiin ja täten myös valita niitä muistista erikseen ajettavaksi. Rinnakkaisajo on mahdollista pbLualla, mutta esimerkkikoodeja tästä ei verkosta ole saatavilla.

4 VERTAILUA

Eri ohjelmointikielet ovat parempia eri käyttötarkoituksiin kuin toiset, minkä myötä yksimielisesti parasta universaalia valintaa ei voida määrittää. Henkilöstä riippuen paras ohjelmointikieli riippuu siitä, mitä haluaa oppia, mitkä tiedot omaa jo etukäteen sekä mitä haluaa tehdä. Vertailua helpottamiseksi tässä kappaleessa keskitytään eri ohjelmointikielillä toteutettujen sovellusten nopeuksiin, kokoihin, ohjelmistoympäristöjen ominaisuuksiin sekä ohjelmointikielien opittavuuksiin ohjelmointirajapintojen avulla tarkasteltuna.

4.1 Ohjelmien nopeus, koko ja EV3-tuki

Team Hassenplugin ylläpitäjä Steve Hassenplug on kerännyt useampien NXT:n ohjelmointikielien statistiikkoja samaan taulukkoon helposti vertailtavaksi. Ohjelmia testatakseen hän on lähettänyt jokaiselle ohjelmointikielen kehittäjälle pseudokoodin, jonka pohjalta jokainen on tehnyt oman haluttua toimintaa vastaavan testisovelluksen. Pseudokoodi testaa ohjelmien nopeutta kierroksina minuutissa, ohjelmien kokoa tavuissa sekä ohjelmien kirjoittamiseen kulunutta aikaa [13].

Taulukko 4.1.1. Eri ohjelmointikielillä toteutettujen suorituskykyä varten luotujen sovellusten nopeus [12].

Kieli	leJOS nxtOSEK	RobotC	NXC	NBC	leJOS NXJ	pyNXC	pbLUA
Nopeus [kierrosta/min]	2695	103k	4285	?	?	~4285	18k
Ohjelman koko [tavua]	18240 (firmware mukana)	561	1428	?	?	~1428	1750
Kirjoittamiseen kulunut aika [min]	30	30	30	?	?	?	?

Taulukosta (4.1.1) voidaan havaita, että RobotC kykenee suorittamaan ohjelmaa nopeammin kuin muut ohjelmointikielet. RobotC on myös ohjelman kooltaan noin kolme kertaa pienempi kuin vertaisensa. NXC:n ja pyNXC:n samankaltaisuudet luvuissa ovat selitettävissä sillä, että pyNXC kääntää koodin ensin NXC-muotoon, minkä jälkeen koodit ovat likimain samanlaisia toistensa kanssa. Kysymysmerkeillä varustetut solut johtuvat testejen puutteista. Kyseiset solut olisi mahdollista täyttää suuntaa antavilla arvoilla, mikäli ohjelmat tehtäisiin itse valmiiksi annetun pseudokoodin pohjalta.

Koska NXT julkaistiin noin 10 vuotta sitten eikä niitä enää ole saatavilla uutena, ja EV3:a käytetään jo monissa kouluissa opetuskäytössä, on syytä tarkastella myös ohjelmointikielien mahdollista tukea EV3-alustalle. Muun muassa LUTin DI-vaiheen *"Digital Control Design"* –kurssilla käytetään harjoitustöissä joinain vuosina EV3:a. Opiskelijat todennäköisesti tutustuvat NXT-alustaan ensimmäisenä, sillä sitä käyttävä kurssi on suunnattu kolmannen vuoden opiskelijoille. NXT:n ohjelmointikieli voi saada lisäarvoa mikäli se tukee jo valmiiksi EV3:a, sillä NXT:stä eteneminen EV3:lle on helpompaa mikäli ohjelmointikieltä on käytetty opetuksessa jo aikaisemmilla kursseilla. EV3 saattaa myös mahdollisesti korvata nykyiset käytössä olevat NXT-alustat ajan saatossa. Seuraavassa taulukossa (4.1.2) on listattuna ohjelmointikielien mahdollinen EV3-tuki.

Taulukko 4.1.2. Eri ohjelmointikielien mukana tulevat ohjelmistoympäristöt sekä mahdollinen EV3-tuki.

Kieli	leJOS nxtOSEK	RobotC	NXC	NBC	leJOS NXJ	pyNXC	pbLUA
EV3 tuki	-	+	+ (EVC)	+ (EVC)	+	-	-

RobotC sekä lejOS NXJ tukevat suoraan EV3-alustaa. NXC:n ja NBC:n kehittäjä John Hansen on aloittanut EVC-kielen kehityksen EV3:lle [14], joka todennäköisesti on samankaltainen kuin hänen NXT:lle kehittämänsä kielet. nxtOSEKin, pyNXC:n ja pbLUAn kotisivuilta ei löytynyt mitään EV3:een viittaavaa, joten voidaan tehdä oletus, ettei näille löydy tukea.

4.2 Ohjelmistoympäristöt

NXT:tä ohjelmoidessa voidaan käyttää joko kielelle ehdotettua ohjelmistoympäristöä työtä helpottamiseksi tai vaihtoehtoisesti käyttäjän omavalintaista sovellusta. Esimerkiksi nxtOSEKia on mahdollista ohjelmoida Eclipsen tai Code::Blocksin avulla erillisen pluginin asennuksen jälkeen.

Koska uudet ohjelmistoympäristöt ovat laajuudeltaan ja toiminnoiltaan erittäin kattavia, on syytä tarkastella pienempää osa-aluetta vertailua tehtäessä. Vertailua varten valittiin ensisijaisiksi tutkittaviksi kohteiksi IDEjen debugger-tuki sekä simulaatiomahdollisuudet. Taulukossa (4.2.1) on listattuna jokaisen kielen ohjelmistoympäristöt sekä niiden tutkittavat ominaisuudet.

Taulukko 4.2.1. Eri ohjelmointikielien ohjelmointia varten suositellut ohjelmistoympäristöt sekä niiden mahdolliset debugger- ja simulaatiotuet.

Kieli	leJOS nxtOSEK	RobotC	NXC/NBC	leJOS NXJ	pyNXC	pbLUA
Ehdotettu IDE	Eclipse CDT	RobotC IDE	BricxCC	Eclipse	Python IDLE	-
Debugger	?	Muuttujalista	Muuttujalista	GUI-tools	BricxCC	-
Simulaatio	-	Robot Virtual Worlds / LCD	LCD	-	BricxCC	-

nxtOSEKin mahdollinen Eclipse CDT:n reaaliaikainen debuggausmahdollisuus ei tullut selväksi eri lähteitä tarkastellessa. Eclipseen on kuitenkin mahdollista asentaa useita eri lisäosia, joten debuggausmahdollisuus voi olla tehtävissä.

RobotC:n debugger sisältää ohjelman sammuttamisen ja käynnistämisen tietokoneen kautta sekä ohjelmassa käytettyjen muuttujien tilojen tarkastelun ohjelman ajon aikana. Myös moottorien ja anturien arvojen lukeminen sekä tilojen suora muuttaminen on mahdollista debuggerin avulla. NXT:tä on myös mahdollista käyttää avattavissa olevan virtuaalisen NXT-laitteen avulla, mikä tulostaa käyttäjälle NXT:n etupaneelin käytettävine kytkimineen ja LCD-näytön kanssa. Simulaatio on mahdollista myös maksullisella Robot Virtual Worlds –ohjelmalla, joka on integroitavissa RobotC IDEen. Virtual Worlds mahdollistaa RobotC:llä käännetyn koodin ajamisen virtuaalimaailmassa itsetehtyjen esteiden ja kappaleiden ympäristössä [17].

BricxCC sisältää ohjelmassa käytettyjen muuttujien tilojen tarkastelun ajon aikana sekä virtuaalisen NXT-alustan näytön. Tarkasteltavien muuttujien määrää on mahdollista säätää haluamakseen, sillä ne täytyy raahata näytölle aukeavaan *”Watch list”* –ikkunaan, joka listaa halutut muuttujat. Näyttöä on hyödyllistä simuloida sellaisissa tapauksissa, kun näytön tarkastelu on vaikeaa tai mahdotonta esimerkiksi robotin rakenteen takia.

leJOS NXJ:n debuggaamista varten Eclipselle on mahdollista asentaa erillisiä GUI-työkaluja. *”nxjconsoleviewer”* sisältää ikkunan, joka tulostaa RConsole-ohjelman debug-ulostuloa. Nämä

ominaisuudet tulevat NXJ:n asennuksen mukana, mutta ne on syytä aktivoida erikseen Eclipsen työkalut-osiosta [18].

Python IDLE ei tarjoa itsessään debuggausmahdollisuuksia, mutta NXC:lle käännettyään ohjelmaa on mahdollista debugata sekä simuloida BricxCC:n avulla. PbLua ei myöskään tue minkäänlaista kehittyneempää ratkaisua, sillä IDEä ei ole, vaan ohjelmointi tehdään tekstitiedostoon joka kopioidaan suoraan komentoriville laitteelle lähetettäväksi.

4.3 Kielien opittavuus

Ohjelmointikielien opittavuuteen vaikuttaa muun muassa henkilön aikaisempi ohjelmointitausta, kielien syntaksin ymmärrettävyys sekä API dokumentaatio. Tässä työssä tarkasteltujen kielien syntaksit ovat pääpiirteittäin esitetty kappaleen 3 eri koodiesimerkeissä, joten keskitytään nyt saatavilla oleviin ohjelmointirajapintoihin sekä niiden dokumentointeihin.

4.3.1 Ohjelmointirajapinnat

Kielien ohjelmointirajapintojen vertailtavuutta helpottamiseksi seuraavaan esimerkkikoodiin (4.3.1.1) aseteltiin eri kielien vastineita määrätylle toiminnolle. Koodissa käytetään yksinkertaisia komentoja servomoottoreiden päälle kytkemiseen, sillä ne ovat usein käytetyin komponentti sekä toiminto NXT:tä ajettaessa.

```
1    // nxtOSEK
2    nxt_motor_set_speed(motor_port, speed_percent, brake);
3
4    // RobotC
5    motor[motorA] = speed_percent;
6
7    // NXC, NBC, pyNXC
8    OnFwd(OUT_A, speed_percent);
9
10   // NXJ
11   Motor.A.SetSpeed(360); Motor.A.forward();
12
13   // pbLua
14   nxt.OutputSetSpeed(port, 32, speed)
15
```

Esimerkkikoodi 4.3.1.1. Eri ohjelmointikielten ohjelmointirajapinnan komentoja NXT:n mottorin käyttöä varten.

Yllä olevasta koodiesimerkistä voidaan havaita, että moottoria varten käytetyt komennot eivät poikkea toisistaan suuresti. Kielien APIen dokumentaatio on helposti löydettävissä näiden kotisivujensa kautta, ja näiden funktioiden yksinkertaisen nimeämisen myötä niiden käyttö on myös suhteellisen helppoa. Yleisesti funktiot ottavat argumenteikseen moottorin portin sekä nopeuden prosentteina nollasta sataan, kuten RobotC:n ja NXC:n tapauksissa. Monimutkaisin toteutus lienee NXJ:ssä, jota käytettäessä moottorin nopeus täytyy ensin alustaa erillisellä funktiolla asteina sekunnissa, minkä jälkeen moottori asetetaan pyörimään joko eteen- tai taaksepäin rivin 4 mukaisesti. Seuraavassa esimerkkikoodissa (4.3.1.2) vertaillaan vielä anturien käyttöä ohjelmointikielien kesken.

```
1    // nxtOSEK
2    ecrobot_get_sonar_sensor(port_id);
3    ecrobot_get_light_sensor(port_id);
4    ecrobot_get_sound_sensor(port_id);
5
6    // RobotC
7    SensorValue(touchSensor);
8    SensorValue(lightSensor);
9    SensorValue(soundSensor);
10
11   // NXC, NBC, pyNXC
12   Sensor(input_port);
13
14   // NXJ
15   LightSensor light = new LightSensor(SensorPort);
16   X = light.getLightValue();
17   SoundSensor sound = new SoundSensor(Sensorport);
18   Y = sound.readValue();
19   UltrasonicSensor sonic = new UltrasonicSensor(Sensorport);
20   Z = sonic.getDistance();
21
22   // pbLua
23   nxt.InputSetType(port,int)
24   light = nxt.InputGetStatus(light_port)
25   distance = nxt.InputGetStatus(us_port)
```

Esimerkkikoodi 4.3.1.2. Eri ohjelmointikielten ohjelmointirajapinnan komentoja NXT:n anturien käyttöä varten.

Funktioiden nimeämiset ovat edelleen hyvin selkeitä nxtOSEKilla, RobotC:llä sekä NXC:llä eivätkä komennot tarvitse syötteikseen yhtä argumenttia enempää. Nämä funktiot palauttavat suoraan halutun anturin antaman arvon, joka voidaan tallentaa muistiin esimerkiksi NXC:llä komennolla *int sound = Sensor(sound_sensor)*. Sen sijaan NXJ käyttää paljon luokkia, minkä myötä myös anturien käyttö on luokkien soveltamista. Tästä esimerkkinä rivillä 11 luodaan uusi *LightSensor*-luokka tietylle portille, minkä jälkeen kyseiseltä luokalta voidaan kutsua toimintoja esimerkiksi rivin 12 mukaan *x* =

light.getLightValue(). PbLuan API antureille on myös porttien alustamisen jälkeen helppoa, sillä anturien luku tapahtuu samalla komennolla *nxt.InputGetStatus(port)*.

Jokaisen kielen ohjelmointirajapinnasta on saatavilla oma dokumentaationsa mikäli on tarvetta syvällisempään tarkasteluun. Dokumentaatioihin pääsee helposti käsiksi kotisivujen kautta tai vaihtoehtoisesti syöttämällä online-hakukoneeseen esimerkiksi ”*NXC API*”.

4.3.2 nxtOSEKin TOPPERS ATK1 API

nxtOSEKin API koostuu TOPPERS ATK1 ohjelmointirajapinnasta, joka on samankaltainen OSEK OS/OIL:n kanssa, sekä ECRobot ohjelmointirajapinnasta, joka sisältää matalan tason laitteiston APIn sekä ECRobot ”wrapper” APIn. nxtOSEK rajoittaa TOPPERS ATK1 ominaisuuksia systeemiarkkitehtuurin takia, minkä takia käyttäjän ei suositella käyttävän ISR-määrityksiä sekä Interrupt handling –APIa [16].

TOPPERS ATK1 APIn dokumentaatio on saatavilla nxtOSEKin verkkosivuilta kohdasta ”API Reference”. nxtOSEKin käyttöönottoa varten tehtävää .OIL-tiedoston käyttöönottoa varten noin 40-sivuinen opas on ladattavissa PDF-muodossa, ja se kattaa kaikki ohjelman alustukseen liittyvät komennot. Tiedostoon määritellään yleensä taskien nimet, prioriteetit, alarmit sekä näiden jaksonnopeus. Koodiesimerkissä (4.3.2.1) on esitetty taskien sekä niiden herätteiden alustus.

```
 1   TASK TaskA {
 2           PRIORITY = 2;
 3           SCHEDULE = NON;
 4           ACTIVATION = 1;
 5           AUTOSTART = TRUE {
 6                   APPMODE = AppMode1;
 7                   APPMODE = AppMode2;
 8           };
 9           RESOURCE = resource1;
10           RESOURCE = resource2;
11           EVENT = event1;
12           EVENT = event2;
13           MESSAGE = anyMesssage1;
14    };
15
16    ALARM WakeTaskA {
17           COUNTER = SysCounter;
18           ACTION = ACTIVATETASK {
19                   TASK = TaskA;
20           };
21           AUTOSTART = TRUE {
22                   ALARMTIME = 50;
```

```
23                  CYCLETIME = 100;
24                  APPMODE = AppMode1;
25                  APPMODE = AppMode2;
26          };
27  };
```

Esimerkkikoodi 4.3.2.1. Taskien sekä niiden käynnistykseen käytetyt komennot .OIL-tiedostossa [15].

LUTin kurssilla harjoituksissa ei painoteta tämän API-kielen syvällisempään hallitsemiseen, sillä laitteen funktionaalinen toteutus on enemmän mielenkiinnon kohteena. Nämä tehtävät vaativat usein ainoastaan alarmien ja taskien määrittelyn lisäyksen tähän tiedostoon. Interruptit ja alarmit tulevat enemmän esille opiskeltaessa reaaliaikakäyttöjärjestelmiä, ja näiden ajankohta on monesti ajoitettu myöhemmälle kuin aiemmin mainitulla johdantokurssilla.

5 YHTEENVETO JA POHDINTA

Tässä työssä selvitettiin, mitä erilaisia tekstipohjaisia kolmansien osapuolien ohjelmointikieliä Legon julkaisemalle Mindstorms NXT –alustalle on ajan saatossa kehitetty. Ohjelmien täytyi myös olla ajettavissa itsenäisesti laitteen omalla prosessorilla. Ohjelmointikieliä vertailtiin keskenään suorituskyvyn, opittavuuden sekä niille suositeltavien ohjelmistoympäristöjen ominaisuuksien avulla. Vertailuissa ei pyritty selvittämään yksinkertaisesti parasta kieltä, sillä eri kielet soveltuvat tietynlaisiin sovelluksiin paremmin kuin toiset (esim. sulautetut järjestelmät, verkko-ohjelmointi), ja opiskelijoiden taustatiedoista riippuen esimerkiksi C-kieleen pohjautuvaa edistyneempää kieltä ei välttämättä ole järkevä opettaa Java-opiskelijalle. Tietoa etsittiin eri verkkolähteistä, sillä kirjajulkaisut sisälsivät ainoastaan tietoa graafisista kielistä.

Työn analysoitaviksi kieliksi kirjattiin seitsemän eri kieltä: kolme C-kieleen pohjautuvaa (RobotC, leJOS nxtOSEK, NXC), sekä neljä muihin kieliin pohjautuvia: leJOS NXJ (java), pyNXC (python), NBC (assembly) ja pbLua (lua).

Työssä tehtyjen vertailujen perusteella havaittiin, että RobotC on nopein sekä vähiten muistia käyttävä ohjelmointikieli. Myös RobotC:n ohjelmistoympäristön todettiin kattavan tutkittavat kriteerit ja lisäpisteitä tulee myös EV3-tuesta. Haittapuolena kyseisellä kielellä on sen maksullisuus, sillä muut kielet ovat ilmaisia. Täten RobotC olisi yksi erittäin hyvä kandidaatti opetuskäyttöön, mikäli maksulliset lisenssit eivät ole ongelma. leJOS nxtOSEK toisaalta tarjoaa hieman kokeneemmalle C/C++ -käyttäjille reaaliaikakäyttöjärjestelmät täyttävät kriteerit, jonka tarjoamat pohjatiedot (mm. taskien prioriteetit sekä ajastukset) on mielestäni hyväksi myös LUTin reaaliaikakäyttöjärjestelmät-kurssilla. NXC sen sijaan on hieman kevyemmin C-ohjelmointia lähestyvä toteutus, joka soveltuu täten C-aloittelijoille. leJOS NXJ on suunnattu entuudestaan kokeneille Java-ohjelmoijille, ja sillä toteutettavat ohjelmat voivat olla erittäin monimutkaisia sen laajan APIn ansiosta, ja EV3-tuki on myös saatavilla. Kyseinen kieli soveltuisi hyvin opetuskäyttöön ohjelmissa, joissa painotetaan enemmän Javan oppimista. NBC on suunnattu assemblyyn tutustuville ohjelmoijille NXC:n APIlla helpotettuna, ja se voisi olla hyvä kieli kokemattomille assemblystä kiinnostuneille ohjelmoijille. pyNXC toimii samalla tavalla kuin NBC, eli pythonin syntaksiin on laitettu sekaan NXC:n API. pbLua on yksinkertaisesti omalla NXT-APIllaan varustettu, komentoriviltä toimiva kieli.

Mahdolliselle lisätutkimukselle on vielä varaa, sillä Legon hiljattain julkaisema uusi EV3-alusta on syrjäyttämässä NXT-alustoita eikä uusia NXT-malleja ole enää myytävissä. Uuden alustan ohjelmointikielille tulisi jossain vaiheessa tehdä samankaltainen kartoitus kuin tässä työssä. Tämä työ ei myöskään kattanut tekstipohjaisia ohjelmointikieliä, joita on mahdollista käyttää etänä tietokoneen kautta ohjauslaitteena.

LÄHTEET

[1] Helsingin yliopisto, tietojenkäsittelytieteen laitos. ”Robottiohjelmointi”, [PDF-esitys]. 2013. Saatavissa http://www.cs.helsinki.fi/u/strommer/robo13/kalvot/robo13-luento1.pdf [viitattu 10.1.2016]

[2] Seifert, U. & Schmidt L. Introduction to robot programming with Lego Mindstorms NXT and Khepera-III for cognitive science of music, cognitive musicology, musical robotics, and artistic human-robot interaction. Saatavissa https://www.jyu.fi/hum/laitokset/musiikki/en/summerschool/2009/teachers/workshops/SeifertSchmidt [viitattu 10.1.2016]

[3] Ahonen, T. Johdanto sulautettuihin järjestelmiin –kurssin harjoitusten kuvaus. 2016. Saatavissa https://noppa.lut.fi/noppa/opintojakso/bl40a1811/harjoitukset [Viitattu 10.1.2016]

[4] Lego NXT Education User Guide. Saatavissa http://www.generationrobots.com/media/Lego-Mindstorms-NXT-Education-Kit.pdf [viitattu 10.1.2016]

[5] Cass, S. The 2015 Top Ten Programming Languages. 2015. http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages [Viitattu 13.1.2016]

[6] Ahonen, T. Johdanto sulautettuihin järjestelmiin. Luento 1. [PDF-esitys]. 2016. Saatavissa https://noppa.lut.fi/noppa/opintojakso/bl40a1811/luennot/lecture_1.pdf [viitattu 13.1.2016]

[7] Chikamasa, T. What is nxtOSEK? 2009. Saatavissa http://lejos-osek.sourceforge.net/whatislejososek.htm [viitattu 15.1.2016]

[8] TOPPERS Project. TOPPERS/JSP kernel. 2015. Saatavissa http://www.toppers.jp/en/jsp-kernel-e.html [viitattu 15.1.2016]

[9] Robomatter, Inc. RobotC, a C Programming Language for Robotics. 2014. Saatavissa http://www.robotc.net/ [viitattu 16.1.2016]

[10] LEJOS. Introduction. What is leJOS NXJ. Saatavissa http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/Intro.htm [Viitattu 17.1.2016]

[11] Hempeldesigngroup. What is pbLua? Saatavissa http://hempeldesigngroup.com/lego/pblua/about/ [viitattu 19.1.2016]

[12] Hassenplug, S. NXT Programming Software. 2008. Saatavissa http://www.teamhassenplug.org/NXT/NXTSoftware.html [viitattu 22.1.2016]

[13] Hassenplug, S. Software Speed Test. 2008. Saatavissa http://www.teamhassenplug.org/NXT/NXTSoftwareSpeedTest.html [viitattu 22.1.2016]

[14] Brickwiki.info. EVC. 2014. Saatavissa http://www.brickwiki.info/wiki/EVC [viitattu 31.1.2016]

[15] TOPPERS Project. TOPPERS ATK1 API Documentation. Saatavissa http://portal.osek-vdx.org/files/pdf/specs/oil25.pdf [viitattu 1.2.2016]

[16] Chikamasa, T. ECRobot API. 2009. Saatavissa http://lejos-osek.sourceforge.net/ecrobot_c_api_frame.htm [viitattu 1.2.2016]

[17] Robomatter, Inc. Robot Virtual Worlds. 2016. Saatavissa http://robotvirtualworlds.com/ [viitattu 19.4.2016]

[18] LEJOS. Using Eclipse. Saatavissa http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm [viitattu 19.4.2016]

[19] Code Google Archive. Pynxc - Getting Started. 2016. Saatavissa https://code.google.com/archive/p/pynxc/wikis/GettingStarted.wiki [Viitattu 19.1.2016]