ROBOTC õppematerjal

Autor: Heilo Altin
MTÜ Robootika
2015

„Me ei tee seda selleks, et inimesed ehitaks roboteid vaid selleks, et robotid ehitaksid inimesi“ – tundmatu autor

Materjali valmimist toetasid

Euroopa Liit
Euroopa Sotsiaalfond

Eesti
tuleviku heaks

HITSA
Hariduse Infotehnoloogia
Sihtasutus

ProgeTiiger

Sisukord

Sissejuhatus

See väike raamat on loodud eesmärgiga õpetada programmeerimist ROBOTC keskkonnas EV3 robotite abil. ROBOTC on C keelel põhinev hariduslike robotite programmeerimiseks loodud keskkond. C-keel on aluseks paljudele tuntud lahendustele nagu Google, Oracle, Adobe Acrobat. C-keel on mõjutanud paljusid teisi programmeerimiskeeli nagu Perl, Java, PHP jt. C-keele tundmine avardab uksi ning annab hea baasi tulevikuks!

Programmeerimiskeele õppimine ei ole sama, kui õppida selgeks võõrkeelt. Uue keele omandamine on lihtsam, kuna tunneme juba sõnu emakeeles. Oskame moodustada lauseid ja keelekonstruktsioone. Võõrkeelt õppides on meil võimalik luua seoseid emakeelega ning jätta meelde uusi sõnu ning õppida kasutama käändeid. Programmeerimiskeele õppimine on raskem, kui eelnevalt ühtegi teist programmeerimiskeelt ei tunne. Puudub võimalus seostada õpitut analoogse olukorraga ning seetõttu oleme justkui visatud külma vette. Selles olukorras toimetulekuks on meil vaja abi, et aru saada, kuidas üks või teine käsk töötab ning mis selle tulemusena juhtub. Õppides programmeerima ainult arvuti abil, on programmil vähe võimalusi meile oma tööst märku anda. Ainukeseks väljundiks on ainult ekraan. Seda olukorda saab parandada roboti abil. Robot annab võimaluse kasutada enda mootoreid ja andureid. Saame ehitada programme, mis panevad roboti suhtlema välismaailmaga nii nagu inimesed seda teevad. Me ei põrka joostes puuga kokku kuna näeme seda. Samamoodi saame programmeerida roboti, mis sõidab ning väldib kaugusanduri abil takistusi. Nii loome analoogi jooksjast, mis koosneb robotist ja programmist. Selline õppimine on efektiivsem ja parem, sest meie programmidel on füüsiline tagasiside. *Programmeerimiskeel on vahend meie mõtete kirjapanekuks ja robot on vahend nende mõtete näitamiseks.* Aga alati ei pruugi õigesti töötav

programm panna robotit tegutsema nii nagu vaja. Robot peab olema terve, aku laetud ning mootorite ja andurite juhtmed korralikult ühendatud.

Soovin edu ja jõudu ROBOTC keele õppimisel!

Õppekava

Kursuse taotletavad pädevused.

Õpilane:

1. Tunneb programmeerimiskeskkonda ROBOTC. Tunneb selle menüüsid ning oskab seda kasutada roboti ühendamiseks juhtmega ja juhtmevabalt, püsivara uuendamiseks, andurite ja mootorite seadistamiseks, programmide kompileerimiseks, robotisse laadimiseks, abi ja näidisprogrammide leidmiseks.
2. Oskab programmeerida robotit sõitma otse, keerama, kasutama mootorite pööreteandureid ning tsüklit tegevuste kordamiseks. Teab ning oskab kasutada for ja while tsükleid.
3. Oskab ja tunneb roboti andureid nagu sonar, valguse-, güro- ning puuteandur; oskab programmeerida robotit kuvama andurite näite ekraanil; oskab näite töödelda ning panna robot liikuma anduritest sõltuvalt; tunneb loogilist võrdlusmärki „==“ ja oskab seda kasutada.
4. Oskab kasutada loogikatehteid nagu AND ja OR tingimuslausetes; oskab kasutada koos kahte andurit tingimuslauses

Programmeerimiskeskkonna tundmaõppimine

Programmeerimiskeskkonna tundmaõppimiseks on loodud video, mis hõlmab endas töötamist ROBOTC-s. Sul läheb selleks vaja EV3 robotit ning vastavat tarkvara. Videos näidatud tarkvara versioon on 4.50

Video leiad, kui lähed aadressile:
https://www.youtube.com/watch?v=-4t2cNP6i34

Paneme roboti liikuma

Õpieesmärgid

- Roboti liigutamine otse, tagasi ja keeramine
- Mootorikäskude tundmaõppimine
- Roboti liigutamine läbi mootorite sünkroniseerimise
- Tsükkel

Ava programmeerimiskeskkond ning proovi järgmist koodi:

```
/*
task main on esimene osa programmist,
mille robot k2ivitab. Kik k2sud tuleb
panna loogeliste sulgude vahele
*/
task main()
{
	//mootor B ja mootor C kiiruseks seatakse 50
	setMotorSpeed(motorB, 50);
	setMotorSpeed(motorC, 50);
	//oodatakse 1 sekund (1000 millisekundit)
	sleep(1000);
}
```

Kas robot sõitis otse ühe sekundi?
Ülesanded:

1. Muuda programmi nii, et robot sõidaks otse kaks sekundit täiskiirusel (100).
2. Muuda programmi nii, et robot keeraks üks sekund paremale ja seejärel üks sekund vasakule. Kurvi raadius ei ole oluline.
3. Kas oskad teha programmi ümber nii, et robot sõidaks kaks sekundit tagasi? (Vihje: kiiruse muutmine)

Kas täheldasid, et robot ei pruugi pikema sõidu korral päris otse sõita? Kuigi mootorid on samad ning mõlemale mootorile antakse samasugused käsklused, tekib reaalselt kõrvalekalle sirgest

sõidujoonest. See pole mitte ainult LEGO robotite vaid robootika üldine probleem – kuidas panna robot otse sõitma. Üks viis probleemi lahendamiseks on pööreteandurite kasutamine. Mootorite sisse on ehitatud pööreteandurid, mis loevad, kui palju on mootor pööranud. Reaalne täpsus EV3 roboti korral on 2°-3°. Kui roboti kontroller saab aru, et roboti üks mootor hakkab ette jõudma, siis vähendab ta selle mootori kiirust. Selle protsessi toimimist on võimalik näha, kui robot korrigeerib sõidu ajal oma suunda.

Vaata järgmist koodi ja loe käskude selgitusi.

```
/*
parempoolne mootor on pordis C
vasakpoolne mootor on pordis B
*/
task main()
{
	setMotorSyncEncoder(motorB, motorC, 0, 360, 50);
	waitUntilMotorStop(motorB);
}
```

Käsk:

setMotorSyncEncoder(mootor1, mootor2, pöörderaadius, kraadide arv, kiirus)

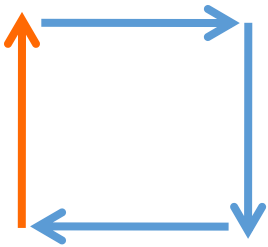
mootor1 – vasakpoolne mootor (motorB)
mootor2 – parempoolne mootor (motorC)
pöörderaadius – arv, mis on -100 kuni 100

- 100 annab positiivse kiiruse mootor1-le ja negatiivse kiiruse mootor2-le.
- -100 annab negatiivse kiiruse mootor1-le ja mootor2-le.
- 0 annab võrdse kiiruse mootor1-le ja mootor2-le.
- 50 määrab kiiruse mootor1-le ja 0 kiiruse mootor2-le.
- -50 määrab nullkiiruse mootor1-le ja kogu kiiruse mootor2-le.

kraadide arv – See määrab, kui palju robot sõidab. 360° paneb roboti sõitma ühe ratta täispöörde, 720° aga kaks täispööret
kiirus – määrab roboti mootori liikumise kiiruse vastavalt pöörderaadiuse väärtusele

Proovi eelpooltoodud programm järgi. **Mida robot tegi?**
Ülesanded:

4. Muuda programmi nii, et robot sõidaks otse kaks täispööret kiirusega 100
5. Muuda programmi nii, et robot keeraks esmalt paremale ja siis vasakule. Kurvi raadius pole oluline
6. Muuda programmi nii, et robot keeraks 90° ehk täisnurga. Üks võimalus oleks muuta pöörderaadiust ja kraadide arvu.
7. Kui oled saanud roboti keerama 90°, siis tee programm, mis paneks roboti ruudu trajektoori sõitma.



8. Mitut käsku läheks tegelikult vaja, et panna robot sõitma ruudu trajektoori, kui oleks võimalik neid käske korrata?

Programmeerimiskeeles on võimalik kasutada tsükleid ehk kordusi. Õpime seda ruudu ülesande abil.

```
/*
parempoolne mootor on pordis C
vasakpoolne mootor on pordis B
*/
task main()
{
      setMotorSyncEncoder(motorB, motorC, 50, 410, 50);
      waitUntilMotorStop(motorB);
      setMotorSyncEncoder(motorB, motorC, 0, 400, 50);
      waitUntilMotorStop(motorB);
}
```

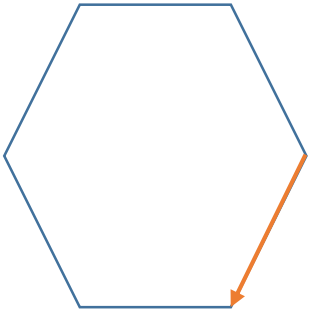
Antud programmis on kaks käsku, mida korratakse. Esimesed kaks käsku panevad roboti keerama täisnurga ning järgmised kaks käsku panevad roboti otse sõitma. Tsüklis näeks programm välja selline:

```
/*
parempoolne mootor on pordis C
vasakpoolne mootor on pordis B
*/
task main()
{
      for (int i=1; i<=4; i++)
      {
        setMotorSyncEncoder(motorB, motorC, 50, 410,
        50);
        waitUntilMotorStop(motorB);
        setMotorSyncEncoder(motorB, motorC, 0, 400, 50);
        waitUntilMotorStop(motorB);
      }
}
```

Tsükkel algab käsuga „for“, „int i=1“ tähistab muutuja i algväärtust. Tsükkel kestab seni, kuni i on väiksem või võrdne 4. Iga tsükli kordusega lisatakse i-le arv 1. Seega kasvab muutuja i iga tsükli käiguga ühe võrra kuni 4-ni. Peale „for“ tsükli on olemas ka teisi tsüklitüüpe. Üks levinumaid on „while“ tsükkel, kuid seda kasutame kursuse lõpupoole.

Ülesanded:

9. Kas oskad muuta programmi nii, et robot sõidaks heksagoni? Kasuta tsüklit ja eelnevaid käske pööramiseks ja otse sõitmiseks.



Kus ma olen?

Inimesed kasutavad ümbritsevast maailmast aru saamiseks silmi, kõrvu, haistmist, kompimist ja maitsmist – meeli. Robotile on samuti antud meeleline taju. Me ei saaks programmeerida robotit, mis ei suuda avastada seina või leida üles joont, kui poleks andureid. Andurid on robotite meelteks, mis võimaldavad mõõta maailma isegi inimestest kõrgemal tasemel. Vastava anduri abil saab robot mõõta magnetvälja, kuulda ja näha inimpiiridest kaugemale.

Kõigepealt peame fikseerima, milliseid andurid on millises pordis. Seda saab teha menüüst *„Tools->Motors and Sensors Setup“*

Proovime lugeda näite roboti neljalt andurilt ning need roboti ekraanil kuvada.

```
#pragma config(Sensor, S1, puute, sensorEV3_Touch)
#pragma config(Sensor, S2, gyro, sensorEV3_Gyro)
#pragma config(Sensor, S3, v2rv, sensorEV3_Color)
#pragma config(Sensor, S4, kaugus,
sensorEV3_Ultrasonic)
//*!!Code automatically generated by 'ROBOTC'
configuration wizard
task main()
{
   while(true){
      displaySensorValues(5, kaugus);
   }
}
```

See kood kuvab roboti ekraanil kaugusanduri näitu. Näit kuvatakse ekraanil real 5.

Ülesanded:

10. Muuda programmi nii, et näeksid näite ka güro-, puute- ning valguseandurilt.

Nüüd, kui oleme saanud anduritelt lugemid, paneme roboti nende järgi liikuma.
Proovi järgnevat koodi, mis paneb roboti sõitma kuni sonar näeb takistust lähemal kui 20 cm.

```
#pragma config(Sensor, S1, puute, sensorEV3_Touch)
#pragma config(Sensor, S2, gyro, sensorEV3_Gyro)
#pragma config(Sensor, S3, v2rv, sensorEV3_Color)
#pragma config(Sensor, S4, kaugus,
sensorEV3_Ultrasonic)
//*!!Code automatically generated by 'ROBOTC'
configuration wizard
task main()
```

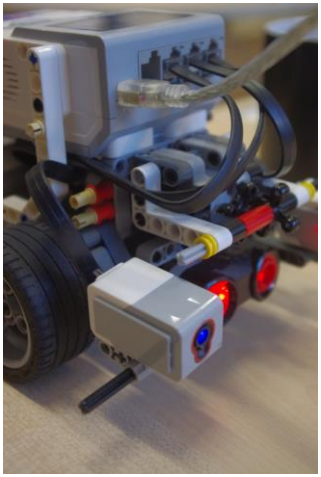
```
{
  setMotorSync(motorB, motorC, 0, 50);
  waitUntil(SensorValue(S4)<=20);
}
```

setMotorSync paneb roboti otse sõitma (pöörderaadius on 0) kiirusega 50. Robot katkestab sõitmise siis, kui andur näeb midagi lähemal, kui 20 cm.

Ülesanded:

11. Tee programm, mis ei lase robotit enne sõitma, kui kauguseandur näeb midagi lähemal kui 20 cm. Vihje! SetMotorSync käsk paneb roboti liikuma, kui käsule järgneb ootamine, nt *„sleep(1000)“.* See paneb roboti üheks sekundiks ootele.
12. Tee programm, mis ei lase enne robotit sõitma, kui pole vajutatud puuteandurit! Vihje! Programmeerimiskeeles kasutatakse võrdlemiseks kahte võrdusmärki: „==“. Näiteks, kui kasutaja tahab, et programm jätkaks oma tööd, kui puuteandur väljastab numbrit 1, siis peab kasutama käsku: *„waitUntil(SensorValue(S1)==1)“*

13. Tee programm, mille järgi sõidab robot seni, kuni valguseandur pole näinud heledat valgust. Vaadake ekraanilt anduri näitu, et aru saada, milline on hele valgus (nt mobiili taskulamp) ning milline on ruumis ümbritsev valgus. Selles ülesandes peab olema valgusanduri režiimiks „ambient“ „Motors and Sensors Setup“ alt. Lisaks peab olema andur paigaldatud pildil näidatud viisil.



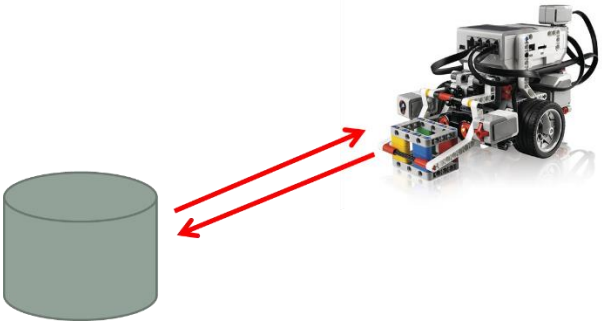
14. Tee programm, mis paneks roboti keerama koha peal 90° güroanduri järgi. Tehke enne kindlaks, mis suunas güroanduri näidud suurenevad ning mis suunas vähenevad. Esimese asjana on programmis mõistlik güroandur ära nullida. See tähendab, et kui robot keeras 90° ning soovite programmi uuesti käivitada, siis on güroanduri näit juba 90°. Güroanduri nullides kirjutatakse vana väärtus üle arvuga 0. Nullimise käsk:

```
resetGyro(gyro);
wait(3);
```

Veenduge, et roboti sisselülitamisel või güroanduri ühendamisel oleks andur liikumatult. Muidu tekib triiv ning robot ei saa anduri järgi keerata.

15. Tee programm, mis järgib algoritmi:
    a. Robot alustab sõitu, kui vajutatakse puuteandurit
    b. Robot sõidab otse seni kuni näeb sonariga takistust, mis on lähemal, kui 30 cm

c. Robot keerab güroanduri abil 180° ringi ning sõidab tagasi samapalju maad, kui kulus takistuseni sõiduks



Vihje: Lisa see programmilõik oma programmis kohta, kus robot peaks hakkama tagasi sõitma. See loeb ära, palju on mootorB sõitnud ning paneb roboti sama palju tagasi sõitma.

```
int i=getMotorEncoder(motorB);
resetMotorEncoder(motorB);
setMotorSyncEncoder(motorB, motorC, 0, i, 30);
waitUntilMotorStop(motorB);
```

Tehisintellekt?

Olete väga tublid, et olete nii kaugele jõudnud. Praeguseks olete selgeks saanud elementaarsema roboti liigutamise ning anduritelt saadud infotöötlemise põhimõtted. Nüüd oleks aeg anda robotile rohkem tarkust, et ehitada keerulisemaid süsteeme. Tehisintellektist jääme esialgu küll kaugemale, sest selleks pole roboti kontroller piisavalt võimekas.

Õpieesmärgid

- Loogikatehete AND, OR tundmaõppimine
- Kahe anduri korraga kasutamine

- Tingimuslause kasutamine

Loogikatehe AND tähendab seda, et võrreldakse kahte sisendit. Kui mõlemad sisendid on tõesed, antakse väljundiks tõene. Selgitav tabel on allpool.

Tabel 1. AND loogikatehe

Sisend 1	Sisend 2	AND tehte tulemus
Tõene	Tõene	Tõene
Tõene	Väär	Väär
Väär	Tõene	Väär
Väär	Väär	Väär

OR loogikatehe annab tõese väärtuse siis, kui vähemalt üks sisenditest on tõene.

Tabel 2. OR loogikatehe

Sisend 1	Sisend 2	OR tehte tulemus
Tõene	Tõene	Tõene
Tõene	Väär	Tõene
Väär	Tõene	Tõene
Väär	Väär	Väär

Proovige programmi, mis paneb roboti tegema heli, kui vajutatakse korraga roboti paremat ja vasakut nuppu. Siin programmis kasutatakse while tsüklit, mis kestabseni kuni programm töötab. Tsükli sees on tingimuslause *„if“*, kuhu on kirja pandud kaks tingimust. Esimene neist on „`getButtonPress(buttonLeft) == 1“;` sellega kontrollitakse, kas vasakpoolset nuppu roboti kontrolleril on vajutatud. Teine tingimus on `getButtonPress(buttonRight) == 1;` sellega kontrollitakse, kas parempoolset nuppu on vajutatud. Järgmisena

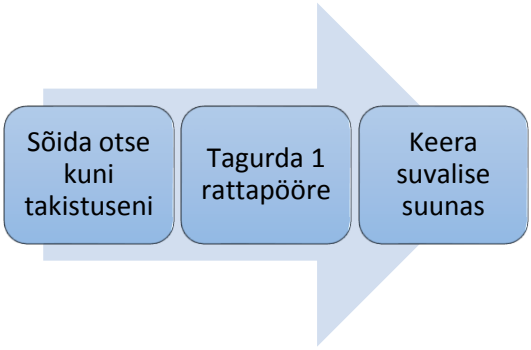
seatakse helitase 100 % peale ning mängitakse vastavat helifaili. Loogikatehet AND tähistab programmeerimiskeeles „&&“.

```
#pragma config(Sensor, S1, puute, sensorEV3_Touch)
#pragma config(Sensor, S2, gyro, sensorEV3_Gyro)
#pragma config(Sensor, S3, v2rv, sensorEV3_Color,
modeEV3Color_
#pragma config(Sensor, S4, kaugus,
sensorEV3_Ultrasonic)
//*!!Code automatically generated by 'ROBOTC'
configuration wizard
task main(){
   while(true){
      if((getButtonPress(buttonLeft) ==
      1)&&(getButtonPress(buttonRight) == 1))
      setSoundVolume(100);
      playSound(soundBeepBeep);
      wait(0.5);
      }
   }
}
```

Ülesanded:

16. Muuda olemasolevat programmi nii, et robot teeks heli siis, kui on vajutatud vähemalt ühte nuppudest (vasak või parem).
17. Lisage uus tingimuslause, kus robot kontrollib ülemise ja alumise nupu vajutamist ning lisage nende nuppude kontrollimiseks enda valikul tingimuslause (AND, OR). Andke töötav robot kaaslastele ning laske neil ära arvata, millise tingimuslausega on tegu.
18. Programmeerige koristajarobot, mis sõidaks mööda tuba ringi. Algoritm näeb välja järgmine:
    Takistuste leidmiseks kasutage nii puuteandurit, kui sonarit. Sonar ei pruugi näha objekte, mille helilaineid peegeldavad omadused pole piisavalt head. Siinkohal on abiks puuteandur,

mille abil saab robot aru, kui on takistusele otsa sõitnud.



Suvalise arvu genereerimiseks kasutage käsku:

```
int suvaline_arv=(rand() % „muutuse suurus"+algväärtus);
```

Muutuse suurus tähistab vahemikku, mille sees suvalist genereeritakse. Algväärtus on minimaalne väärtus, millest väiksemat arvu ei genereerita. Näiteks:

```
int suvaline_arv=(rand() % 400+100);
```

genereerib arvu vahemikus 100…500.

Otsustage iseseisvalt, kuidas kasutada suvalist arvu roboti suvalises suunas keerama panemiseks. Selleks võib olla pööramise aeg, pööramise kiirus või mootorite kraadide arv.