Ülevaade LEGO Mindstorms NXT ja Java ühildumisest

Java on SUN Microsystems'i poolt arendatav objektorienteeritud programmeerimiskeel. Erinevalt teistest levinud programmeerimiskeeltest ei kompileerita Java programme otse masinkeelde, vaid vahepealsesse baitkoodi (*ingl. k. bytecode*). Java programmi käivitamisel käivitab arvuti eelnevalt Java virtuaalmasina (JVM), mis omakorda kompileerib baitkoodi riistvaraspetsiifilisse masinakeelde ja käivitab selle. See võimaldab sama Java programmi käivitada erinevatel platvormidel (erinevate riistvaralahenduste ja operatsioonisüsteemidega arvutitel). Seoses sõltumatusele platvormist on Java levinud peale lauaarvutite ka väike-süsteemidesse (*ingl. k. embedded devices*) nagu mobiiltelefonid, pihuarvutid, IP-telefonid jne.

NXT programmeerimiseks keeles Java on välja töötatud rakendus leJOS NXJ. leJOS NXJ on vabavara ning on loodud avatud lähtekoodiga leJOS projekti raames. leJOS püsivara sisaldab endas virtuaalmasinat Java baitkoodi jaoks ning lisatarkvara Java programmide laadimiseks ning jooksutamiseks. LeJOS NXJ sisaldab lisaks endas kõiki NXJ API (*Application Programming Interface*) klasse, mis on välja töötatud programmeerimaks NXT juhtklotsi.

LeJOS NXJ kasutamisel NXT-G (LEGO standard) või teiste keelte asemel on palju eeliseid: kasutatakse standardset Java keelt objektorienteeritud programmeerimise võimalustega, kaasneb kiirus ja täpsus, kasutada saab erinevaid programmeerimiskeskkondi - näiteks Eclipse ja Netbeans.

Kuna leJOS NXJ on püsivara, siis tuleb see NXT juhtklotsi välkmälule paigaldada, asendades LEGO MINDSTORMS püsivara. LEGO püsivara taastamiseks on vajalik NXT-G tarkvara. See võib osutuda probleemiks, kui tegemist on NXT Education komplektiga, kus NXT-G tarkvara kaasas ei ole.

Installeerimine (Win XP)

Kirjeldatud on installeerimise järjekorras vajalikud komponendid leJOS NXJ'i kasutamiseks MS Windows XP operatsioonisüsteemis.

Java

Vajalik on nii Java JRE (Java Runtime Environment) kui ka Java JDK (Develepment Kit), mis on vajalik Java programmide kompileerimiseks. Versioonidest on vajalik vähemalt JRE 5 ja JDK 1.5.

LEGO USB draiver

LEGO USB draiver (*ingl. k. driver*) on tarvilik selleks, et luua NXT-ga suhtlus üle USB. On võimalik ka ühendus läbi sinihamba, kuid läbi USB on ühendus kiirem ning kindlam. LEGO NXT USB draiveri saab LEGO Mindstorms veebilehelt. Kui eelnevalt on installeeritud LEGO NXT-G tarkvara, siis on küll USB draiver olemas, kuid tuleks siiski kontrollida, kas tegemist on viimase versiooniga. Peale installeerimist ühendada NXT juhtplokk arvutiga kasutades USB kaablit ning kontrollida, et LEGO NXT seade eksisteerib seadmete halduris (*My Computer>Properties>Device manager*).

LeJOS NXJ

Arvutis peaks olema alla laetud LeJOS NXJ viimane versioon (kasutatud versioon 0.7.0 beta), seejärel tuleb luua kohalikule kettale uus kaust, mille nimi ei sisaldaks tühikuid (näiteks C:\NXJ), tühik failinimes võib mõnikord Java tehnoloogiat kasutades probleeme tekitada. Installeerida leJOS NXJ viimati loodud kausta alamkausta *lejos_nxj*. Installeerimise lõppfaasis avatakse programm *NXJ firmware flash utility,* mis võimaldab meil NXT juhtklotsile paigaldada leJOS püsivara. Paigaldada leJOS püsivara NXT klotsile järgides programmi juhiseid.

Keskkonnamuutujad

Ava Keskkonnamuutujate vahetamise liides (*My computer>Properties>Advanced> Environment Variables*) ning kontrolli, kas *NXJ_HOME* on määratud süsteemi- või kasutaja-muutuja ning *PATH* väärtuse (*ing. k. value*) väljale on lisatud JDK ja leJOS *bin* kausta asukoht.[*Tabel 1*]

Muutuja (*Variable*)	**Väärtus (*Value*)**	**Näide**
NXJ_HOME	Kaust kuhu on installeeritud leJOS NXJ	C:\NXJ\lejos_nxj
PATH	JDK ja leJOS *bin* kausta asukoht	C:\Program Files\Java\jdk1.6.0_06; C:\NXJ\lejos_nxj\bin;

Tabel 1. Vajalikud keskkonnamuutujad ja nende väärtused

Eclipse

LeJOS NXJ rakendust on võimalik kasutada populaarses arenduskeskkonnas Eclipse.

LeJOS NXJ kasutamine

leJOS NXJ kasutamine eeldab algteadmisi programmeerimiskeelest Java. Lahendustes on kasutatud Mindstorms NXT Tribot robotit koos standardkomplekti kuuluvate anduritega: heli-, kaugus-, puute- ja valgusandur.
Allpoolt toodud õpetuste ja näidete koostamisel on kasutatud leJOS NXJ projekti veebilehel asuvat ametlikku juhendit.

„Tere Maailm"

Eeldame, et leJOS NXJ on installeeritud korrektselt. Testimiseks kirjutame ja kompileerime esimese programmi – traditsioonilise „Tere Maailm".
Loome `TereMaailm` klassi ning meetodi `main`, kasutades tavapärast Java süntaksit.

```
public class TereMaailm {
        public static void main (String[] args) {
        }
}
```

leJOS NXJ toetab standardset `System.out.println` meetodit, väljund kuvatakse NXT LCD ekraanile.

```
public class TereMaailm {
        public static void main (String[] args) {
                System.out.println("Tere Maailm");
        }
}
```

Hetkel väljund „Tere Maailm" kuvatakse ekraanile ainult vilksamisi. Tahame, et tekst püsiks ekraanil seni, kuni vajutatakse mõnda nuppu. Selleks tuleb lisada programmi algusesse `import lejos.nxt*`, mis lubab kasutada kõiki lejos.nxt teegi klasse. Nüüd saame kasutada `Button` klassi, näiteks meetodit `waitForPress()`, mis ootab juhtploki nupuvajutust.

```
import lejos.nxt.*;

public class TereMaailm {
        public static void main (String[] args) {
                System.out.println("Tere Maailm");
                Button.waitForPress();
        }
}
```

Salvestame programmi faili *TereMaailm.java* ning avame käsurea (näiteks Command Prompt Win XP keskkonnas). Kompileerimiseks on käsurealt käsk `nxjc failinimi.java`:

```
nxjc TereMaailm.java
```

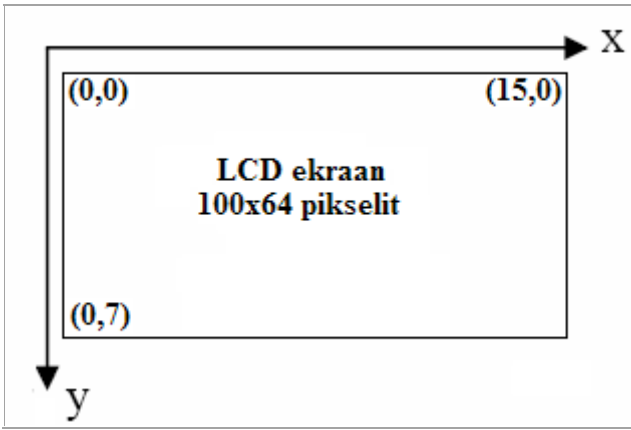
Kompileeritud programmi üleslaadimiseks ning jooksutamiseks on käsk `nxj -r`: (Veendu, et on olemas ühendus USB kaabliga ning NXT on sisse lülitatud)

```
nxj -r TereMaailm
```

Programmi tulemusena, ilmub ekraani alaosasse kiri „Tere Maailm" ning püsib seal senikaua, kuni vajutatakse mistahes juhtploki nuppu. Järgnevalt vaatleme lähemalt lejos.nxt teegi erinevaid klasse.

NXT juhtploki LCD ekraan

Klassi `LCD` kasutades saame ekraanile kirjutada ridade haaval sümboleid või piksli kaupa joonistada.[*Joonis 3*]



Joonis 3. LCD ekraan on mõõdus 100x64 pikslit. Üks rida on 16 sümbolit lai ning kokku on 8 rida. Koordinaadid (0,0) tähistavad esimest tähte esimesel real.

Näide 1: Väljastada ekraanile LCD mõõtmed pikslites.

Kasutame `LCD` klassi meetodeid: `drawString(String str, int x, int y, boolean invert)`, kus `str` on väljastatav tekst, `x` ja `y` tähistavad vastavalt sümboli

asukohta x- ja y-teljel ning *invert* lubab meil muuta teksti nähtavaks mustal taustal; *drawInt(int i, int x, int y)*, kus i on väljastatav täisarv.

```
public static void main (String[] args) {
      LCD.drawString("laius:", 0, 0);
      LCD.drawInt(LCD.SCREEN_WIDTH, 0, 1);
      LCD.drawString("k6rgus:", 0, 3);
      LCD.drawInt(LCD.SCREEN_HEIGHT, 0, 4);
      Button.waitForPress();
}
```

Kontsandid *SCREEN_HEIGHT* ja *SCREEN_WIDTH* tähistavad ekraani kõrgust ja laiust pikslites. Tulemuseks väljastatakse ekraanile: 1. reale „laius:", 2. reale „100", 4. reale „k6rgus:" ning 5. reale „64".

Ekraani puhastamiseks, näiteks kui pika programmi käigus on vaja rohkem teksti väljastada, kasutatakse meetodit *clear()* või *clearDisplay()*. Üksikuid sümboleid võimaldab väljastada meetod *drawChar(char c, int x, int y, boolean invert)*. Pikslite kaupa väljastamine on võimalik meetodi *setPixel(int rgbColor, int x, int y),* kus *rgbColor* on värvi kood RGB süsteemis.

NXT juhtploki nupud

Esimeses osas [*„Tere Maailm"*] kirjeldatud „Tere Maailm" koodis kasutasime juba klassi *Button* meetodit *waitForPress()*, mis jätab programmi ooteseisundisse seniks, kuni vajutatakse mõnda nuppu. NXT juhtploki paneelil on kokku neli nuppu, mida tähis-tatakse: *ENTER, ESCAPE, LEFT, RIGHT*. Seega võib programm igale nupuvajutusele reageerida erinevalt.

Näide 2: Iga nupuvajutuse korral väljastada ekraanile, nupu nimi, mida vajutati.

```
public static void main (String[] args) throws Exception{
      while (true){
            LCD.clear();
            if (Button.ENTER.isPressed()){
```

```
                        LCD.drawString("ENTER",0,0);
                    Thread.sleep(1000);
                    }
                    if(Button.ESCAPE.isPressed()){
                                LCD.drawString("ESCAPE",0,0);
Thread.sleep(1000);
                        break;
                    }
                    if (Button.LEFT.isPressed()){
                        LCD.drawString("LEFT",0,0); Thread.sleep(1000);
                    }
                    if (Button.RIGHT.isPressed()){
                        LCD.drawString("RIGHT",0,0);
Thread.sleep(1000);
                    }
                }
            }
```

Kasutame meetodit *isPressed(),* mis kontrollib kas nupp on alla vajutatud, lisaks hoiame nupu nime ekraanil kasutades lõimede meetodit *sleep()*(*Thread.sleep(1000)*).

Heli

Sound klassi meetodite abil on võimalik väljastada NXT juhtploki kõlari kaudu erinevaid toone erineva helitugevusega ning ka WAV formaadis helifaile.

Näide 3: Modifitseerida eelmise näite [*Näide 2*] nupuvajutusülesande lahendust nii, et erineva nupu vajutusel kõlab erinev toon.

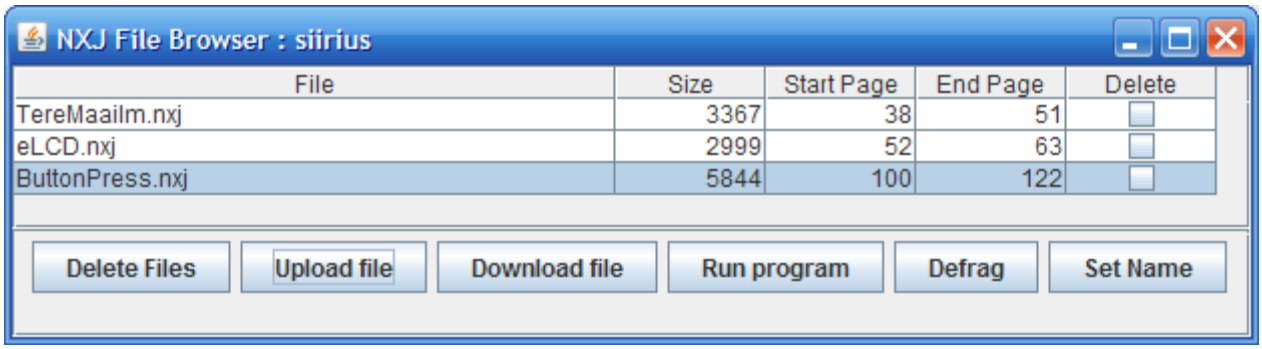
Kasutame meetodit *playTone(int aFrequency, int aDuration, int aVolume),* kus *aFrequency* on tooni sagedus hertsides (*Hz*), *aDuration* on tooni pikkus millisekundites, *aVolume* on helitugevus vahemikus 0-100 (vastab 0% kuni 100%).

```
            if (Button.ENTER.isPressed()){
                LCD.drawString("ENTER",0,0);
                Sound.playTone(10 000, 1000, 80);
                Thread.sleep(1000);

            }
```

Lisades rea `Sound.playTone(10 000, 1000, 80)`, mängitakse meile „*Enter*" nupu vajutusel ühe sekundi pikkune toon, sagedusel 10 000 Hz, helitugevusega 80%.

WAV formaadis failide ettemängimiseks on meetod `playSample(File file, int vol)`, kus `file` on *.wav* laiendiga faili nimi ja `vol` on helitugevus 0-100 (0% kuni 100%). Meetod `playSample()` eeldab WAV faili olemasolu NXT mälus. NXJ mälu sirvimiseks on kasutatakse käsurealt käsku `nxjbrowse`. Avatakse programm *NXJ File Browser*, mis võimaldab faile jooksutada, kustutada, arvutis üles või juhtplokilt alla laadida, failinime muuta ja mälu optimeerida.[*Joonis 4*]



Joonis 4. NXJ File Browser programmi kasutajaliides.

Mootorid

NXT servomootorite kontrollimiseks on klass `Motor`. Klassi `Motor` meetodite abil saab lisaks mootorite liigutamise teada ka infot mootori tegevuse kohta, näiteks saada teada kiiruse või suuna. Mootorite tarbeks on NXT juhtklotsil kolm porti, mida tähistatakse A, B ja C. Vastavalt kasutatud pordile lisame klassinimele pordi tähe ning kasutatava meetodi: `Motor.A.Forward()`, käsk mootorile A liikuda edasi. Liikuda saab ka ühe kraadi täpsu-sega, kasutades `rotateTo()` meetodit. Mootori kiirusühik on kraadi sekundis. Kiiruse määramiseks kasutatakse meetodit `setSpeed()`. Näiteks `Motor.A.setSpeed(360)` tähendab, et mootor A teeb ühe täispöörde sekundis. Mootori peatamiseks kasutatakse meeto-deid `stop()` ja `flt()` – esimene neist peatab mootori piduriga, teine aga sujuvalt ainult mootori veojõu kaotamisega. Lisaks on

võimalus hoida mootoreid paigal jõudu rakendades, kasutades `lock(int power)` meetodit, kus `power` on väärtus vahemikus 1 kuni 100. `lock()` meetod võimaldab robotil seista näiteks kaldteel või koormuse vastu.

Näide 4: Modifitseerida teises näites [*Näide 2*] välja toodud nupuvajutusülesande lahendust nii, et iga nupuvajutus tooks esile erinevas suunas liikumise kaheks sekundiks: liikumine otse ja tagurpidi, pööre vasakule ja paremale.

```
if (Button.ENTER.isPressed()) {
      LCD.drawString("ENTER",0,0);
      Motor.B.forward();
      Motor.C.forward();
      Thread.sleep(2000);
      Motor.B.stop();
      Motor.C.stop();
}
```

Antud juhul on mootorid ühendatud B ja C porti, juhul kui vajutatakse juhtplokil „Enter" nuppu, pannakse liikuma mõlemad mootorid edasi kaks sekundit ning seejärel peatatakse. Tagurpidi liikumiseks on lahendus analoogne, kasutades `backward()` meetodit. Vasakule või paremale pööramiseks liigutada ainult ühte mootorit edasi või tagasi.

Ümbritseva keskkonna tajumiseks on NXT standardkomplektis olemas neli andurit: puute-, heli-, valgus- ja kaugusandur. Järgnevalt vaatleme lähemalt kuidas neid kasutada.

Puuteandur

Andurite tarbeks on neli eeldefineeritud muutujat, igale pordile vastav muutuja: `SensorPort.S1`, `SensorPort.S2`, `SensorPort.S3` või `SensorPort.S4`. Selleks, et kasutada sensorit on vaja luua olukord, kus sensor on ühendatud porti. Selleks kasutades konstruktorit:

```
Public TouchSensor(ADSensorPort port);
```

(AD(*Analog-to-Digital*) tähendab, et analoogsisend muudetakse anduris digitaalkujule)

Enne sensori kasutamist tuleb konstruktori abil luua peameetodis uus objekt, mille parameetriks on port, kuhu andur ühendatud on.

```
TouchSensor puutePort1 = new TouchSensor(SensorPort.S1);
```

Puutesensori tarbeks on klass `TouchSensor`, millel on ainult üks meetod – `isPressed()`, mis tagastab väärtuse 0 või 1 vastavalt sellele kas nupp on vaba või alla vajutatud.

Kasutamine koodis:

```
puutePort1.isPressed();
```

Näide 5: Modifitseerida eelmise punkti [*Näide 4*] ülesande lahendust nii, et programmist väljumine toimuks „*Escape*" nupuvajutuse asemel puuteanduri abil.

Kõigepealt loome `TouchSensor` objekti ning lisame koodi uue tingimuse, mis sisaldab katkestusdirektiivi `break`. Puuteanduriks soovitatakse kasutada neljandat porti, sest on teada, et mõned andurid, näiteks ultraheli- ja valgusandur ei pruugi funktsioneerida selles pordis korrektselt.[20]

```
import lejos.nxt.*;
public class ButtonPress{
      public static void main (String[] args) throws Exception{
            TouchSensor puutePort4 = new TouchSensor(SensorPort.S4);

            while (true){
                  ...
                  if (puutePort4.isPressed()) break;
                  ...
            }
      }
}
```

Lisatud tingimus jääb ootama kuni neljandas pordis olev puuteandur tajub vajutust, seejärel katkestab programmi töö käsuga `break`.

Heliandur

Heliandur võimaldab mõõta helirõhutaset (*SPL - Sound pressure level*) dB või dBA (A-filtriga korrigeeritud helirõhu taseme mõõtmiseks) mõõtühikutes. Mõõdetav vahemik on ligikaudu 30 kuni 90 dB, kuid tagastatavad väärtused on 0 kuni 100%. Helianduri konstruktor on (juhul kui `boolean dba` on `1` siis mõõdetakse DBA, kui `0` siis DB):

```
public SoundSensor(ADSensorPort port, boolean dba)
```

Meetodeid on helianduril kaks: `readValue()` sensori väärtuse lugemiseks (0 kuni 100%) ja `setDBA(boolean dba)` mõõteviisi vahetamiseks.

Näide 6: Teha uus programm, mis käsib robotil kuulata ümbrust ning kuvab mõõtmistulemused ekraanile graafiku kujul. Kasutada meetodit `LCD.setPixel(1, int x, int y)`, mis värvib ekraanil piksli (x,y) koordinaatidega .

```
import lejos.nxt.*;
public class Helitugevus {
      public static void main (String[] args) throws Exception{
            SoundSensor heli = new SoundSensor(SensorPort.S1, true);
            while (!Button.ESCAPE.isPressed()) {
                  LCD.clear();
                  for(int i=0;i<100;i++){
                  LCD.setPixel(1,i,63 - (heli.readValue()/2));
                  Thread.sleep(20);
                  }
            }
      }
}
```

Tsükli jätkutingimuseks on määratud `!Button.ESCAPE.isPressed()`, see tähendab et `while()` tsükkel töötab seni kuni „*Escape*" nuppu pole vajutatud. Kuna LCD ekraan on mõõtmetes 100x64 pikslit, siis kasutame `for` tsüklit, mis käib läbi juhud `i = 0` kuni `i = 100` ning märgib iga `i` korral piksli mille kõrgus on: `63` miinus helianduri väärtus (0%-100%) jagamisel kahega – kõrguste vahe ekraanil on `13` (kõige kõrgem helitase) kuni `63` (kõige madalam helitase).

Valgusandur

Valgusandur mõõdab valguse tugevust ruumis ja ka pinnalt tagasipeegeldunud valguse inten-siivsust. Valgusanduris on kaks LED'i, üks kiirgab valgust ja teine võtab vastu. `Floodlight` tähistab valgust kiirgavat LED'i - kui see välja lülitada, siis mõõdetakse valguse tugevust ruumis. Kui valgustav LED on sisse lülitatud, siis saab mõõta pinnalt tagasi peegeldunud valguse intensiivsust. Valgusanduri konstruktor (`boolean floodlight` väärtus `1` või `0` vastavalt, kas valgustus sisse või välja lülitatud):

```
public LightSensor(ADSensorPort port, boolean floodlight)
```

Enne valgusanduri kasutamist tuleks kalibreerida mõõdetav piirkond, määrates valge ja musta pinna. Selleks kasutame vastavaid meetodeid `calibrateHigh()` ja `calibrateLow()` – mille abil määrame pinnalt peegeldunud valguse maksimumi(valge) ning miinimumi(must) meetodi `readValue()` tarbeks.

Näide 7: Aseta lauale NXT roboti (valgusandur suunatud alla) ette valge paber. Kirjutada programm, mis paneks roboti liikuma otse edasi ning peataks roboti, kui valgusandur tajub valget paberit.

```
import lejos.nxt.*;
public class ValgusAndur {

            public static void main (String[] args) throws Exception {

            LightSensor valgus = new LightSensor(SensorPort.S3);
            //alustame kalibreerimist
            LCD.drawString("m22ra valge:", 0, 0);
            Button.ENTER.waitForPress();
            valgus.calibrateHigh();
            LCD.drawInt(valgus. readValue(), 0, 1);
            Thread.sleep(2000);
            LCD.clear();
            LCD.drawString("m22ra must:", 0, 0);
            Button.ENTER.waitForPress();
            valgus.calibrateLow();
            LCD.drawInt(valgus. readValue(), 0, 1);
            Thread.sleep(2000);
                //kalibreerimine valmis
            //alustame programmiga
            LCD.clear();
            LCD.drawString("Alusta programmiga?", 0, 0);
            LCD.drawString("vajuta Enter", 0, 1);
```

```
            Button.ENTER.waitForPress();
            Thread.sleep(1000);
            while(true){
                  Motor.B.forward();
                  Motor.C.forward();
                  if(valgus. readValue() > 98) break;
            }
      }
}
```

Programmis kasutatakse mõõtetulemuse väljastamiseks meetodit *readValue(),* mis tagastab mõõtmistulemused protsentides (0% ja 100% on vastavad kalibreeritud väärtused). Juhul, kui kalibreerimist pole läbi viidud, saame väljastada tulemusi meetodiga *readNormalizedValue()*, mis väljastab tulemuse vahemikus 0 kuni 1023 vastavalt peegeldunud valguse intensiivsusele. Kalibreerime valgusanduri valge ja musta paberiga. Valge paber on peale kalibreerimist väärtusega 100. Alustame liikumist, kui *readValue()* väärtus on üle 98 (jätame sisse mõõtevea võimaluse), siis lõpetame programmi töö.

Ultraheliandur

Ultraheliandur (kaugusandur) suudab märgata objekti vahemaaga 0-255 cm, veaga +/- 3 cm. Tema vaateväli on ligikaudu 30°. Ultraheliandur (nimetatakse ka kaugusanduriks) on oma olemuselt ultrahelilokaator.

Ultraheliandur kasutab I2C protokolli, *UltrasonicSensor* klass on ühtlasi *I2CSensor* klassi laiendus. Kaugusanduri konstruktor:

```
UltrasonicSensor(I2CPort port)
```

Põhiliselt kasutame meetodit *getDistance(),* mille abil saame mõõta objekti kaugust andurist sentimeetrites (vahemikus 0 kuni 255).

Näide 8: Kirjutada uus programm, mis valvaks 40 cm laiust ala enda ees – juhul, kui objekt satub roboti valvatavale alale tagastatakse kaugus ultraheliandurist.

```
import lejos.nxt.*;
public class KaugusAndur {

	public static void main (String[] args) throws Exception {

		UltrasonicSensor Kaugus = new
	UltrasonicSensor(SensorPort.S1);

		while(true){
			LCD.drawString("...Valvan...", 0, 0);
			if(Kaugus.getDistance()<41){
				LCD.drawString("Objekt leitud!", 0, 0);
				LCD.drawString("Kaugus objektist:", 0, 1);
				LCD.drawInt(Kaugus.getDistance(), 0, 2);
				LCD.clear();
			}
			if(Button.ESCAPE.isPressed()) break;
		}
	}
}
```

Kasutame tingimust `if(Kaugus.getDistance()<41)` välistamaks objektide mõõtmise, mis asuvad kaugemal kui 40 cm.

Järgnevalt teeme läbi näiteülesande, kus on kasutame mootoreid, LCD ekraani ning kõiki eelpool vaadeldud andureid.

Näiteülesanne kasutades LeJOS NXJ'i

Ülesande nimi: Koolikoti otsija

Tase: Keskmine

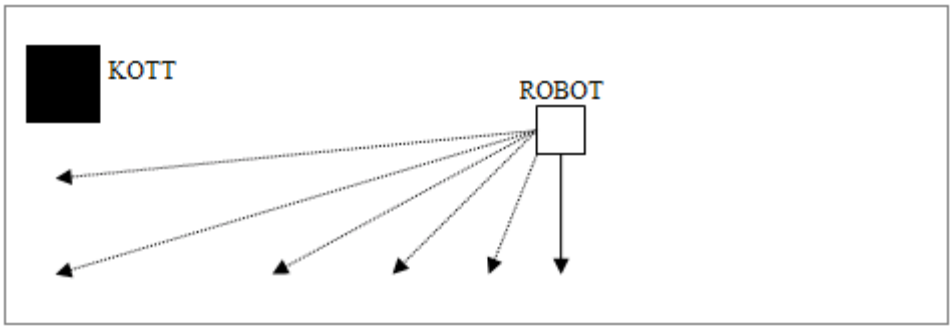
Eesmärgid: Muutujate, tsüklite, konstruktorite ja tingimusdirektiivide kasutamine Javas.

Ülesande täitmiseks vajalikud tarvikud:

- Robot: Standard LEGO Mindstorms NXT Tribot robot - valgusandur eesosas, suunatud ette ning puuteandur taga. Kasutatud pordid: mootorid portides B ja C, kaugusandur: port 1, heliandur: port 2, valgusandur: port 3, puuteandur: port 4 [*Lisa 1: Pilt kasutatud robotist*].
- Muud tarvikud: Heledate seintega tuba ja must koolikott.

Ülesande kirjeldus:

Heledas toas on seina ääres must koolikott, programmeerida robot selliselt, et mistahes ruumi punktis alustades otsib robot seina ääri seni, kuni leiab musta koolikoti. [*Joonis 5*]



Joonis 5. Otsimise põhimõte - sõidame otse, jõudes seinani, robot tagurdab ja keerab parema-le. Alustame programmi algusest, seni kuni leiame musta koti.

Programmi koostamisjuhend:

Klassi `ValgusAndur` all defineerida kaks meetodit – `main` ja `kalibreeri`. Meetodi `kalibreeri(LightSensor valgus)` abil kalibreerime valgusanduri, määrates musta ja valge pinna [*2.2.9 Valgusandur, Näide 7*]. Peameetodis `main` loome konstruktorite abil uued objektid valgus-, puute-, kaugus- ja helianduri tarbeks. Kasutame

eelnevalt defineeritud abimeetodit *kalibreeri* valgusanduri kalibreerimiseks. Seejärel loome *while* tsükli, mille all kirjeldame kaks tingimus-direktiivi *if* – Objekt on kaugemal kui 23 cm, objekt on lähemal kui 24 cm. Kaugusanduri tingimuse, lähemal kui 24 cm, alla kirjeldame lisaks veel kaks tingimusdirektiivi – kas valgus on heledam kui 4 või tumedam kui 5 (must on peale kalibreerimist 0 ja valge 100). Juhul kui valgus on tumedam kui 5, lõpetame programmi töö ja väljastame ekraanile teksti „Kott leitud". Teisel juhul, kui valgus on heledam, tagurdame ja keerame paremale.
Lisame veel kaks tingimusdirektiivi eriolukordade tarbeks: puuteandurile, juhuks kui robot tagurdab millegi otsa ning helianduri, mille abil võime programmi töö lõpetada (näiteks käteplaksu abil), kui robot kusagile kinni jääb.

Võimalikud probleemid:

Valgusandur ei ole täpne ning võib juhtuda, tagastatavad lugemid võivad vahepeal teha kõrvalekaldeid. Tuleb arvestada, et kaugusandur asub roboti peal ning mõõdab umbes 20 cm kõrgusel, madalaid objekte ei märgata. Tuleks lülitada valgusanduri LED valgustus sisse (*valgus.setFloodlight(true)*), kuna katsed näitasid, et nii on värvide eristamine täpsem.