Tallinna Tehnikaülikool
Mehhatroonikainstituut

Mikrokontrollerid ja praktiline robootika - MHK0011

aruanne

Kapten: Heiko Pikner

Üliõpilased: Matthias Kuus
Rando Pokk

2009

Sisukord

Ülesanne

Robotiklubi viis 2009. aastal läbi kevadise robotiehitamise kursuse TTÜ tudengitele. Kursus toimus õppeaine „Mikrokontrollerid ja praktiline robootika“ (MHK0011) raames. Eesmärk oli anda osalejatele esmane arusaam robotitest ja nende programmeerimisest. Kursus lõppes võistlusega Baltic Robot Sumo 2009 Tallinn Cup, mis toimus 04.04.2009. Sellest lähtuvalt moodustati võistkonnad, mida juhtisid TTÜ Robotiklubi liikmed.

Võistlus

Roomba sumorobotite võistlus toimub kahe roboti vahel spetsiaalsel väljakul. Ühes võistluses on kolm roundi. See robot, kes teise välja lükkab, võidab roundi ja saab punkti. Võistluse võidab aga robot, kes saab rohkem punkte. Ühe roundi kestuseks on maksimaalselt kolm minutit. Aega hakatakse mõõtma 5 sekundit peale starti. Varem ei tohi ka robotid tegutsema hakata.

Väljak

Väljak on tehtud terasest plaadist ja asetseb ülejäänud põrandast natuke kõrgemal. Ringikujuline väljak on pealt kaetud must mati värviga. Selle servas on 5cm valge joon. Väljaku ümbruses peab võistluse ajal olema vaba ala. Sinna ei tohi ka võistluse ajal keegi siseneda. Isikul, kes asetab roboti väljakule, on aega 5 sekundit alast lahkumiseks.

Nõuded robotile

Võistlusel kasutatav robot peab olema originaalne või täiendatud iRobot Roomba™ tolmuimeja. Keelatud on muuta veosüsteemi (mootorid, reduktorid jms) ja akut. Samuti peab roboti diameeter samaks jääma. Roomba kõrgus ei ole piiratud, maksimaalne kaal võib olla 3,8kg.
Roomba peab olema täielikult autonoomne. Selleks võis sinna lisada mikrokontrolleri. Lubatud on kasutada ka traadita ühendust ja välist arvutit, kus jookseb roboti juhtalgoritm. Ruumi saamiseks võib Roombalt originaalse imemisosa eemaldada.

Ideed

Kaugusandureid rohkem ja laiali

Kasutasime viit kaugusandurit. Idee oli need paigutada rohkem laiali, et robot peaks kogu ümbrusest pildi saamiseks end vähem pöörama. Reaalselt sai pandud ikkagi esimesed kolm üksteisele lähemale ning tagumised kaks omavahel lähemale, kuna nii oli konstruktsiooniliselt lihtsam.

Sujuv liikumine

Kui Roomba™ tuvastab vastase mitte-keskmise kaugusanduriga, peab ta korrigeerima oma liikumissuunda sõidu ajal. Roomba ise võimaldab seda, kuid meil jäi puudu ajast programmi koodi kallal töötamisel. Samas on Roomba™ kiirus piisavalt väike, nii et seistes kursi korrigeerimine ei osutunud komistuskiviks.

Robot raskemaks teha

Otsustasime teha roboti raskemaks lubatud maksimaalse kaaluni, et vastasel oleks meie robotit ringist raskem välja tõugata ning meie robotil oleks rohkem jõudu, et vastast kergemini välja tõugata. Kasutasime „ballastiks” akut.

Vastane hooga väljakult välja lükata

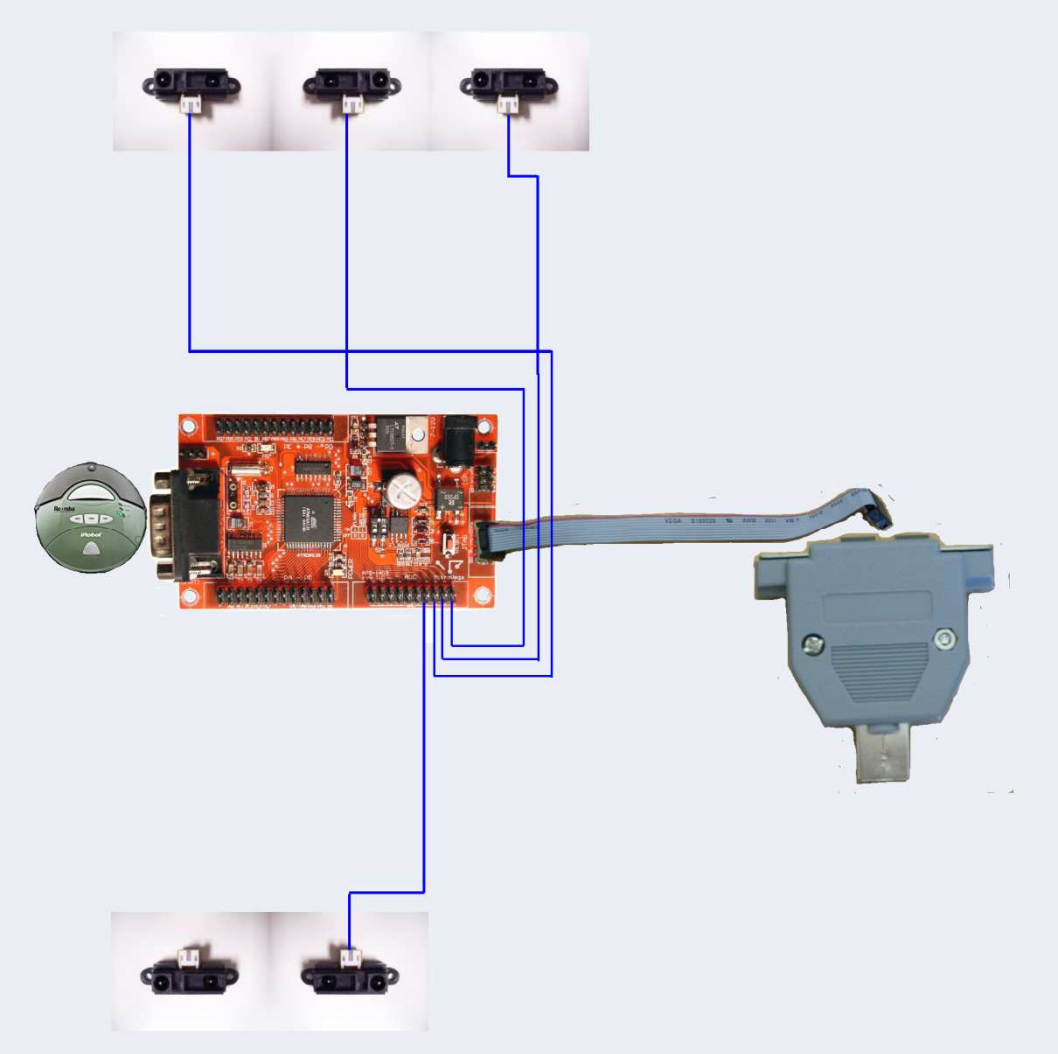
Kui vastane on otse ees ning põrkeandurid on rakendunud (võitlus asend), ei ole otstarbekas enam kulutada aega *cliff* ehk ääre-andurite kontrollimisele. Eesmärk on vastane kindlalt välja lükata esimesena. Selleks tegime väikse muudatuse koodis.

Roboti kirjeldus

Ehitasime sumoroboti iRobot Roomba™ 416™ mudelist, eemaldasime kõik sisemised puhastusharjad. Roboti kaalu tõstmiseks etteantud väärtuseni kasutasime metallist raskuseid. Parema tasakaalu saavutamiseks modifitseerisime veorattaste kinnitusi. Ka taha sai toestav tugiratas pandud.

Eemaldatud imemissüsteemi kohale seadsime Atmel Atmega128 kiviga kontrollerplaadi, JTag'i ja hulgaliselt juhtmeid, mis ühendasid kontrollerit ja andureid. Atmega plaat ühendus Roomba™ ga läbi RS-232 jadaliidese. Atmega plaadi pingeregulaatori jahutamiseks sai sinna suur radikas külge pandud. Ette paigutasime kolm ja taha kaks infrapunakaugusandurit. Lisaks sellele lisasime Roombale erinevatesse kohtadesse LED-e.

Elektroonika üldskeem



Enne võistlusi ütles üks tagumistest anduritest üles ning me olime sunnitud selle välja lülitama, ka koodis.

Algoritmi selgitus

Programmi algoritm küsitleb lõpmatu tsükli sees kõiki andureid. Seejärel käiakse läbi rida *if* tingimuslauseid, mille sees uuritakse, kas mõni andur on „rakendunud" ja loetakse kaugusandurite infot. Algoritmi tähtsaim osa põhineb viie kaugusanduri küsitlemisel. Kolm kaugusandurit on ees, kaks taga. Kui tuvastatakse vastane ükskõik kumma tagumise anduri lähedal, siis pöörab robot end senikaua, kuni vastane satub keskmise kaugusanduri nägemisvälja. Kui vastane on esimese vasaku või parema anduri nägemisväljas, korrigeerib robot oma sihti, kuni vastane on keskmise anduri alas. Edasi sõidab robot vastase suunas, vajadusel korrigeerib suunda. Kui robot tuvastab, et mõni osa on üle väljaku ääre, siis robot taganeb ja pöörab ümber. Kui vastast ei ole ühegi anduri nägemisalas, sõidetakse üsna etteaimamatult mööda väljakut, otsitakse vastast.

Programmikoodi tähtsamad osad

Roomba™ kontrolleri programmi tegemisel on aluseks võetud Robotiklubi loodud C++ teek AVRCppLib. Meie programm koosneb väljakutsetest AVRCppLib käskude poole, mis tegeleb madalama taseme käskude andmisega AVR mikrokontrollerile. Tänu antud teegi kasutamisele saime rohkem keskenduda programmi algoritmile ja lõpptulemusele, kui sellele, kuidas täpselt tuleb toimida, et lugeda analoog-digitaalmuunduri väärtuseid või kuidas pidada sidet meie kontrollerplaadi ning Roomba™ enda kontrolleri vahel.

Iga C (ja C++) programmi kohustuslik osa on *main* funktsioon. Meie *main* funktsioon asub omaette failis main.cpp ja näeb välja lihtne:

```
1  int main (void){
2         Initialize();
3         Program program;
4         program.Run();
5  };
```

Esmalt kutsume välja AVRCppLib initsialiseerimise funktsiooni (2), loome üksuse klassist Program (3) ja kutsume välja üksuse program meetodi Run (4).

Klassi Program meetod Run on defineeritud failis Program.cpp. Klassi deklaratsioon on failis Program.h. Tüüpiliselt C ja C++ koodi puhul ongi nii, et klasside ja funktsioonide deklaratsioonid on .h (*header* ehk päis) failis ja definitsioonid (ehk kood ise) .c või .cpp failis.

Program::Run() on kõige tähtsam meetod, mis teeb ära kogu töö anduritelt info lugemiseks, selle põhjal otsuste tegemiseks, ning käskude saatmiseks Roomba enda kontrollerile. Funktsiooni alguses on muutujate deklaratsioonid ja nende algväärtustamine.

```
1  void Program::Run(){
2         roomba.Init();
3         clock.Delay(5000);
```

Luuakse ühendus Roomba kontrolleri ja meie kontrolleri vahel (2). Oodatakse 5000 millisekundit ehk 5 sek enne liikuma hakkamist (3). Nii on ette nähtud Sumo-Roomba reeglites.

Järgneb lõpmatu *while* tsükkel, mis on omane paljudele lihtsamatele mikrokontrolleril baseeruvatele süsteemidele. Kui ei kasutata op süsteemi (nagu meil), siis ei tohi programm kunagi väljuda, vastasel korral ei ole edasine mikrokontrolleri käitumine teada.

```
1  while (true) {
2         roomba.ReadSensors(SUBSET_ALL);
3         FrontLED.Toggle();
```

Tsükli algul küsitakse roomba kontrolleri käest tema andurite info (2). Vilgutatakse valgusdioodi (3), mis on hea indikaator, et programm ei ole hangunud.

Seejärel vaadatakse *if* lausetega läbi iga anduri info ja käitutakse vastavalt andurite väärtustele. Näitena ühe anduri info uurimine ja sellele vastav tegutsemine.

```
1  if(roomba.Sensors.CliffFrontRight){
2         roomba.LED.DirtDetect(1);
3         roomba.Drive(-350, GO_STRAIGHT);
4         clock.Delay(400);
5         roomba.LED.DirtDetect(0);
6         roomba.Drive(500, 1);
7         clock.Delay(400);
8  }
```

Siin vaadatakse läbi Roombalt saabunud Sensors paketi bitt CliffFrontRight. Kui see on püsti (=„1”), tähendab, et Roomba esimene parem äär on üle väljaku ääre ja mõistlik oleks end ringi pöörata. Seda kood käsibki teha. Vaatame lähemalt. Rida (2) paneb põlema LEDi, mis natuke hiljem, real (5), kustutatakse. See on arendajatele visuaalseks infoks, et Roomba tõepoolest nägi, et on üle väljaku ääre ning reageeris sellele. Rida (3) käsib veidi tagasi sõita, seejärel tehakse ümberpöörd (6). Vahepealsed *delay* käsud on selleks, et robotil oleks aega väljaku äärest piisavalt kaugele sõitmiseks.

Sarnaselt vaadatakse läbi ülejäänud Roomba ja kontrolleri külge ühendatud andurid, ning käitutakse vastavalt. Vt. Lisa 1.

Fotod


Kokkuvõte

Kokkuvõtvalt võib öelda, et antud õppeaine ja sellega kaasnev praktiline töö tasusid end ära ning andsid korraliku lühiülevaate robootikast.

Võistlustulemus

Esimeseks me ei tulnud kuid kokkuvõttes läks üsna hästi.

Järeldused

Tolmuimejast on võimalik sumorobotit ehitada.
Robootika pole lihtsalt (arendus)töö, vaid ka hobi.
Roomba on odav robotplatvorm mitmetele projektidele.

Kasutatud kirjandus

1. AVR C++ Teegi juhend. http://www.robotiklubi.ee/juhendid/avr-cpp-lib
2. iRobot Roomba Open Interface Specifiation (ROI)
3. Kaugusandurite juhend: http://www.acroname.com/robotics/info/articles/sharp/sharp.html
4. Baltic Robot Sumo – Roomba reeglid:
http://www.balticrobotsumo.org/read_write/file/ROOMBA_Sumo(en)_final_v.1.1.pdf.

Lisa 1. Algoritm.

```
while (true)
{
    FrontLED.Toggle();
    roomba.ReadSensors(SUBSET_ALL);

    if(roomba.Sensors.CliffFrontLeft){      // Üle ääre?
       roomba.LED.DirtDetect(1);
       roomba.Drive(-350, GO_STRAIGHT);
       clock.Delay(400);
       roomba.LED.DirtDetect(0);
       roomba.Drive(500, -1);
       clock.Delay(400);
    }

    if(roomba.Sensors.CliffRight){          // Üle ääre?
       roomba.LED.DirtDetect(1);
       roomba.Drive(-350, GO_STRAIGHT);
       clock.Delay(400);
       roomba.LED.DirtDetect(0);
       roomba.Drive(500, 1);
       clock.Delay(400);
    }

    if(roomba.Sensors.CliffFrontRight){     // Üle ääre?
       roomba.LED.DirtDetect(1);
       roomba.Drive(-350, GO_STRAIGHT);
       clock.Delay(400);
       roomba.LED.DirtDetect(0);
       roomba.Drive(500, 1);
       clock.Delay(400);
    }

    if(((666-irMiddle.GetRawDistance(5))/2)<325) {    // Vastane otse ees?
        roomba.LED.Clean(1);                          //
        roomba.Drive(500, GO_STRAIGHT);               // On, sõida otse
    }
    else
    {
        if(((650-irRight.GetRawDistance(5))/2)<250) { // Vastane ees, paremal?
            roomba.LED.Max(1);
            int n=0;

            while(((666-irMiddle.GetRawDistance(5))/2)>250) // On, pööra kuni vastane on otse ees
            {
                roomba.Drive(500, -1);

                if(((666-irMiddle.GetRawDistance(5))/2)<250) {
                    roomba.LED.Clean(1);
                    roomba.LED.Max(0);
                    roomba.Drive(500, GO_STRAIGHT);
                    break;
                }

                n++;
                if (n>=1000)
                   break;
            }
            roomba.LED.Max(0);
            continue;
        }

        if(((650-irLeft.GetRawDistance(5))/2)<250) {  // Vastane ees, vasakul?
            roomba.LED.Spot(1);
            int n=0;

            while(((666-irMiddle.GetRawDistance(5))/2)>250) // On, pööra kuni vastane on otse ees.
            {
                roomba.Drive(500, 1);

                if(((666-irMiddle.GetRawDistance(5))/2)<250) {
                    roomba.LED.Clean(1);
```

```
                    roomba.LED.Spot(0);
                    roomba.Drive(500, GO_STRAIGHT);
                    break;
                }

            n++;
            if (n>=1000)
                break;
            }
            roomba.LED.Spot(0);
            continue;
        }

        if(DigiIrBackLeft.IsSet()) { // Vastane taga vasakul?
            int n=0;
            roomba.LED.Spot(1);
            while(((666-irMiddle.GetRawDistance(5))/2)>250) // On, pööra kuni vastane on otse ees
            {
                roomba.Drive(500, GO_RIGHT);
                if(((666-irMiddle.GetRawDistance(5))/2)<250) {
                    roomba.LED.Clean(1);
                    roomba.LED.Spot(0);
                    roomba.Drive(500, GO_STRAIGHT);
                    break;
                }

            n++;
            if (n>=1000)
                break;
            }
            roomba.LED.Spot(0);
            continue;
        }

        if(((650-irBackRight.GetRawDistance(5))/2)<250) { // Vastane taga paremal?
            int n=0;
            roomba.LED.Spot(1);
            while(((666-irMiddle.GetRawDistance(5))/2)>250) // On, pööra kuni vastane otse ees
            {
                roomba.Drive(500, GO_LEFT);

                if(((666-irMiddle.GetRawDistance(5))/2)<250) {
                    roomba.LED.Clean(1);
                    roomba.LED.Spot(0);
                    roomba.Drive(500, GO_STRAIGHT);
                    break;
                }

            n++;
            if (n>=1000)
            break;
            }
            roomba.LED.Spot(0);
            continue;
        }
    }

    if(roomba.Sensors.Clean) {                // Roomba ja programm seisma
        roomba.Drive(0, GO_STRAIGHT);
        roomba.PowerOff();
        while(true);
    }
} //end while
```