

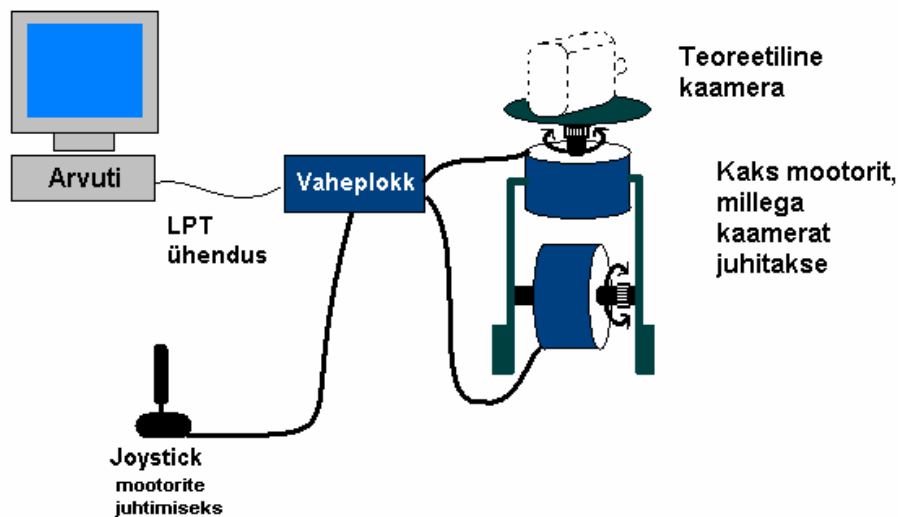
# Kahe sammumootori juhtimine

Laas Toom  
Henri Valdmann

Meie süsteem on välja töötatud nii, et terve LTP värati kaudu saab juhtida ainult kahte mootorit. (Kui kasutada ära ka *CONTROL*-värati, siis võib saada tööle ka kolmanda mootori.) See tähendab, et arvuti programm sooritab vajalikud loogilised arvutused, et selgitada välja, millised pinged tuleb mootori viikudele kirjutada. (Muidugi võiks koostada vastava kivi, millele edastataks siis vaid suund ja sammu takt, aga sellise kivi koostamine ei mahtunud meie töö raamesse).

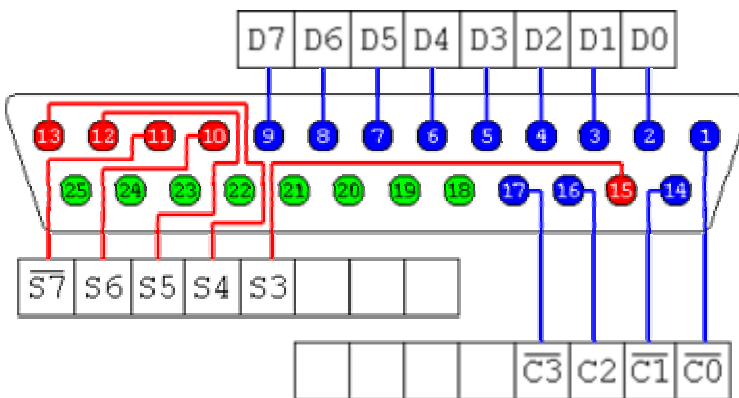
Kahe sammumootori juhtimiseks on vaja koostada 8-st võimendi-plokist koosnev süsteem, mis võtab arvuti LTP pordist signaali ning võimendab selle üles. Ühe sellise skeem on toodud allpool. Põhimõtteline skeem siis oleks järgmine:

## SKEEM



Arvuti LPT värati külge ühendatakse vaheplokki, mille kaudu toimub kogu edasine suhtlus juhtseadme ja mootoriga. Sellest plokist siis väljuvad ühendused juhtseadmesse, milleks meil on kasutusel lihtne kontaktidel baseeruv joystick, aga sama hästi võib kasutada lihtsalt nelja lülitit, mis vajutamisel lühistavad ühe

signaaliliini maha. Selline ühendus on eriti kohane just LPT värati *STATUS*-porti, sest selle neli inverteerimata sisendit on pidevalt kõrges olekus ja aktiivseks loetakse neid maha lühistatult.



**D0-D3** on esimese mootori ühendusliinid.  
**D4-D7** on teise mootori ühendusliinid  
**S3-S6** on juhtseadme sisendliinideks, kusjuures nendele kirjutatud info loetakse välja 8-bitise sõnana, nii et meile vajalikud bitid on just nende numbritega tähistatud kohtades. Nende kätte saamiseks tuleb see sõna bittideks lahutada.

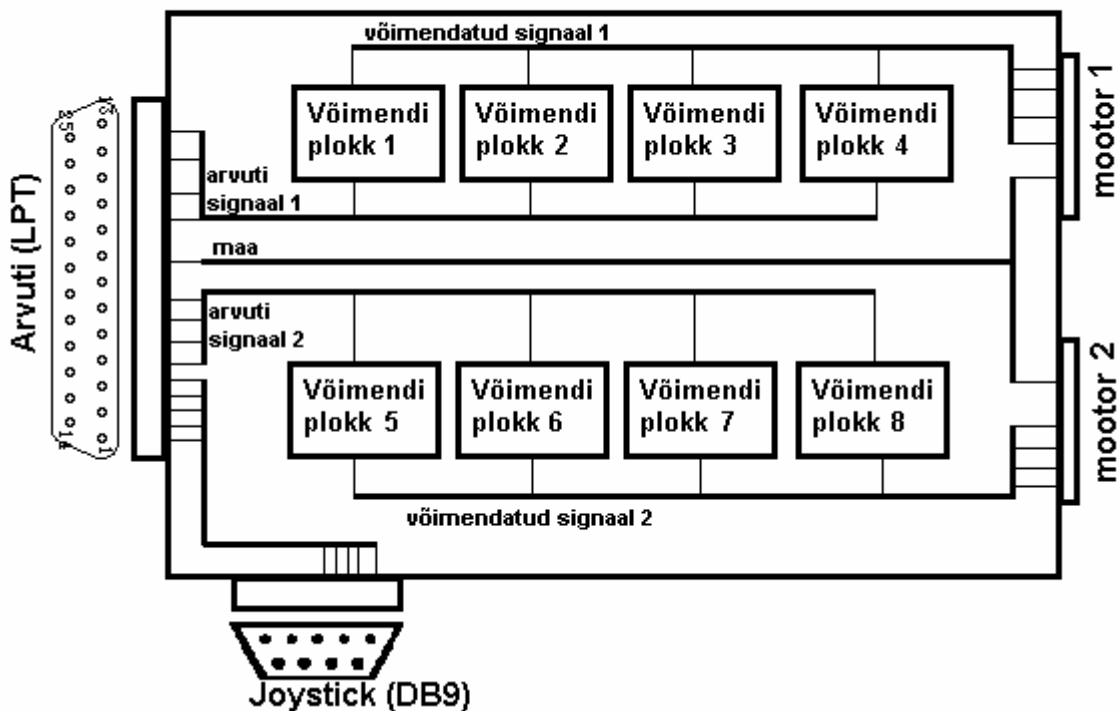
Nagu näha jääb varatil veel nii mõnigi viik kasutamata, mis tähendab, et antud ülesannet on võimalik tunduvalt edasi arendada. Näiteks kasutada ka inverteeritud **S7**-t ja **CONTROL**-bitte, et kontrollida kaamera jõudmist mingisse äärmisse seisundisse.

Siinkohal ei hakka lähemalt selgitama, kuidas on meie *joystick* implementeeritud, vaid täpsustan lihtsalt juhtseadme ühendusmeetodit. **STATUS**-vārati neli esimest bitti (**S3-S6**) on nõ inverteerimata, mis tähendab, et jõudeolekus on nendel peal +5V pinge ja aktiivne seisund on lühistatuna maha. See asjaolu lubab kasutada eriti lihtsat juhtseadet, milles mingid 4 lülitit lihtsalt lühistavad õige viigi maha.

Ja loomulikult on võimalik juhtseade ühendada üldse **gameporti**, mis võimaldaks sujuvalt kiirenevat kaamera pöörämist, sest gameporti on ühendatud ka AD muundi, mis teeb näiteks reostaatidelt saadud info digitaalseks.

## VAHEPLOKK

Arvuti ja muude seadmete vahele läheb nn vaheplokki, mille põhimõtteline skeem on järgmine:

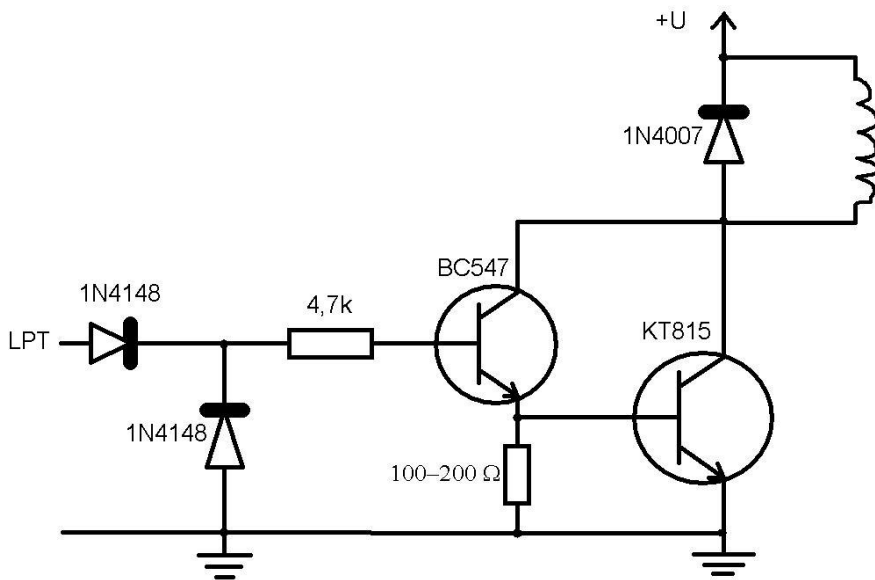


LPT-vāratiist saadav info jaguneb kolmeks:

1. Juhtseadme ühendused
2. Esimese mootori juht-viigid
3. Teise mootori juht-viigid.

Lisaks nendele juhtmetele, tuleb loomulikult igasse ühendusse lisada ka maandus, mis skeemil on juhtseadme ühenduses tähistamata, mootoriteni aga selgelt välja joonistatud.

Jooniselt on näha, et arvutist tulev signaal läbib mingi võimendusastme ja alles siis väljub mootori ühendusse. See võimendusaste kujutab endast järgmist:



Ning selle ülesanne on pakkuda mootoritele vajalikku voolu, ilma arvutit kurnamata. Võimalik ka, et üldse vajatakse toiteks +12V, mida LPT-st kätte ei saagi.

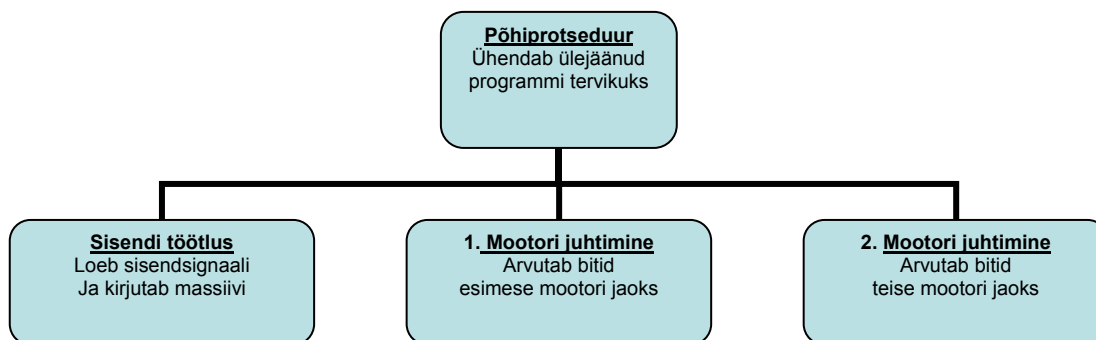
## Programm

Programm on koostatud keeles *PASCAL*, milles on väga lihtne töötav süsteem valmis saada. Hilisemad rakendused võivad aga olla (ja soovitatav ongi) kirjutada mõnes hilisemas keeles, ning näiteks lisada graafiline liides ja interneti-tugi.

Programmi tööpõhimõte:

1. loetakse sisendandmed
2. arvutatakse 1. mootori järgmise sammu info
3. arvutatakse 2. mootori järgmise sammu info
4. kirjutatakse saadud info väljundisse
5. jäädakse uut sisendit ootama

Ja ülesehitus:



Programm on võimalik väga lihtsalt (mõningate kommentaarimärkide eemaldamisega) muuta selliseks, et lüliti all hoidmisele järgneb vaid üks samm, mitte mootori sujuv pöörlemine. Üldiselt on programm aga lihtsasti mõistetav ning pole mõtet rohkem seletamisele aega kulutada.

```

Uses Dos,crt;
const outport=$378; {LPT1 väljundport}
      inport=outport+1; {LPT1 sisendport}
      viivitus=20; {viivitus, et elimineerida kontaktide värelust}
      outbits: array[0..3] of integer=(1,2,4,8);
var inbits: array[0..3] of integer;
      m1,m2,outword1,outword2:integer;

{funktsioon astendamiseks}
function pow(x,y:integer):integer;
var i,abi:integer;
begin
  abi:=x;
  for i:=2 to y do
    abi:=abi*x;
  pow:=abi;
  if (y=0) then pow:=1;
end;

{lahutab LPT status porti kirjutatud sõna bittideks}
procedure inbitid(x:integer);
var i:integer;
begin
  x:=x div 8; {jätame viimased 3 "ebahuvitavat" bitti ära}
  for i:=3 downto 0 do
    if (x>= pow(2,i)) then
      begin
        inbits[i]:=0;
        x:=x-pow(2,i);
      end
    else inbits[i]:=1; {ühendati vastav suund sisse}
  end;

{funktsioon arvu hoidmiseks vahemikus 0..3}
function liida4(a,b:integer):integer;
var abi:integer;
begin
  abi:=a+b;
  { writeln('a = ',a,' b = ',b,' abi = ',abi);}
  if (abi>3) then liida4:=0
  else
  begin
    if(abi<0) then liida4:=3
    else liida4:=abi;
  end;
end;

{arvutab bitid 1. sammumootoris kirjutamiseks}
procedure mootor1;
var abi2:integer;
begin
  if (inbits[3]=1) then {astuti paremale}
  begin
    m1:=liida4(m1,1);
    outword1:=outbits[m1];
  end
  else
  begin
    if (inbits[2]=1) then {astuti paremale}
    begin
      m1:=liida4(m1,-1);
      outword1:=outbits[m1];
    end
    else outword1:=0;
  end;
end;
end;

```

```

{arvutab bitid 2. sammumootoris kirjutamiseks}
procedure mootor2;
var abi2:integer;
begin
  if (inbits[0]=1) then {astuti yles}
  begin
    m2:=liida4(m2,1);
    outword2:=outbits[m2];
  end
  else
  begin
    if (inbits[1]=1) then {astuti alla}
    begin
      m2:=liida4(m2,-1);
      outword2:=outbits[m2];
    end
    else outword2:=0;
  end;
end;

end;

var i,x,outword:integer;
begin
  i:=0;
  m1:=0;
  repeat
{Järgnev on kontroll, et ei astuta mitu korda sama info järgi, dekommenteeri, et seda
võimaldada, hektel tehakse samme senikaua, kuni lüliti vabastatakse}
  if (x<>port[inport]) then
  begin
    clrscr;
    x:=port[inport];
    inbitid(x);
    {writeln('Input = ',x);
    for i:=0 to 3 do
    begin
      write(i+1,'(',inbits[i],') ');
    end;
    writeln;}
    mootor1;
    mootor2;
    outword:=(outword2*16)+outword1; {liidab saadud sõnad õigesti kokku}
    port[outport]:=outword; {kirjutab sõna väljundisse}
    {järgnevad 2 rida kuvaks infot mootorite hetkeseisu kohta}
    {writeln('1. mootor = ',m1);
    writeln('2. mootor = ',m2); }
    writeln('Output = ',outword); {kuvab ekraanile, mis info väljundisse
kirjutati}
    delay(viivitus); {ootab ettenähtud aja, et vältida kontaktide värelusest
tekkivat valeinfot}
    {end;} {lõpetab korduse-vältimise kontrolli}
  until keypressed; {programm töötab kuni klahvivajutuseni}
end.

```