TTÜ Roboriklubi

Teeme ise 2013

Micromouse

Siim Kristjan Pariis
Marten Pärg
Erik Albert

Tallinn 2013

Eesmärk

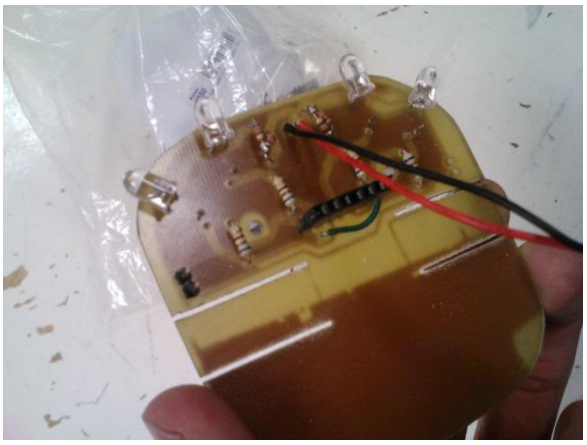
Projekti eesmärgiks oli ehitada laburünti läbiva roboti, mis vastaks micromouse reeglitele. Kuna meie rühmast keegi ei ole varem robotiehitusega kokku puutunud, siis peamine eesmärk oli kursusest saada praktilisi kogemusi.

Kokkuvõte reeglitest:

- Roboti tingimused
  - Maksimaalsed mõõtmed 160 x 160mm
  - Täielikult autonoomne
  - Ei või endast maha jätta mitte midagi
  - Ei tohi liikuda üle seinte
- Väljaku tingimused
  - mõõtmed 160 x 160mm ruutudest koosnev labürint
  - Seina kõrgus 50mmm
  - Seina paksus 12mm
  - Seina värv valge, põrand must (või tume), sein pealt punane. (Värvimisel võib esineda defekte liitekohtades)
  - Start on ühest labürindi nurgast
  - Lõpp labürindi keskel ja koosneb neljast ruudust, millel on vaid üks sissekäik

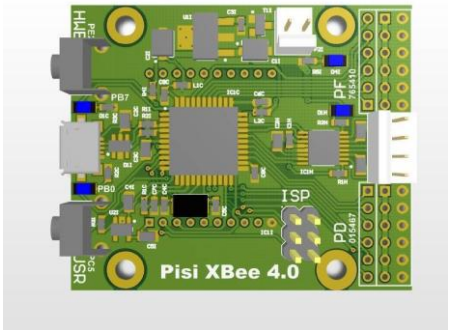
Roboti disain

Roboti ehitamine algas paberile roboti kere üldiste mõõtmete peale kandmisega . Algne soov oli kogu roboti kereks jätta trükkplaat, mille peale kõik komponendid mahutada ning külgedele rattad paigutada. Sealjuures üritasime roboti suuruse võimalikult väikeseks jätta. Roboti kuju võimalikult lihtsaks jättes leppisime kokku ruudu kujulise disainiga, mille nurgad me hiljem ära ümardasime.
Elektrimootorid koos ratastega sättisime ruudu külgedele võimalikult keskele, nii et rattad jääksid võimalikult roboti kere juurde ning ei lisaks robotile oluliselt laiust juurde. Mootorid kinnitasime trükkplaadi peale termorüüga.
LEDid ning andurid otsustasime loomulikult roboti ette panna; LEDid trükkplaadi peale ning andurid trükkplaadi alumisele poolele, et need liigse valguse käest võimalikult peidus oleksid. Kokku sai neid robotile neli paari pandud, kaks IR LEDi ette keskele, mis näitaksid otse ette, ning kaks tükki veidi eemale ääre peale, mis näitaksid 45 kraadise nurga all külgedele. Tagantjärgi mõeldes oleks palju parem olnud kuue paariga, kus kaks tükki oleks näitanud otse külgedele ning lisaks veel poleks pidanud LEDid nii ääre peale panema.



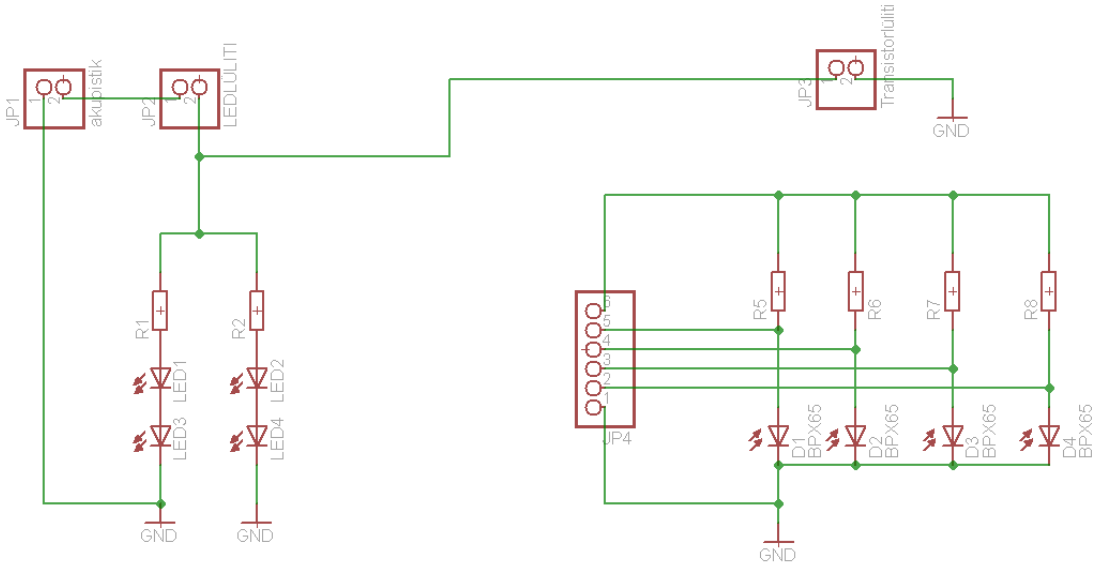
Aku otsustasime panna roboti tagumise poole peale, sest soovisime raskuskeskme jätta tahapoole ning see oli meie komponentidest kõige raskem . Et roboti taguosa aga vastu maad ei vajuks ning LEDid liiga ülesse ei vahiks lisasime aku kohale trükkplaadi alla 9.5mm plastist tugikuuli.

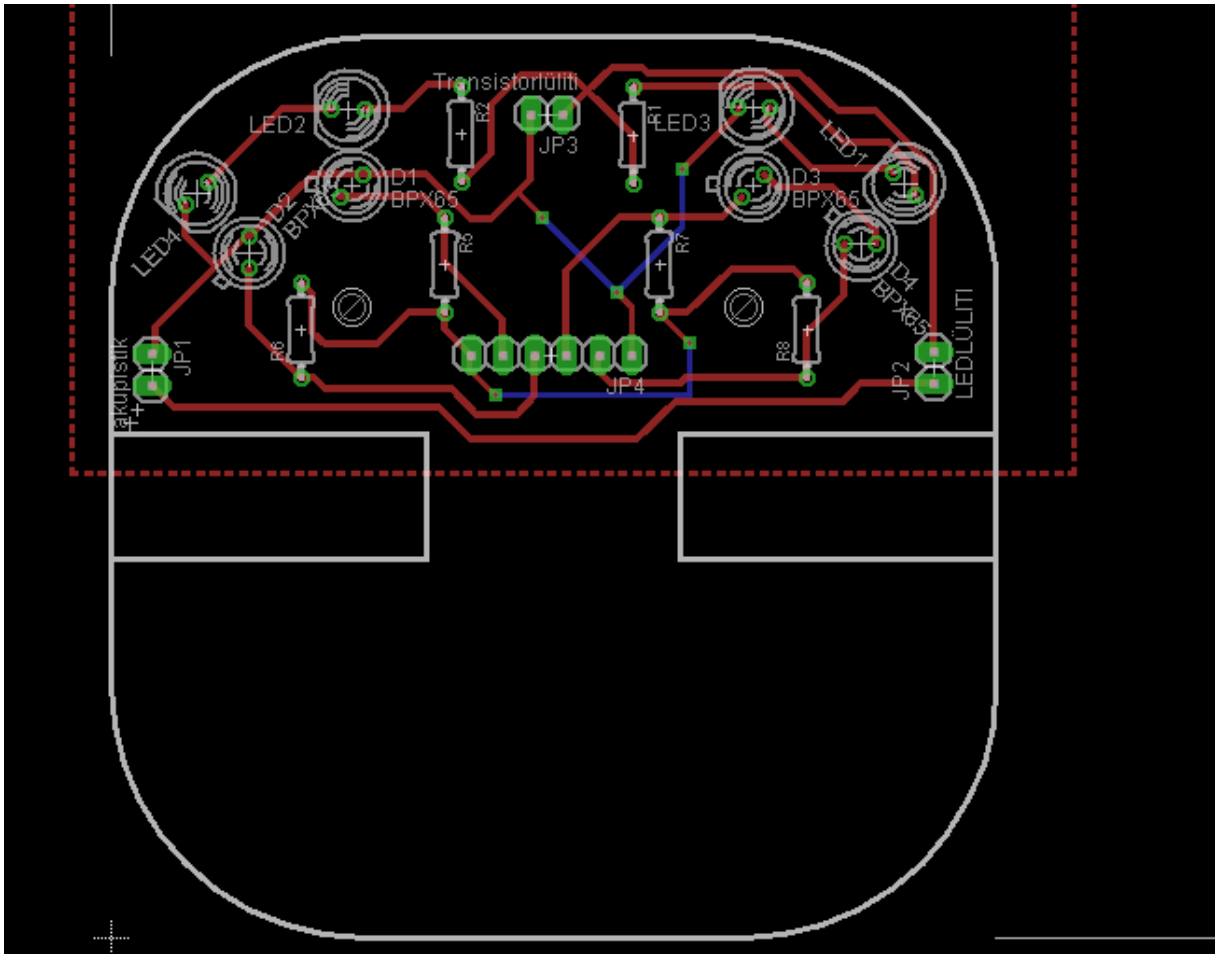
Elektroonika poole pealt anti meile Pisi-Xbee4 elektroonikaplaat ATMega32u4 mikrokontrolleriga. Sest roboti mõõtmed olime üritanud väikseks jätta ning plaat ei mahtunud kuhugile hästi ära ning lisaks sellele oli ka küllaltki kerge, paigutasime selle roboti keskele mootorite kohale pukkide peale.



Elektriskeem

Elektriskeemi koostamiseks kasutasime EagleCad nimelist tarkvara. Kuna varasemad kogemused meil puudusid, siis võttis selle koostamine küllalt kaua aega ning pidime enamasti lootma Robotiklubi liikmete abile.





Kood

Ka koodi kirjutamises olime me kõik alles „rohelised“ niiet alguses meil polnud õrna aimugi mida või kuidas asja teha, kuid aja jooksul hakkasime aru saama koodi struktuuri olemusest ning hakkasime looma seoseid mis parameetrid roboti käitumist mõjutavad.
Tarkvarana kasutasime Atmel Studio 6.0.

Põhiline aeg kulus selleks, et robot sõidaks ning manööverdaks ühtlaselt, kuna meie kood töötaks ainult ideaalsetes tingimustes(st. robot peab olema alati raja keskel). Kuid see ei tahtnud meil õnnestuda, sest hiljem saime aru et aku pinge mõjutab roboti käitumist väga oluliselt ning meil puudus selleks pingeregulaator.

Seinte vahel suutis robot hoida üpris hästi kurssi. Idee seisnes selles, et kood luges mõlema diagonaalis asuva IR Ledi näitu ning lahutas ühe teisest. Kui robot asetseb ühele seinale ligemal, siis näitab vastav IR Led ka suuremat näitu. Seega kui vastus tuli negatiivne või positiivne, reguleeris kood vastavat mootori kiirust suuremaks ning niimoodi hoidis ta trajektoori täpselt seinte vahel.

```
a = line[1]-line[3];
    if(a<0){
      Motors_Speed(200+10*a,200);
    }
    else
      Motors_Speed(200,200);

    if(a>0){
      Motors_Speed(200,200-10*a);
    }
    else
      Motors_Speed(200,200);


    Motors_Speed(200,200);
```

Küll aga tekkisid probleemid ristmikel ning tupikutes.
Kui eesmised andurid tuvastasid seina otse ees, jäi robot seisma, keeras otsa ümber ning jätkas teekonda. Kood oli meil kirjutatud selliselt, et antud olukorras mootorid töötavad üksteise suhtes vastupidises suunas teatud aja jooksul kuni teoreetiliselt peaks ta olema ümber pööranud. Kuna igakord ei olnud aku pinge võrdne, keeras ta vahest rohkem, vahest vähem. See aga põhjustas järgmise probleemi. Kui robot jäi seinte suhtes liiga viltu, ei ulatanud üks diagonaalsetest anduritest vastavat seina enam lugeda ning arvas, et tegu on ristmikuga. Ristmikut loeb ta nii, et kui vastav diagonaalandur tuvastab, et sein kadus vaateväljast ära, saab ta aru, et tulemas on ristmik. Siis sõidab ta väikese viivitusega veel edasi niikaua, kui teoreetiliselt peaks ta olema ristmiku keskel. Siis teeb ta vastamise pööramise ning jätkab sõitu.
Tupiku olukordades nägi probleem välja järgmine:
1)pärast tupikus ümber keeramist jäi robot viltu kuna aku pinge oli kas suurem või väiksem kui eelmisel korral.

2)üks andur luges, et sein on kaugel ning kood arvas et tegu on siis ristmikuga kuna üks sein jäi kaugele
3)robot käitus vastavalt ristmiku koodile, sõitis teatud delayga otse, mis tähendab antud olukorras vastu seina sõitmist, ning siis keeras 90 paremale või vasakule
4)robot sattus segadusse kuna ta asend ei olnud enam paraleelne seintega, ning hakkas lõpmatuseni ristmike alamkoode kordama.

Antud probleemi üritasime mitu nädalat lahendada erinevate improviseerimiste abil kuid päris toimiva lahenduseni me ei jõudnudki. Jõudsime järelduseni, et oleksime võinud ka külgedele andurid panna ning neid kasutada ristmike jaoks.

```
 if((line[4]<940) && (line[2]<940))
    {   //ümber pööramine kui sein vastu tuleb
       Motors_Speed(0,0);
       _delay_ms(150);
       Motors_Speed(200,-200);
       for(v=0;v < n; v = v+1) _delay_ms(1);
       v=0;
       Motors_Speed(0,0);
       _delay_ms(500);
       Motors_Speed(200,-200);
       for(v=0;v < n; v = v+1) _delay_ms(1);
       v=0;
       Motors_Speed(0,0);
       _delay_ms(1000);
if(line[1]>pa+20)  //ristmik-paremale
    {


       Motors_Speed(0,0);
       _delay_ms(150);
       Motors_Speed(200,-200);
       v=0;
       for(v=0;v < n; v = v+1) _delay_ms(1);
       v=0;
       Motors_Speed(0,0);
       _delay_ms(150);
       v=0;
       while (v<500)
       {
          v=v+1;
          a = line[1]-pa;
          if(a<0){
             Motors_Speed(200+10*a,200);
             _delay_ms(1);
             PORTB |= (1<<PB0);
          }
```

```
      if(a>0){
         Motors_Speed(200,200-10*a);
         _delay_ms(1);
         PORTB |= (1<<PB7);
      }


   }
   v=0;
}


if(line[3]>1012){    //ristmik-vasakule
//    if(p==1){
   Motors_Speed(200,200);
   _delay_ms(600);
   Motors_Speed(0,0);
   _delay_ms(150);
   Motors_Speed(-200,200);
   for(v=0;v < n; v = v+1) _delay_ms(1);
   Motors_Speed(0,0);
   _delay_ms(150);
   //Motors_Speed(200,200);
   //_delay_ms(150);
   while (v<500)
   {
      v=v+1;
      a = line[1]-line[3];
      if(a<0){
         Motors_Speed(200+10*a,200);
         _delay_ms(1);
      }
      else Motors_Speed(200,200);
      if(a>0){
         Motors_Speed(200,200-10*a);
         _delay_ms(1);
      }
      else Motors_Speed(200,200);

   }
   v=0;
//    }
}
```

Kokkuvõte

Olgugi, et meie robot jäi poolikuks ning ebatäiuslikuks, oli meil peamine eesmärk saada praktilist ettekujutust elektroonikaskeemidest ning programmeerimisest. See eesmärk sai ka täidetud.
Kui peaksime sarnast robotit uuesti ehitama, teaksime järgmine kord arvestada järgmiste asjadega:

1) Andurite paigutus tuleb teha paremini. Andurid viia sissepoole, et seinte vastu sõitmine neid ei kahjustaks ning panna ka mõned andurid külgedele.

2) Kompaktne disain tuleb kasuks, mida me ka seekord rakendasime.

3) Kasutada pingeregulaatorit