Robotiehitamise juhend
Ultraheli sonar

Autorid: Alar Ainla
Alvo Aabloo

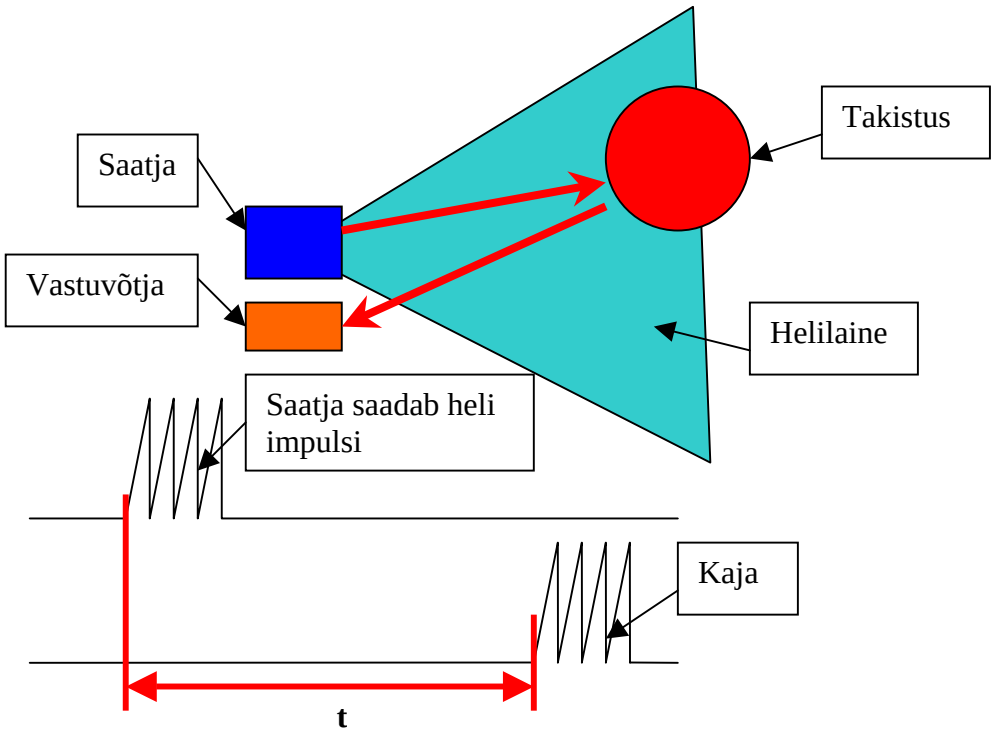
Juhendi koostamist on toetanud EITSA

Tartu 2003

SISUKORD

Sonari põhimõte

Sonar on andur, millega saab mõõta ees olevate takistuste kaugust. Kauguse mõõtmiseks kiirgab sonar välja heli impulsi, mis levib ja peegeldub takistuselt – kaja. Siis tuleb ära mõõta kaua aega selleks kulub, et kaja jõuaks sonarini tagasi, teades heli kiirust (ligikaudu **343m/s**, sest see sõltub temperatuurist jm. muutuvatest atmosfääri parameetritest) saab leida objekti kauguse. Sonarid erinevad oma mõõtmisulatuselt ja sellelt, kuidas neid ühendatakse mikrokontrolleriga.



Välja kiiratav heli ei ole kitsas kiir, vaid hõlmab koonuse kujulise ruumi osa kogu laiusega **umbes 70°**. Selles mõttes ei ole sonar kaamera, mis annaks infot selle kohta, kus objekt täpselt on, vaid ainult, selle et lähim ees olev takistus asub nii kaugel.

Sonar Devantech SRF08

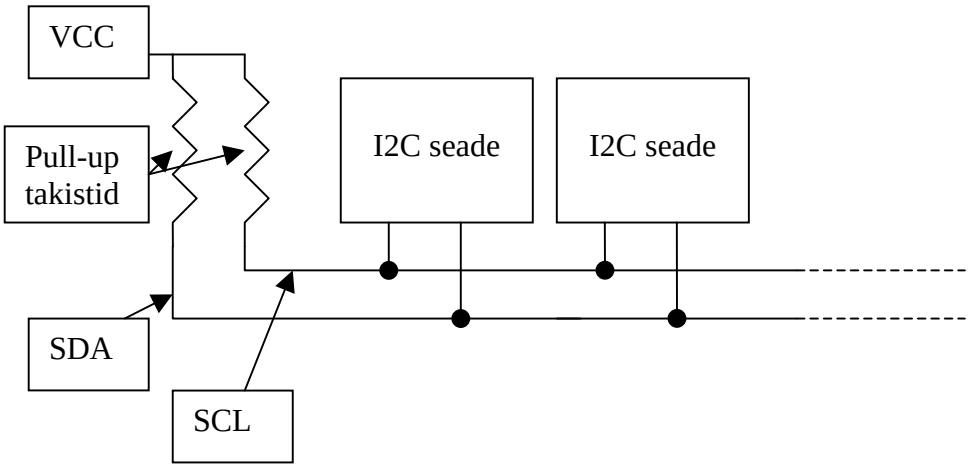
Vaatleme lähemalt sonarit, mis on kasutamiseks Robotexil.
Kui enamus lihtsamaid sonareid töötab nii, et mikroprotsessor saadab sonarile impulsi ja ootab kuni saabub kaja ning peab ära mõõtma selleks kuluva aja, siis selle sonaril on juba endal mikrokontroller (PIC), mis suudab täielikult sooritada mõõtmise. Käskude andmine ja tulemuste lugemine toimub sellel sonaril **I2C protokolliga**, mis on lihtne üldtuntud jadaside protokoll.

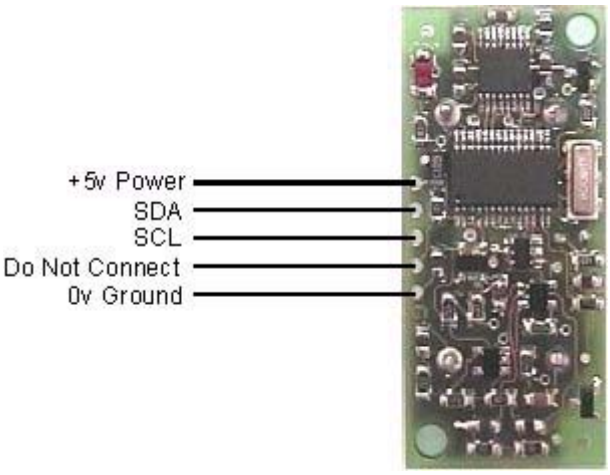
I2C protokoll

I2C on lihtne sünkroonne jadaside protokoll, see tähendab, et alati üks pool juhib (**master**) ja teine allub käskudele (**slave**). I2C on andmete edastamiseks kaks juhet, mis moodustavad I2C siini, millel võib olla rohkem kui 1 slave seade.

- **Taktsignaali juhe (SCL)**
- **Andmete juhe (SDA)**

Juhtmed ühendatakse läbi takisti (**10kΩ**) toitepingega (**+5V**) – **pull-up takisti**

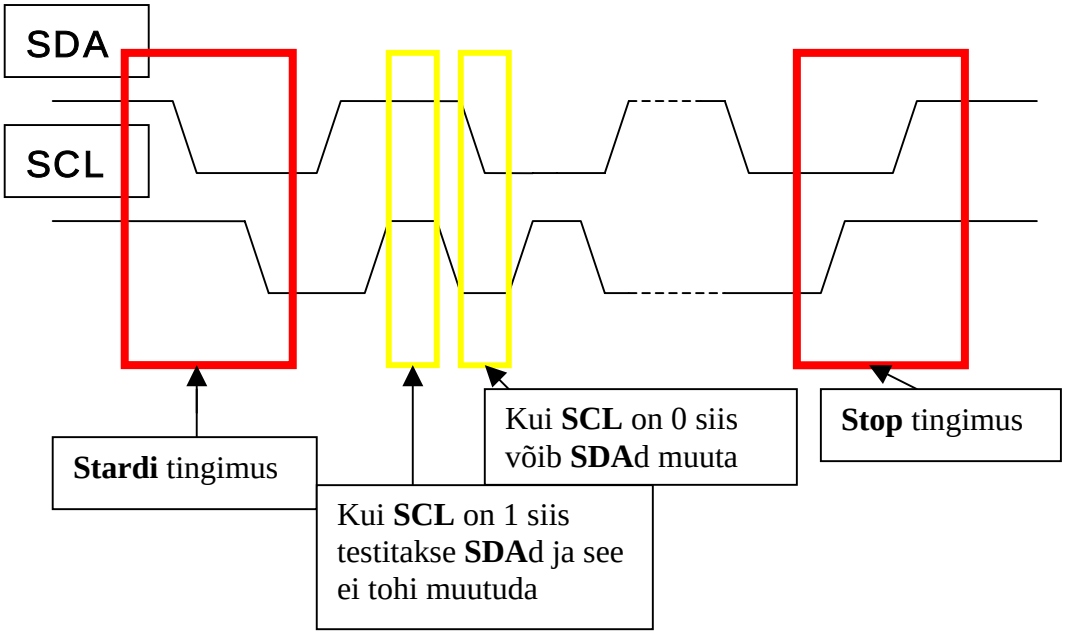




I2C andmeside:

- Kõigepealt saadab I2C master siinile side stardi tingimus
- Seejärel saadab master seadme poole pöördumise käsu, mis sisaldab seadme addressi (milline seade valitakse) ning seda kas kirjutatkse või loetakse (R/W)
- Kui tegu on kirjutamis käsuga:
  - Saadetakse seadmele registri addressi, mille järgi seatakse sisemine addressi register slave seadmes.
  - Saadab andmebaite, mis kirjutatakse registritesse. Iga baidiga suurendatakse sisemist registri addressi.
- Kui tegu on lugemis käsuga:
  - Hakkab lugema slave seadmest baite, alates viimati seatud addressist.
- Kasutus lõppeb pärast stop tingimust

Lisaks on vahepeal pärast igat baiti **ACK**nowledge (millega master küsib slave käest kas see on valmis jätkama), pärast viimast baiti ei ole **ACK**d.



Kuidas I2C siini juhtida?
1 saatmiseks pannakse jalg kõrgoomilisse olekusse (sisendiks) ja siis **pull-up** takisti tõmbab siini 1ks. Nulli saatmiseks pannakse jalg olekusse 0, ehk tõmmatakse siin maha. **I2C seade ei tohi siini ise olekusse 1 panna**.

I2C Algoritmid

Madala taseme I2C algoritmid (driver)

I2C andmeside alustamine

```
void i2c_start(void)
{
        SDA_0ks();
        oota(); //Ootuse pikkus peaks olema 50µs, SFR08 jaoks
        SCL_0ks();
        oota();
}
```

Baidi saatmine siinile

```
//Saadab andmed ning tagastab väärduse, mis väljendab tulemust
//..kas õnnestus saata, seade on hõivatud või ei vasta
char i2c_kirjuta(char andmed)
{
   unsigned char biti_loendur=0;

        for(biti_loendur=0; biti_loendur<=7; biti_loendur++)
         //Saada kõik 8 bitti järjest välja
          {
              if( andmed & 0x80 ) //Testib kõrgemat andme bitti
                      { SDA_1ks(); } //Vastavalt biti testi tulemusele sea kas SDA 1ks
                  else { SDA_0ks(); } //või 0ks, ehk kõrgem andmebit saadetakse
              SCL_1ks(); //Sea SCL 1ks, algab testimine
              oota(); //Oota, kuni vastuvõtjad bitti testivad
              SCL_0ks(); //Sea SCL 0ks tagasi
              oota(); //Oota kuni vastuvõtja “märkab”, et SCL on tagasi 0
              andmed = <<1; //Nihuta andme bitte … järgmine kord on ühe võrra madalam
                           //bit seega see, mida saadetakse
          }
        //ACK, tehakse pärast andme bittide saatmist
```

```
        SCL_1ks(); SDA_1(); //Lase siin vabaks, ehk 1 deks
        oota(); //oota kuni slave seade reageerib

        if(SDA==0)
         {
             SCL_0ks(); //Kui SDA on 0 siis slave tegi ACK ja võib lõpetada
             oota();
         }
        else{
                pikk_viide(); //Oota veel
                if(SDA==0)
                 {
                    SCL_0ks(); //Kui SDA läks lõpuks 0ks siis on OK
                    oota();
                 }
                //Kui seade ikka ei reageeri, siis seade ei vasta
                else { return DEVICE_NOT_RESPONDING; }
            }
            //Pärast SCL 0ks seadmist, peab ka SDA 1ks minema, kui ei lähe on seade
            //hõivatud
        if(SDA==0)
         {
               pikk_viide(); //PIKK VIIDE
               if(SDA==0) { return DEVICE_BUSY; }
         }
return I2C_OK; //Kui bait õnnestus edukalt saata
}
```

Baidi lugemine siinilt

```
//Kui ACK on 0 siis ei tehta lugemise lõpus ACKd, kui 1 siis tehakse
//ACKd ei tehta viimasase baidi lugemise järel.
char i2c_loe(char ack)
{
  unsigned char biti_loendur=0, andmed=0;

        SDA_1ks();
        for(biti_loendur=0; biti_loendur<=7; biti_loendur++)
          {
              SCL_1ks(); //Sea SCL 1ks
              oota();  //Oota
              andmed=<<1; //Nihuta andmeid baidis
              if(SDA==1) { andmed|=0x01; } //Kui jalg on olekus 1 siis sea madalaim bit
              SCL_0ks(); //Sea SCL 0ks tagasi
              oota();
          }
        //Kui a=0 siis viimane bait, muidu tee ACK
        if(ack!=0) { SDA_0ks(); } //Ei tee ACK
            else { SDA_1ks(); } //Teeb ACK
        SCL_1ks();
        oota();
        SCL_0ks();
        oota();
return andmed; //Tagasta tulemus
}
```

I2C andmeside lõpetamine

```
void i2c_stop(void)
{
        SDA_0ks();      //SDA 0ks
        SCL_1ks();      //SCL 1ks
        oota();         //Ootab
        SDA_1ks();      //SDA 1ks, mis on lõpu sümbol
        oota();
}
```

I2C kõrgema taseme algoritmid (driver)

Kirjutamine registrisse

```
void kirjuta_registrisse(char seade, char address, char andmed)
{
        i2c_start(); //Alusta sidet
        i2c_kirjuta(seade); //Saada seadme address
        i2c_kirjuta(address); //Saada registri address valitud seadmes
        i2c_kirjuta(andmed); //Saada andmed
```

```
        i2c_stop(); //Lõpeta side
}
```

Lugemine registrist

```
char loe_registrist(char seade, char address)
{
        char a;
        i2c_start(); //Alusta sidet
        i2c_kirjuta(seade); //Saada seadme address
        i2c_kirjuta(address); //Sea registri address

        i2c_start(); //Saada uus start
        i2c_kirjuta(seade|0x01);
                //Saada seadme address, esimene bit 1 tähendab, et seejärel loetakse
        a=i2c_loe(); //Loe bait
        i2c_stop();
        return a; //Saada tulemused
}
```

Mõõtmine

Mõõtmise alustamiseks tuleb saata käsk, mis kirjutatakse **command registrisse** addressiga **0x00**. Käsust sõltub kas tulemus on **tollides** (kirjutatakse **0x50**) **sentimeetrites** (kirjutatakse **0x51**) või **mikrosekundites** (kirjutatakse **0x52**). Iga I2C käsu juurde kuulub ka seadme address, mille poole pöördutakse sellel sonaril on see tüüpiliselt (default) **0xE0**, kuid seda võib muuta (16 võimalust) juhuks kui panna samale siinile mitu sonarit. Kuna selles ülesandes pole väga paljude sonarite kasutamine vajalik, siis neid detaile siin ei käsitleta. Samuti on võimalik mõõtmise kiirendamiseks muuta mõõtmise kestust (tüüpiliselt on see **~65µs**), kui seda aga vähendada, siis kahaneb maksimaalne mõõteulatus (seda võimalust selles juhendis ei käsitleta). Seda, kas mõõtmine on valmis, saab kontrollida seadme poole pöördudes, kui mõõtmine kestab, siis seade ei võta vastu pöördumist (!=I2C_OK). Kui mõõtmine on valmis, siis võib tulemuse välja lugeda: **kõrgem bait addressilt 0x02** ja **madalam addressilt 0x03**. Vt näidis funktsiooni

```
int ping(void) //Soorita mõõtmine
{
        int tulemus=0;
        kirjuta_registrisse(0xE0, 0x00, 0x51);
        //0xE0 – seadme default address
        //Kirjutamine 0x00 registrisse - Command registrisse
        //0x51 – käsk alustab mõõtmist, kus tulemus cm'des

        //Testi kas mõõtmine valmis
        do{
                i2c_start();
                a=i2c_kirjuta(0xE0);
                i2c_stop();
          }while(a != I2C_OK); //Kuni I2C siin vabaneb

        //Siis loe mõõdetud tulemus,
        //Mis on 2 baidine suurus
        tulemus=loe_registrist(0xE0, 0x02); //0x02 tulemuse kõrgem bait
        tulemus<<=8; //Nihuta 8 bitti
        tulemus|=loe_registrist(0xE0, 0x03); //0x03 tulemuse madalam bait
        return tulemus; //tagasta tulemus
}
```

Sonari spetsifikatsioon ja lisa info

Tööpinge: **5V**
Voolutarve: **3-15mA**. Hetkeliselt kuni **275mA**
Ultraheli sagedus: **40kHz**
Mõõtmisvahemik: **3cm-6m**

SFR08 sisaldab lisaks sonarile ka ümbritseva valgustatuse andurit.

Lisaks esimesele tagasi jõudvale impulsile registreeritakse kokku kuni 17 impulsi, mis võivad saabuda erinevatelt objektidelt, need lugemid salvestatakse vastavalt järgmistesse registri paaridesse (4, 5) jne… kui mõni neist sisaldab väärtust 0, siis sai mõõte aeg otsa ilma, et kaja oleks selle aja sees tagasi jõudnud ja ka kõik järgnevad registrid on tühjad (see kehtib ka esimese registrite paari kohta, kui väärdus neis on 0 siis ei jõudnud midagi tagasi). Lisaks tüüpilisele sonar funktsioonile on seal ka ANNi (Artificial Neural Network) toetav tulemuste esitamise mood.

NB: See juhend ei käsitle sonari kõiki funktsioone, vaid ainult peamisi, mis vajalikud antud ülesandes. Täpsemaks tutvumiseks sonariga vaata ka inglise keelseid andmelehti!!!

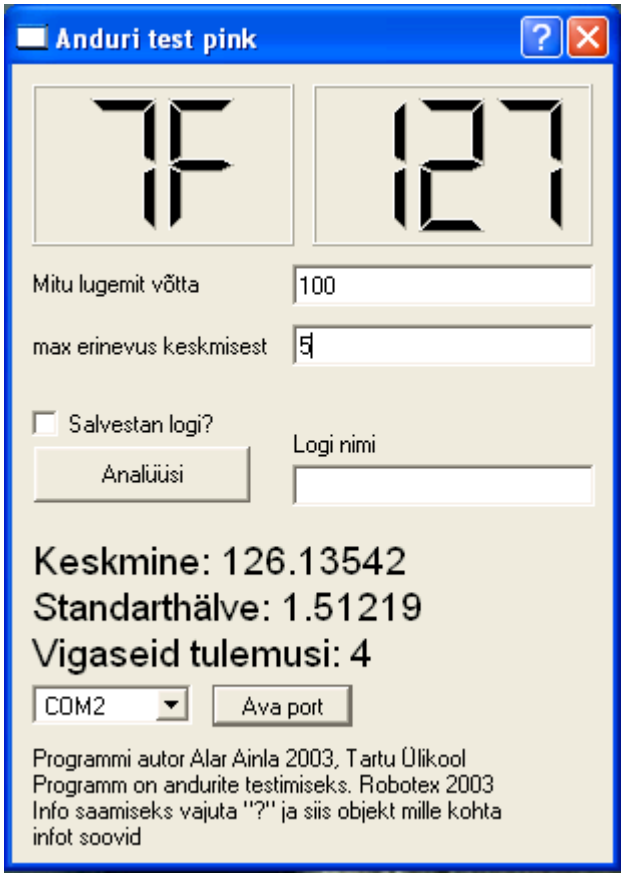
Ultraheli sonari SFR08 testimine ja kallibreerimine

- Test skeem, **PIC16F870** mikrokontrolleriga
- **PIC** (www.microchip.com tarkvara C keeles (**CC5X** http://www.bknd.com/cc5x/index.shtml)
- Windows® tarkvara, testimis tulemuste vaatamiseks/töötlemiseks

PICi tarkvara on lisas, kui ei soovi teha muutusi – võib kasutada juba kompileeritud **hex** faili. PICi kirjutamiseks kasutage **configuratsion** bitte Oscillator: **XT**, kõik järgenavad valikud olgu kas **Off** või **Disabled**. Device: **PIC16F870**. Programeerimiseks valige: **File->Import->Import to Memory…**, siis valige **hex** fail ja laadige see. Seejärel valige **PICSTART Plus->Enable Programmer**, seadge **Device** ja **Configuration Bitis**, ning programeerige seade.

1. Windowsi tarkvara, pakkige lahti zip fail, käivitaga .exe.
2. Valige port, millesse on ühendatud test seade.
3. Avage port

Näete hetke väärtust, HEX ja DEC kujul.
Saate teha statistikat - märgi ära, mitu lugemit koguda ja kui suurt kõikumist keskmisest lubada. Seejärel vajuta **Analüüsi**. **Logi** võimaldab salvestada lugemeid faili **logi.txt**



Tulemused:
Keskmine, aritmeetiline keskmine
Standarthälve, lugemite standart hälve
Vigased tulemused: nii mitu tulemust visati enne statistilist analüüsi