Innhold

1 Prosjektgjennomføringen

God planlegging har vært essensielt da gruppens tre medlemmer går på forskjellige studieprogram, også deltidsprogram. Arbeid har hovedsakelig gått på kveldstid på laben og hjemme i helgene. For å være mest mulig effektive har vi delt opp oppgavene i mindre deler, og gjort disse delene hver for oss. Disse delene har så blitt sydd sammen til en helhet når vi har vært samlet.

Kommunikasjon har vært viktig og vi har sørget for å alltid ha sikkerhetskopier og revisjonskontroll av programmene vi utviklet. Vi har delt filene våre via Dropbox.

Når vi har vært samlet har fokuset vært på hovedoppgaven slik at alle skulle forstå arbeidet og prinsippene bak det.

Gruppen har fungert godt. Vi har vært sammen om de fleste programmene, men har også laget noen individuelle. Vi har jobbet ca. 85 timer hver med dette prosjektet.

Arbeidslogg ligger som vedlegg på slutten av denne rapporten.

2 Integrasjon, Derivasjon og Filtrering (Hele gruppen)

2.1 Problemstilling

Vi ønsket å dokumentere at programmene vi har laget for numerisk integrasjon, numerisk derivasjon og filtrering fungerer. For å gjøre dette så oversiktlig som mulig har vi laget tre små programmer:

- Gruppe1331_LesInnLysVector.m
- Gruppe1331_IntegreringAvLysVerdi.m
- Gruppe1331_DeriveringAvLysVerdi.m

2.2 Les inn rådata fra lyssensor

Programmet *Gruppe1331_LesInnLysVector.m* brukes til å hente inn data fra lyssensoren. Måledataene lagres i en vektor. Samtidig lagrer vi også tiden i en tidsvektor. Begge vektorene blir oppdaterte ved hver måling.

Måledataene som brukes for å demonstrere integrasjon skrives til «RaadataLysIntegrasjon.mat». Data fra denne filen blir så brukt videre til å integrere som beskrevet i kapitel 0 nedenfor.

Måledataene som brukes for å demonstrere derivering skrives til «RaadataLysDerivering.mat». Data fra denne filen blir så brukt videre til å derivere som beskrevet i kapitel 2.4 nedenfor.

Hovedgrunnen til at vi leste data inn til filer var at vi kunne bruke disse om igjen så mange ganger vi ville under utvikling og testing av programmer, uten å måtte kjøre NXT for hver gang vi skulle teste.

Utdrag fra kode for innlesing av data

```
%% index for lagring i vektorer
i=1;
Lys         = zeros(1,100);  % lagrer lyssignal

while true
    tic;  % Hent tid siden sist scan

    Lys(i) = GetLight(SENSOR_3); % Hent sensordata og lagre i vektor
    figure(1)                    % Plot data i figur #1
    subplot(2,2,1)
    plot(Lys(end-90:end));       % Plot 90 siste måleverdier
    axis([0 100 0 700])
    title('lys')

    deltaT(i) = toc;             % Ta vare på tiden.
    i=i+1;
end
% Lagre vektorene for bruk i Integrasjonsprogrammet
save ('RaadataLysIntegrasjon.mat','Lys','deltaT');
% Lagre vektorene for bruk i Derivasjonsprogrammet
% save ('RaadataLysDerivering.mat','Lys','deltaT');
```

2.3 Numerisk integrasjon (Hele gruppen)

Data for integrasjon blir hentet fra filen «RaadataLysIntegrasjon.mat». Denne filen ble laget med programmet *Gruppe1331_LesInnLysVector.m*. (Beskrevet i kapitel 2.2 ovenfor.)

Data ble lest inn etter følgende metode:

Vi benyttet skalaen på banearket som viser de forskjellige gråtonene.

Lyssensoren plasseres 5mm over arket på feltet i midten med det røde lyset på lyssensoren aktivert.

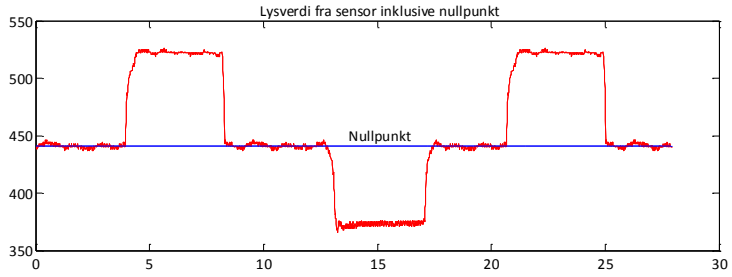
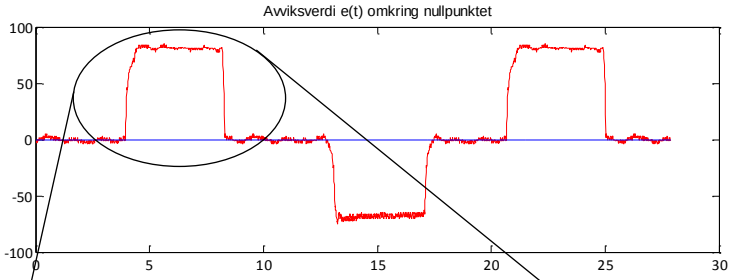
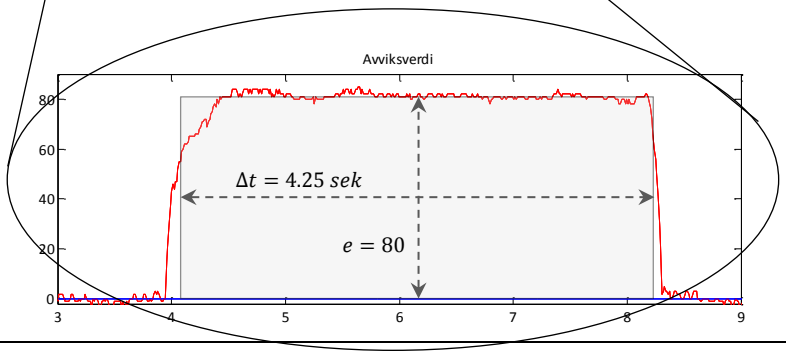
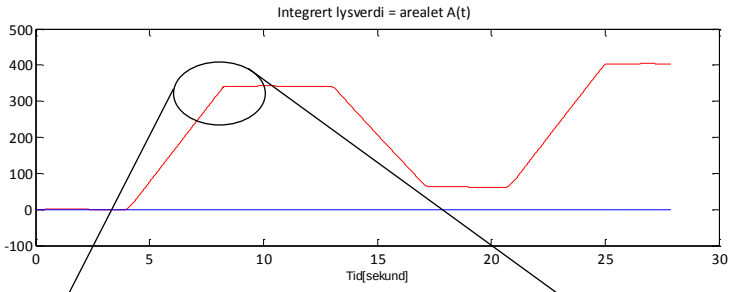
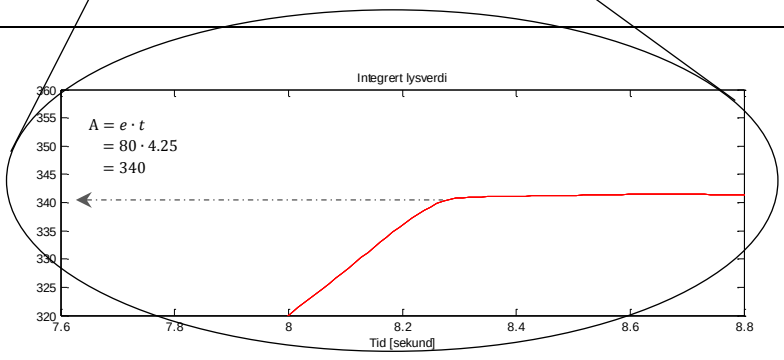
- programmet «LesInnLysVector.m» startes
- lyssensoren holdes ca. 4 sekunder på feltet i midten.
- lyssensoren flyttes to gråtoner til høyre, holdes der i ca. 4 sekunder.
- lyssensoren flyttes tilbake til midten, holdes der i ca. 4 sekunder.
- lyssensoren flyttes to gråtoner til venstre, holdes der i ca. 4 sekunder.
- lyssensoren flyttes tilbake til midten, holdes der i ca. 4 sekunder.
- lyssensoren flyttes igjen to gråtoner til høyre, holdes der i ca. 4 sekunder.
- lyssensoren flyttes tilbake til midten, holdes der i ca. 4 sekunder.
- Programmet stoppes.

For å utføre integrering brukes programmet *Gruppe1331_IntegreringAvLysVerdi.m*. Her blir rådataene fra filen «RaadataLysIntegrasjon.mat». hentet inn og bearbeidet. Selve integrasjonen foregår i koden som er ringet inn.

Utdrag av kode for Integrasjon:

```
%% Henter rådata fra tidligere samplinger
load RaadataLysIntegrasjon.mat

%% Initialiserer vektor for nullpunkt samme størrelse som andre vektorene
for i = 1:length(SampleNr)
    NullpunktOffset(i) = 441;
end
Nullpunkt = zeros(1,length(SampleNr));
%% Initialiserer scalare variable
ArealOld = 0;

%% index for lagring i vektorer
i=1;

for i = 1:length(SampleNr)
    Error(i) = LysVerdi(i) - NullpunktOffset(i);  %Finner avvik mellom
målt
                                                  %LysVerdi og  Nullpunkt
    Areal(i) = ArealOld + DeltaTid(i) * Error(i);
    ArealOld = Areal(i);                          %Oppdaterer ArealOld
end

figure(1)                          %Lager grafisk figur objekt #1
plot(Tid,LysVerdi,'r')             %Plotter Lysverdi langs tidsakse med
rød
                                   %heltrukken linje
hold on                            %Beholder figur objekt #1 for neste
plot
plot(Tid,NullpunktOffset,'b')      %Plotter nullpunktsoffset (50% grå)
langs
                                   %tidsakse med blå heltrukken linje
                                   %Legger til tittel på grafisk objekt.
Title('Lysverdi fra sensor inklusive nullpunkt');

figure(2)
plot(Tid,Error,'r')                %Plotter avvik
hold on
plot(Tid,Nullpunkt,'b')
title('Avviksverdi e(t) omkring nullpunktet');

figure(3)
plot(Tid,Error,'r')                %Plotter utsnitt avvik
hold on
plot(Tid,Nullpunkt,'b')
title('Avviksverdi e(t) omkring nullpunktet');
axis([3 9 -2 90])

figure(4)
plot(Tid,Areal,'r')                %Plotter integrert avvik
hold on
plot(Tid,Nullpunkt,'b')
title('Integrert lysverdi = arealet A(t)');

figure(5)
plot(Tid,Areal,'r')                %Plotter utsnitt av integrert avvik
hold on
plot(Tid,Nullpunkt,'b')
title('Integrert lysverdi ');
axis([7.6 8.8 320 360])
```

Nedenfor er de ulike plottene som viser at integreringen fungerer:

Figur 1: Plot av Lysverdi fra sensor sammen med nullpunkt. Nullpunkt er grått felt markert med 50.	
Figur 2: Plot som viser avviket mellom den målte lysverdien og nullpunkt.	
Figur 3: Plot som viser utsnitt av avviksplottet i figur 2. Her er verdiene som brukes til areal beregning i Figur 5 indikert i plottet.	
Figur 4: Plot som viser det integrerte avviket.	
Figur 5: Plot som viser utsnitt av det integrerte avviket i figur 4. Her ser vi at verdien av det integrerte avviket stemmer med arealet av rektangelet vist i figur 3.	

2.4 Numerisk derivasjon (Hele gruppen)

Data for derivering blir hentet fra filen «RaadataLysDerivering.mat». Denne filen ble laget med programmet *Gruppe1331_LesInnLysVector.m*. (Beskrevet i kapitel 2.2 ovenfor.)

Data ble lest inn etter følgende metode:

Vi benyttet skalaen på banearket som viser trinnløs gråtone skala.

Lyssensoren plasseres 5mm over arket på feltet i midten med det røde lyset på lyssensoren aktivert.

- Programmet «LesInnLysVector.m» startes
- Lyssensoren holdes først i ro over det lyseste området i ca. 5 sekunder.
- Lyssensoren beveges så med konstant fart mot venstre, mot det mørkere området.
- Lyssensoren holdes så i ro der over det mørkeste området i ca. 5 sekunder.
- Lyssensoren beveges så med konstant fart mot høyre, mot det lyseste området.
- Lyssensoren holdes så i ro der over det lyseste området i ca. 5 sekunder.
- Programmet stoppes

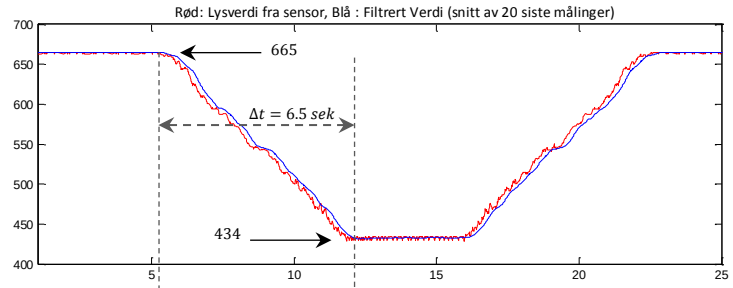
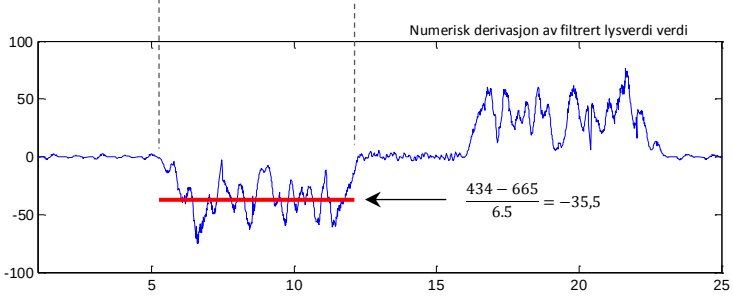
For at deriveringen skulle gi noenlunde fornuftige verdier, var vi nødt å filtrere lyssignalet før vi derivert det. Vi brukte et snitt av de siste 20 målingene hvor vi vektet hver måling med verdien 0.05.

For å utføre derivering brukes programmet *Gruppe1331_DeriveringAvLysVerdi.m*. Her blir rådataene fra filen «RaadataLysDerivering.mat». hentet inn og bearbeidet. Selve derivasjonen foregår i koden som er ringet inn.

Utdrag fra kode for Derivering:

```
%% Henter rådata fra tidligere samplinger
load RaadataLysDerivering.mat

%% index for lagring i vektorer
i=1;
LysVerdiOld = 0;
for i = 1:length(SampleNr)
%% Filter med snitt av de 20 sistemålingene. likt vektet
     if i > 20
        LysFilter(i) = Lys(i)*0.05+Lys(i-1)*0.05+Lys(i-2)*0.05+Lys(i-
        3)*0.05+
        Lys(i-4)*0.05+Lys(i-5)*0.05+Lys(i-6)*0.05+Lys(i-7)*0.05+Lys(i-
        8)*0.05+
        Lys(i-9)*0.05+Lys(i-10)*0.05+Lys(i-11)*0.05+Lys(i-12)*0.05+
        Lys(i-13)*0.05+Lys(i-14)*0.05+Lys(i-15)*0.05+Lys(i-16)*0.05+
        Lys(i-17)*0.05+Lys(i-18)*0.05+Lys(i-19)*0.05;
     else
        LysFilter(i) = Lys(i);
     end
%% Derivasjon av Lysverdi
     Derivert(i) = (LysFilter(i) - LysVerdiOld)/DeltaTid(i)
     LysVerdiOld = LysFilter(i);
end

%Plotting av lysverdi
figure(1)
plot(Tid,Lys,'r')
hold on
plot(Tid,LysFilter,'b')
title('Lysverdi rå og filtrert');
axis([1 25 400 700])

%Plotting av derivert verdi
figure(2)
plot(Tid,Derivert,'b')
title('Derivert verdi');
axis([1 25 -100 100])
```

Nedenfor er de ulike plottene som viser at derivering og filtrering fungerer:

Figur 6: Plot som viser lysverdi fra sensor og filtrert lysverdi. Her er lysverdiene hentet fra det nederste feltet på arket. Verdiene indikert i plottet brukes i figur 7.	
Figur 7: Her ser vi at den deriverte er -35,5, som er den gjennomsnittlige vekstraten mellom de stiplede linjene.	

2.5 Filtrering (Hele gruppen)

Utklipp av kode for Derivering som viser filtreringen som blir utført for å lage plottene i kapitel 2.4 ovenfor.

```
for i = 1:length(SampleNr)
%% Filter med snitt av de 20 sistemålingene. likt vektet
     if i > 20
LysFilter(i) = Lys(i)*0.05+Lys(i-1)*0.05+Lys(i-2)*0.05+Lys(i-3)*0.05+
Lys(i-4)*0.05+Lys(i-5)*0.05+Lys(i-6)*0.05+Lys(i-7)*0.05+Lys(i-8)*0.05+
Lys(i-9)*0.05+Lys(i-10)*0.05+Lys(i-11)*0.05+Lys(i-12)*0.05+
Lys(i-13)*0.05+Lys(i-14)*0.05+Lys(i-15)*0.05+Lys(i-16)*0.05+
Lys(i-17)*0.05+Lys(i-18)*0.05+Lys(i-19)*0.05;
     else
LysFilter(i) = Lys(i);
     end
```

Dette er et «Finite impulse response» filter (FIR). En slik type filter kan ha få eller mange ledd alt etter hva man ønsker å oppnå. Leddene kan vektes forskjellig etter behov. Det er viktig er at summen av all vektingen blir 1. Dersom man velger å vekte alle leddene likt, slik som vi har gjort her, så får man i praksis et «Moving Average» filter.

2.5.1 FIR filter

Vi har også brukt filtrering av lyssignal i.f.m. automatisk kjøring av NXT robot. Da brukte vi først et FIR filter med få ledd, og med mest vekting på de «ferskeste» målingene. Etter hvert koblet vi ut de eldste leddene ved å vekte disse med 0.0.

```
% FIR-filter
if i > 10
     LysFiltrert(i) = Lys(i)*0.3+Lys(i-1)*0.3+Lys(i-2)*0.3+Lys(i-3)*0.1+
     Lys(i-4)*0.0+Lys(i-5)*0.0+Lys(i-6)*0.0+Lys(i-7)*0.0+Lys(i-8)*0.0+
     Lys(i-9)*0.0;
else
     LysFiltrert(i) = Lys(i);
End
```

2.5.2 IIR filter

Vi prøvde også et Infinite impulse response (IIR) filter for å teste ut om det ga bedre resultater, noe vi syntes at det gjorde. Her brukes tidligere «filtrerte» verdier sammen med verdier av målesignalet til å beregne den nye filtrerte verdien.

```
% IIR-filter
if i > 1
    LysFiltrert(i) = 0.4*Lys(i)+0.6*LysFiltrert(i-1);
else
    LysFiltrert(i) = Lys(i);
End
```

2.6 Program 1: Kjøring av Legorobot (Hele gruppen)

Program: Gruppe1331_Program1 KjoringAvNXT.m

2.6.1 Generelt

Vi lagde et program som kunne kjøre NXT robot både i Manuell modus, og i Automatisk modus. Ved manuell kjøring styres roboten med ved hjelp av Joystick. Ved automatisk kjøring brukes en enkel PID regulator. Vi velger mellom modus ved å bruke knappene på joystick.

2.6.2 Joystick

Vi startet med å laste inn joystick og definere en manuell trigger for å veksle mellom automatisk og manuell kjøring, en hendel for å justere farten til NXT roboten og en knapp for å lukke sensorer og avslutte programmet.

Utdrag fra kode:

```
%% Få tak i joystickdata
joystick        = joymex2('query',0);  % spør etter data fra joystick
JoyStopSwitch   = joystick.buttons(1);
if  joystick.buttons(3);
    JoyRegSwitch = 1;
end
if  joystick.buttons(4)
    JoyRegSwitch = 0;
end
% 32768 fremover, -32768 bakover
JoyForover(i)   = -joystick.axes(2)/327.68;
JoySideways(i)  = -joystick.axes(1)/327.68;

% Bruker JoyScale
JoyScale(i)     = -joystick.axes(3)/3276.8;
```

Da joysticken var veldig sensitiv rundt midtstilling lagde vi et død bånd på den slik vi kunne styre NXT mer behagelig.

Utdrag fra kode:

```
% Lager dødbånd på joystick aksene ut fra raw-data
if abs(-joystick.axes(2)) > 1500
   JoyForover(i)= -joystick.axes(2)/327.68;
else
   JoyForover(i)= 0;
end
if abs(-joystick.axes(1)) > 1500
       JoySideways(i)= -joystick.axes(1)/327.68;
else
JoySideways(i)= 0;
end
```

2.6.3 Lesing fra sensorer og filtreringen

Vi leser inn lysnivå fra lys-sensoren og lagrer det i en vektor Lys(i). Da inndataene er sensitive til støy, så filtrerer vi resultatet.

Etter å ha prøvd et FIR-filter, valgte vi å bruke et IIR-filter som bruker tidligere filtrerte verdier sammen med målesignalet. Det filtrerte signalet blir videre brukt i en PID-regulator. Se kapitel 2.5.1 og 2.5.2 ovenfor for informasjon om filtrene.

2.6.4 PID-regulator

For å kunne kjøre automatisk langs brukte vi enkel PID regulator. Regulatoren ble bygget basert på en pseudo kode fra en bruker manual til en Triconex PLC.

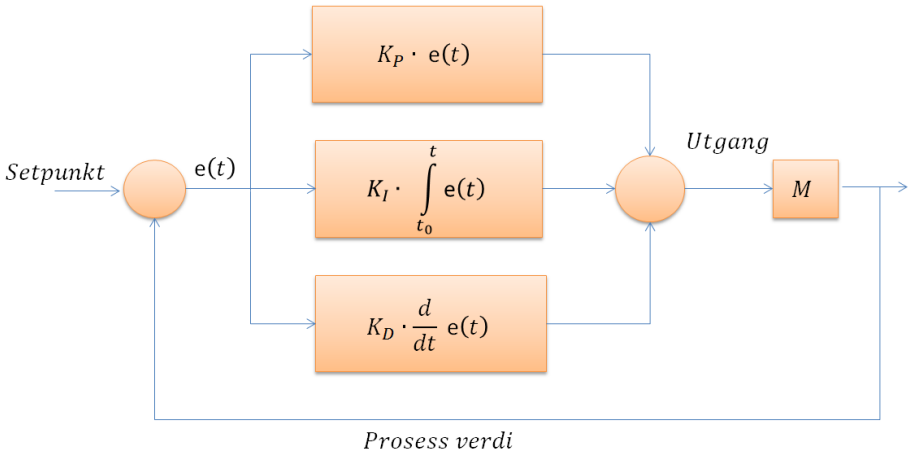
PID regulatoren har 3 forskjellige ledd, P, I og D som sammen eller hver for seg kan brukes til å fjerne et avvik i en prosess.

P leddet, eller forsterkningen, ser på avviket mellom prosessverdi (målt verdi) og settpunkt. Dette avviket kalles ofte «Error». Forsterkningsfaktoren P multipliseres med avviket, og resultatet sendes til regulator utgangen. Brukes kun P leddet i regulatoren, ender man oftest opp med et fast avvik enten like over eller like under settpunkt.

I leddet ser på avviket over tid, og integrerer dette. I en industriell regulator sier man ofte at P leddets verdi dobles i løpet av integrasjonstiden. I leddet brukes for å fjerne det faste avviket som P-leddet etterlater seg.

D leddet ser på endringen i avviket. Dersom avviket endrer seg og blir større, så reagerer D leddet med en rask respons på regulatoren for å motvirke dette. Dette virker også motsatt vei, så selv om avviket endres seg mot et mindre avvik, så prøver D leddet å motvirke dette også.

Disse 3 leddene adderes og sendes til regulator utgang som vist på skissen nedenfor:



Utdrag fra kode:

```
%% Regulator
 if JoyRegSwitch==1
   dt= deltaT;
   rSp= nullpunkt;       % Setter setpunkt lik nullpunkt (50% grått)
   rPv= LysFiltrert(i); % Setter prosessverid lik filtrert lysverdi
   rErr = rSp - rPv;     % Kalkuler Error : Setpoint - Process Value
   if pidI > 0.001       % Dersom  regulator har et I ledd
                         % Beregn Integrasjons bidraget
       iDel= iDel+ ((dt * rErr)/pidI * pidP);
   else                  % Hvis ikke
       iDel = 0;         % Nullstiller I ledd
   end
   if dt > 0.001        % Dersom  regulator har et D ledd
                         % Beregn Derivasjon bidraget
       dDel = (rErr - (pidD*(rPv - pvLast)) / dt) * pidP;
   else                  % Hvis ikke
       dDel = 0;         % Nullstiller D ledd
   end
   if iDel> maxCv
       iDel= maxCv;      % Anti-reset Windup function positiv retning
   end
   if iDel < minCv
       iDel= minCv;      % Anti-reset Windup function negativ retning
   end
   rCv = iDel+ dDel;     % Adderer
   if rCv > maxCv
       rCv = maxCv;      % Begrenser utgang til max verdi
   end
   if rCv < minCv
       rCv = minCv;      % Begrenser utgang til min verdi
   end
   pvLast = rPv;         % Oppdaterer forrige verdi
   PidOutput(i) = rCv;       % logger i vektorer for plotting
   Pbidrag(i)= rErr * pidP;
   Ibidrag(i)= iDel;
   Dbidrag(i)= dDel;
 else                    % Nullstiller når i manuell modus
   PidOutput(i) = 0;
   Pbidrag(i)= 0;
   Ibidrag(i)= 0;
   Dbidrag(i)= 0;
end
```

2.6.5 Motorpådrag

I manuell modus styres pådraget til motorene med Joystick. Joystick fremover gir fart til begge motorene fremover. Joystick bakover gir fart til begge motorene bakover. Sideveis bevegelse av joystick reduserer eller legger til et bidrag til fart, og får NXT robot til å svinge.

I automatisk modus setter vi en basisfart med «hendelen» på joysticken. PID regulatoren styrer sving funksjonen.

Vi har lagt inn en begrensning som hindrer at pådraget går ut over +/- 100%.

Utdrag fra kode:

```
%% beregner motorpådrag og lagrer i vektor
if JoyRegSwitch==1
    FartA = abs(2*JoyScale(i)) + (round(PidOutput(i)));
    FartB = abs(2*JoyScale(i)) - (round(PidOutput(i)));
else
    FartA = JoyForover(i)+ JoySideways(i);
    FartB = JoyForover(i)- JoySideways(i);
end
% Begrenser pådrag til +/- 100% for at NXT skal bli fornøyd
if FartA> 100
    FartA=100;
end
if FartA< -100
    FartA=-100;
end
if FartB> 100
    FartB=100;
end
if FartB< -100
    FartB=-100;
end
PowerA(i) = FartA;
PowerB(i) = FartB;

%% set output data
motorA.Power = PowerA(i);
motorA.SendToNXT();
motorB.Power = PowerB(i);
motorB.SendToNXT();
```

2.6.6 Derivert og integrert avvik

Når NXT roboten kjører langs gradienten øker det integrerte avviket dersom lyssensoren måler lysnivå utfor 50 % gråtone. Dette avviket vil bli større jo lengre roboten er utenfor midtlinjen. Avviket integreres som en absoluttverdi, slik at avvik på begge sider av senterlinjen legges til. Endringen i avvik pr. tid er lik den deriverte av avviket. Dette gir oss informasjon om hvor fort roboten beveger seg inn mot eller vekk fra senterlinjen. For detaljert informasjon om integrering og derivering se kapitel 2.3 og 2.4 ovenfor.

2.6.7 Plotting av data

For å se at ting fungerer som de skal plotter og sammenligner vi data samtidig som programmet kjører. Det nyttigste her er plott som sammenligner rå og filtrert lysverdi, regulatorens P, I og D-ledd, og de deriverte og integrerte avvikene.

Utdrag av kode:

Nedenfor viser 2 av plottene vi brukte under kjøring:

```
%% Plott data
  figure(1)
  subplot(2,2,1)
  bar(PowerA(i));
  axis([0 1 -100 100])
  title('Pådrag motor A')

  subplot(2,2,2)
  bar(PowerB(i));
  axis([0 1 -100 100])
  title('Pådrag motor B')
```

2.7 Resultat

NXT fungerer godt i manuell kjøring. Automatisk kjøring fungerte også tilfredsstillende. Vi fant at vi ikke kunne bruke Derivasjons leddet, da det var for mye støy på målesignalet. Integrasjonsleddet vårt integrerte avviket gjennom hele svingen, og dette førte til problemer med at «rette opp» igjen når svingen var ferdig. Dette kunne vi ha løst med å integrere kun de siste 20 målingene. Siden NXT kjørte tilfredsstillende valgte vi å gå videre og prioriterte kreative oppgaver.

3 Program 2: Dekoding av ASCII kode (Hele gruppen)

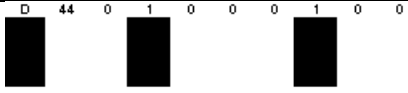
Program: Gruppe1331_Program2_Strekkode.m

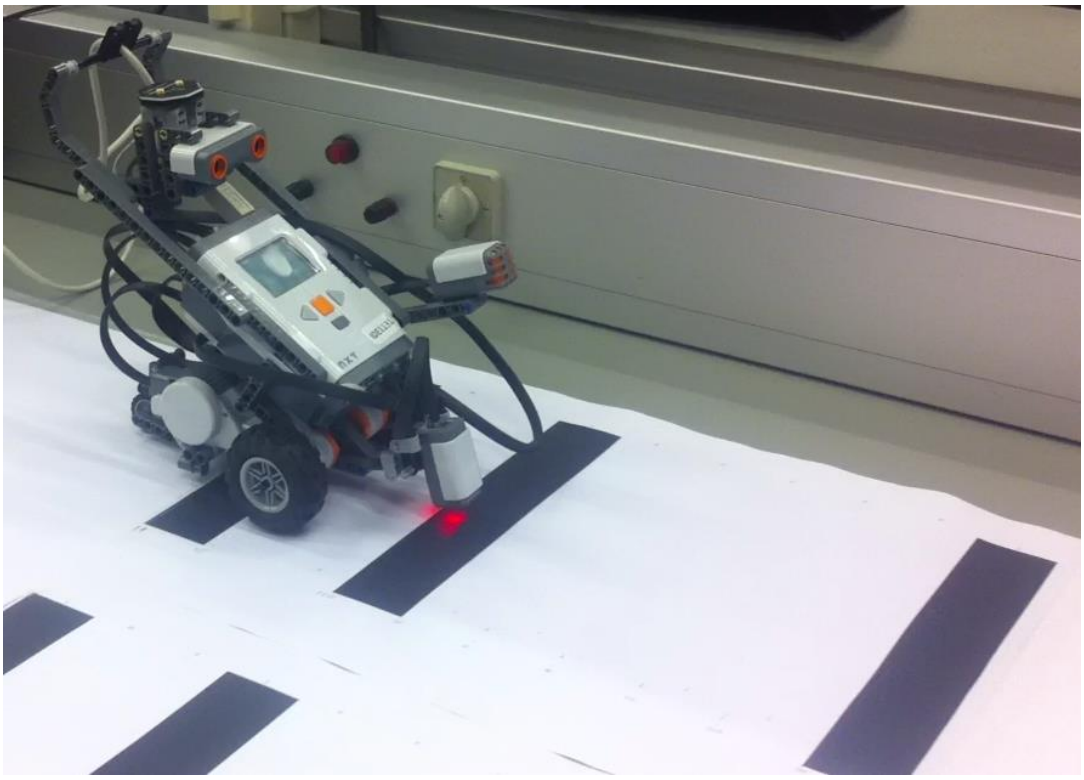
3.1 Problemstilling

Vi ønsket å lage et program som kunne tolke strekkoder som representerer bokstaver i ASCII binær kode. Dette har vi løst ved å la NXT kjøre over flere forskjellige A4 ark med strekkode som vist i tabell i kapitel 3.2 nedenfor. Vi bruker lyssensor til å lese inn binærkoden. Når NXT har lest inn 8 bit (8 hvite eller sorte søyler), stopper den automatisk og kalkulerer ASCII verdien og gjør den om til en bokstav. Bokstaven blir presentert på PC skjerm som vist i kapitel 3.3 nedenfor.

3.2 ASCII kode mønster

Forskjellige bokstaver er vist i tabellen nedenfor.

Bokstav	HEX Verdi	Desimal Verdi	Binær Verdi	Strekkode mønster (med kalibrerings strek)
A	41	65	0100 0001	
B	42	66	0100 0010	
C	43	67	0100 0011	
D	44	68	0100 0100	
E	45	69	0100 0101	



Lesehesten NXT in action.

3.3 Utskrift på skjerm

Utskrift på skjerm etter kjøring over bokstaven A:

```
143     0
173     1
203     0
233     0
263     0
293     0
323     0
353     1
Avlest Binærkode i desimal : 65
Tolket kode : A
```

3.4 Utdrag fra kode

Utklipp av koden fra Gruppe1331_deloppg1_Strekkode.m :

```
while ~JoyStopSwitch
%% Gir motor jevn, konstant fart
    motorA.Power = 10;
    motorA.SendToNXT();
    motorB.Power = 10;
    motorB.SendToNXT();

%%  Få tak i nye sensordata og lagre i vektor
Lys(i)   = GetLight(SENSOR_3);
distance = GetUltrasonic(SENSOR_4);

%% Filter
% IIR-filter
if i > 1
    LysFiltrert(i) = 0.4*Lys(i)+0.6*LysFiltrert(i-1);
else
    LysFiltrert(i) = Lys(i);
end

%% Fyller Strekkode(i) med enten 1 eller 0 etter målt Lys
if Lys(i) < LysGrense
    Strekkode(i) = 1; % gir 1 for sort
else
    Strekkode(i) = 0; % gir 0 for hvit
end


% Kode som kjøres før strekbredde er detektert
% Antall program sykluser telles for at programmet skal kunne
% vite hvor lant NXT kjørte før den kom til kalibreringsstreken
if (funnetBredde == 0) && (Strekkode(i) == 0)
    antallHviteStart = antallHviteStart + 1;
end

% Kode som kjøres når man har kommet frem til kalibreringsstreken
% Her telles program sykluser for å registrere hvor bred
% kalibreringstreken er.
if (funnetBredde == 0) && (Strekkode(i) == 1)
    bredde2 = bredde2 + 1;
end

% Kode som kjøres når man har funnet bredde.
% Stopp posisjon beregnes slik at NXT stopper automatisk når
% 8 sterker (hvite eller svarte) =8 bit/1 byte er lest inn.
if (bredde2 > 1) && (Strekkode(i) == 0)
    funnetBredde = 1;
    stoppVed = bredde2 * 10 + round(bredde2/2) ;
end

% Når NXT har kjørt til stopp posisjon, stoppes While loopen
% og programmet går videre til  dekoding av måleresultat.
if length(Strekkode) == (antallHviteStart + stoppVed);
    JoyStopSwitch=1;
end
%% Plotter måledata fortløpende
figure(1)
bar(Strekkode(end-90:end));
axis([0 100 -1 2])
title('Strekkode rådata')
drawnow
```

```
    %% Oppdaterer index
    i=i+1;

end

%% Initialiserer variabler til behandling av data
i = 1;
k = 1;
l = 0;
bredde = 0;
TolketVerdi(k) = 0;

%% Behandling av målt data

% Teller opp målte 0 før breddeindikator (første sorte søyle)
while Strekkode(i) == 0
    i = i + 1;
end

% Finner bredden til første sorte søyle
while Strekkode(i) == 1
    i = i + 1;
    bredde = bredde + 1;
end

% Plukker ut verdier fra Strekkode(i) med inkrement = bredde
% "round((bredde/2))" slik vi unngår feil verdi i grensen mellom 0 og 1
% Siden vi bruker round ved hver runde legger vi til 3 til bredden.
for j = (i + round((bredde/2)) + bredde):bredde+3:length(Strekkode)
    TolketVerdi(k) = Strekkode(j);
    k = k + 1;
    fprintf('%i   %i   \n', j,Strekkode(j))
end

%% Oversetter TolketVerdi til ASCII
m = 0;
AsciiVerdi = 0;
TolketVerdi2 = fliplr(TolketVerdi);


% Ascii verdi regnes ut slik:
%    verdi av bit0 * 2^0
%  + verdi av bit1 * 2^1
%  + verdi av bit2 * 2^2
%  + verdi av bit3 * 2^3
%  + verdi av bit4 * 2^4
%  + verdi av bit5 * 2^5
%  + verdi av bit6 * 2^6
%  + verdi av bit7 * 2^7
%  = Asciiverdi

for m = 1:length(TolketVerdi)
    AsciiVerdi = (TolketVerdi2(m)) * 2^(m-1) + AsciiVerdi;
end

bokstav = char(AsciiVerdi);

% Skriver ut tolket verdi på skjerm:
fprintf('Avlest Binærkode i desimal : %i\n', AsciiVerdi)
fprintf('Tolket kode : %s\n', bokstav)
```

3.5 Resultat

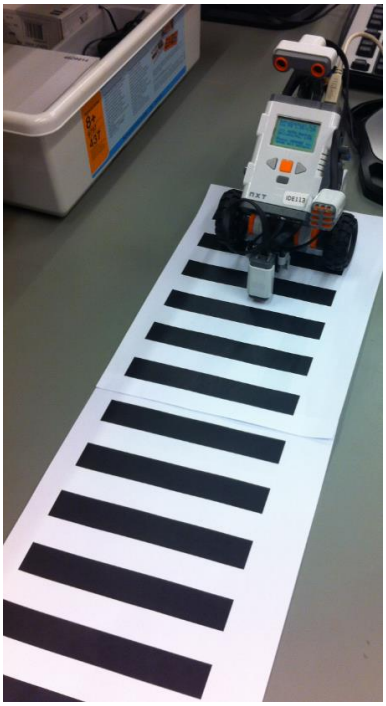
I dette prosjektet fikk vi sett på sammenhengen mellom binære tall, hexadesimale tall og desimaltall. Siden vi klarte å dekode alle mønstrene, så er konklusjonen at programmet virker etter hensikten.

4 Program 3: Fartsmåling v.h.a. strekkode (Hele gruppen)

Program: Gruppe1331_Program3_Fart.m

4.1 Problemstilling

Oppgaven var å lage et program som kunne fortelle hvilken hastighet NXT kjører med, ved å bruke måleverdier fra lyssensor.



4.2 Strekkode

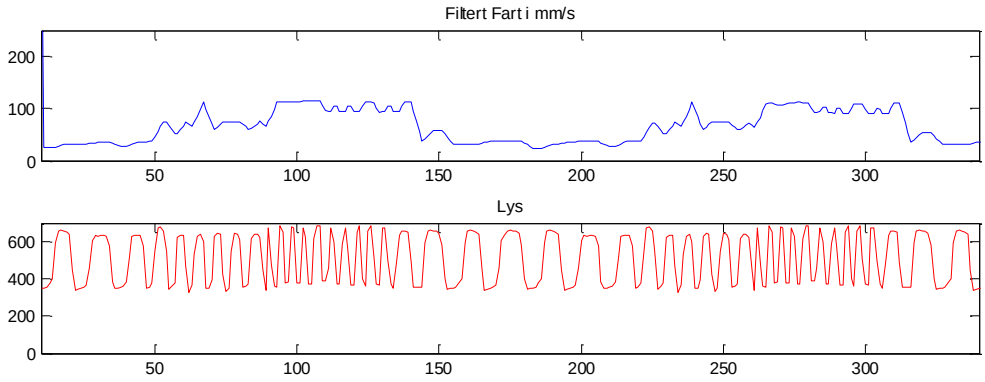
Dette programmet beregner farten NXT kjøres med basert på signaler fra lyssensor når NXT beveger seg over et strekkode mønster som vist i figur.

Bredden på de hvite og svarte feltene er 20mm. Programmet måler tiden det tar mellom hver gang lys signalet skifter mellom svart og hvit. Vi har satt lysgrensen mellom hvit og svart til konstanten 450, basert på testmåling av 50% gråtone.

Alle måleverdier mindre en 450 gir «svart», og alle verdier større eller lik 450 gir «hvit»

For å få en mest mulig representativ verdi for fart, bruker vi et FIR filter med 4 ledd og samme vekting, før denne verdien plottes.

4.3 Plotting av resultat



4.4 Utdrag fra kode

Koden som kalkulerer og filtrerer *fart*. Kode som er presentert tidligere i dokumentet er fjernet fra utdraget nedenfor.

```
tic;
while ~JoyStopSwitch
    %% Bruker knapp 1 på joystick som stoppknapp
.
.
    %% Bruker Joystick til å styre Robot
.
.
    % Sjekker at motorpådraget ikke overskrider max og min grenser.
.
.
    % Kopierer fartB inn i vektor PowerB
.
.
    % Sender pådrag til motorene
.
.
    %% Hent nye sensordata og lagre i vektor
.
.
    % Bruker grenseverdien "lysGrense" til å bestemme om fargen som
    % lyssensor ser er sort eller hvit.
        if Lys(i) < lysGrense
        Farge(i) = 1; % gir 1 for sort
    else
        Farge(i) = 0; % gir 0 for hvit
        end

  % Dersom fargen er sort, utføres koden nedenfor.
    if Farge(i) == 1 % Sort
        if j < 1            % Ved første måling på sort felt
            deltaT = toc;   % Måles tid siden første måling på hvit felt
            tic;            % Stoppeklokken startes igjen
        end
        j = j + 1;          % j er antall sorte målinger
        k = 0;              % k er antall hvit målinger
    end                     % og nullstilles her.

    % Dersom fargen er Hvit, utføres koden nedenfor.
    if Farge(i) == 0 % Hvit
        if k < 1            % Ved første måling på hvit felt
           deltaT = toc;    % Måles tid siden første måling på sort felt
           tic;             % Stoppeklokken startes igjen
        end
        k = k + 1;          % k er antall hvit målinger
        j = 0;              % j er antall sorte målinger
    end                     % og nullstilles her.
    DeltaT(i) = deltaT
    Fart(i) =   ((strekBreddeImm)/DeltaT(i));      % Kalkulert fart (v=s/t)

    % Filtrerer Farten med et FIR filter med 4 ledd.
    if i > 4
        FartFiltrert(i)= Fart(i)*0.25+Fart(i-1)*0.25+Fart(i-2)*0.25+
        Fart(i-3)*0.25;
    else
        FartFiltrert(i)= Fart(i);
    End


     %% Plotter variablene
.
.
   end
```

4.5 Resultat

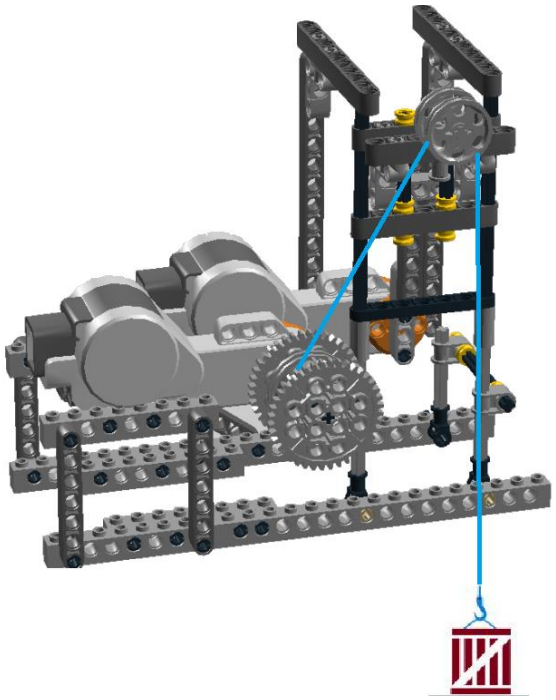
Vi klarte å få til en forholdsvis bra beregning av fart med forholdsvis breie striper. Hadde utstyret taklet smalere striper kunne nok resultatet blitt enda bedre. Vi endte opp med å bruke forholdsvis breie striper for at lyssensoren skulle kunne se dette som svart og hvitt. Dersom stripene ble for smale endte vi opp med å lese grått når farten ble stor. Og da kunne vi ikke bruke dette måleprinsippet.

5 Program 4: Heave Compensator

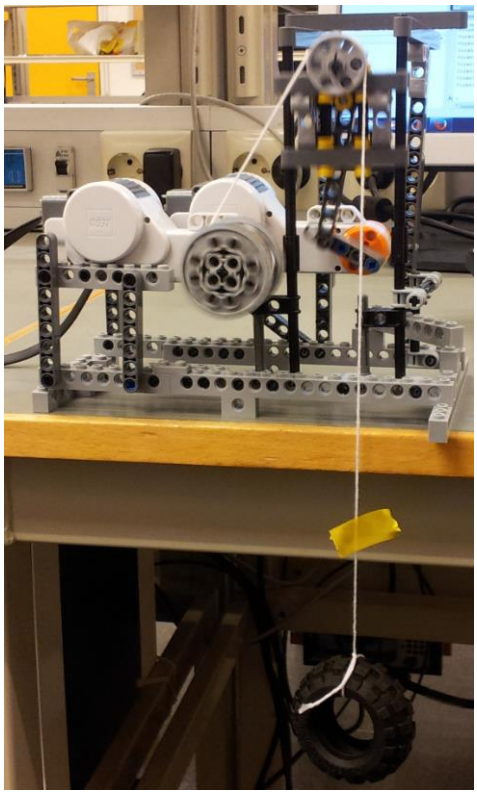
Program: Gruppe1331_Program4_HeaveComp.m

5.1 Problemstilling

Bygge en vinsj som kan kompensere for bølgebevegelse. Først tenkte vi å bygge en maskin som brukte ultralydsensoren til å måle en vertikal bevegelse for på den måten å detektere vertikal posisjonsendring og videre også vertikal fart v.h.a. derivasjon. Fant så ut at det ville bli en kjekkere løsning dersom vi lagde en maskin hvor en av motorene fungerte som bølge generator, og en annen som vinsj. Da kunne ved å bruke sinus til vinkelen på motoren til å finne tilnærmet vertikal posisjon på trinsen som tråden/wiren går over.



Model av maskin laget med Lego Digital Desingner



Maskin i operasjon.
(gul tape for å vise at heave kompensering fungerer)

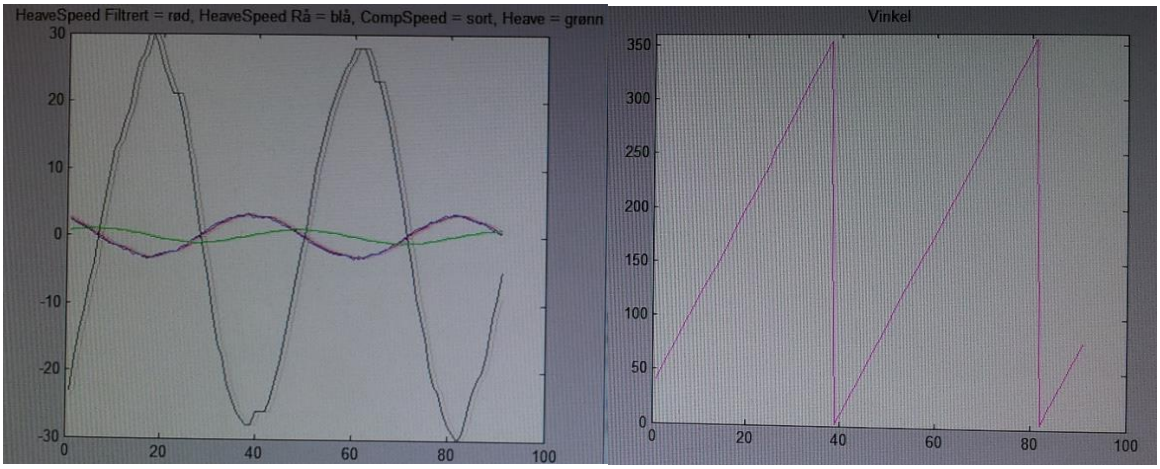
5.2 Maskinens virkemåte

Lego digital designer ble brukt til å konstruere en maskin som vist i figuren over til venstre. P.g.a. litt begrenset utvalg Legoklosser ble maskinen bygget slik som bilde over til høyre viser.

Motorene styres av samme kontroller, men har helt uavhengige program. Motoren som løfte trinsen vertikalt opp og ned går med en tilnærmet konstant hastighet. Motorvinkelen hentes ut, og bearbeides slik at vi får en vinkel på 1 til 360° for hver omdreining. Dette blir så gjort om til radianer.

Ved å bruke sinus til vinkelen, får vi ut den vertikale bevegelsen. Denne bevegelsen deriveres slik at vi får ut den vertikale farten.

Den andre motoren fungerer som vinsj. Denne forsøker å holde lasten i konstant posisjon. Altså kompensere for «bølgebevegelsen». Dette gjøres med å forsterke den vertikale bølgehastigheten med et enkelt P ledd å bruke dette som pådrag inn til vinsjmotoren.



Figurene til høyre over viser motor vinkelen til bølge generator. På figuren til høyre viser grønn pen bølge høyde. Den rød og blå viser bølgens vertikale fart. Den sorte viser pådraget til vinsj motoren.

5.3 Utklipp fra kode

```
%% While Loop
while ~JoyStopSwitch
    tic;

    %% få tak i joystickdata
    joystick        = joymex2('query',0);       % spør etter data fra joystick
    JoyStopSwitch   = joystick.buttons(1);

    % Bruker Motor A som heave generator
    motorA.Power = 18;
    motorA.SendToNXT();
    data = motorA.ReadFromNXT();            % Hent motor info
    Vinkel(i) = mod(data.Position,360) ;    % Flytt posisjonsdata inn i vinkel
    heaveNow = sin((Vinkel(i)/180)*pi);     % Beregn Heave verdi
    Heave(i) = heaveNow;                    % Flytte  Heave verdi inn i vektor
    heaveSpeed = (heaveNow-heaveOld)/deltaT; %Derivere posisjon til fart
    HeaveSpeed(i) = heaveSpeed;             % Flytte fart inn i vektor
    % Filtrerer fart
    if i > 2
        SpeedFiltrert(i) = HeaveSpeed(i)*0.5+HeaveSpeed(i-1)*0.5;
    else
        SpeedFiltrert(i) = HeaveSpeed(i);
    end

    MotorSpeed(i) = -round(SpeedFiltrert(i)*9); %Forsterke Signal til Vinsj
    % Begrenser motor pådrag til +/- 100
    if MotorSpeed(i) >= 100
        MotorSpeed(i) = 100;
    end
    if MotorSpeed(i) <= -100
        MotorSpeed(i) = -100;
    end
    motorB.Power = MotorSpeed(i);
    motorB.SendToNXT();
    heaveOld = heaveNow;
    % Ta vare på tid i tidsvektor
    Tid(i) = tidOld+deltaT;
```

```
    tidOld = Tid(i);
    % Generere utskrifter til skjerm
    fprintf('Vinkel = %d  , ', Vinkel(i));
    fprintf('Heave  = %f   ', Heave(i));
    fprintf('HeaveSpeed  = %f   ', HeaveSpeed(i));
    fprintf('MotorSpeed  = %d   ', MotorSpeed(i));
    fprintf('Tid    = %f   \n', Tid(i));
    i = i + 1;          % Inkrementerer omløpsteller

    %% Plot
    figure(1)
    plot(SpeedFiltrert(end-90:end),'r')
    hold on
    plot(HeaveSpeed(end-90:end),'b')
    plot(MotorSpeed(end-90:end),'k')
    plot(Heave(end-90:end),'g')
    hold off
    title('HeaveSpeed Filtrert = rød, HeaveSpeed Rå = blå, CompSpeed = sort,
                                                        Heave =  grønn')
    axis([0 100 -30 30])

    figure(2)
    plot(Vinkel(end-90:end),'m')
    title('Vinkel')
    axis([0 100 0 360])

    deltaT = toc;
end
```

5.4 Resultat.

Heave kompensatoren fungerte bra. Det kunne likevel vært interessant å prøvd å forbedre maskinen ytterligere ved å bruke akselerometer som input i tillegg til posisjonsmåling.

6 Prohgram 5: Robot Dans (Utført av Per-Ove)

Program: Gruppe1331_Program5 RobotDans.m

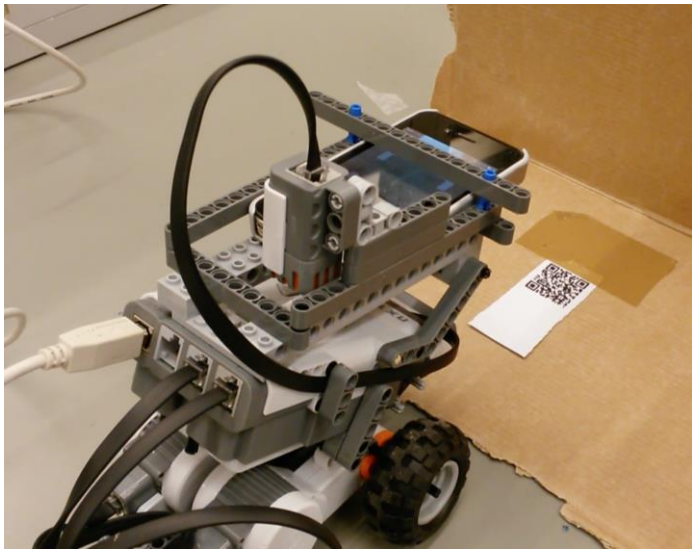
6.1 Problemstilling

Dette programmet ble laget for å få NXTen til å danse etter en ekstern lydkilde. En kreativ måte for å få til dette var å la mobilen skanne en QR-kode, åpne YouTube og spille av en sang. I stedet for å ha et eget program hvor NXT følger etter hånden, ligger det posisjonsregulering med justerbart settpunkt inne i danseprogrammet. Når avstanden til settpunktet er tilnærmet lik null, snur NXTen 180° og beveger seg etter musikken. Dette skjer ved hjelp av et enkelt derivasjonsledd som bestemmer retningen til bevegelsen. Hastigheten til bevegelsen bestemmes av amplituden til registrert lyd.

6.2 Hensikt

Målet med oppgaven er å undersøke hvordan roboten reagerer når mer abstrakte måleverdier som bevegelse og lyd i rommet sendes rett til motorene. Ved å la avstandsmåleren og lydsensoren styre motorpådraget ble det enkelt å se om programmeringen virker eller ikke.

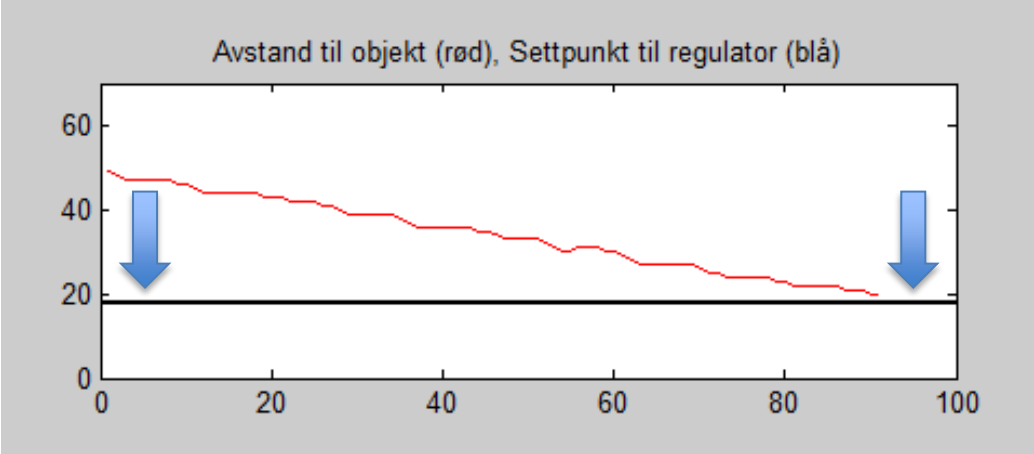
Roboten beveger seg elegant etter musikken.



6.3 Posisjonsregulering

I første del av programmet kjører roboten frem slik at avstanden fra distansemåler til objekt er 20 cm. Avviket regnes ut ifra settpunktet minus posisjonen til roboten. Når avviket er stort blir motorpådraget stort. Pådraget til motorene øker **P**roporsjonalt med avviket og kan dermed betraktes som en P-regulator. Siden avviket raskt korrigeres ved P-leddet er det ikke nødvendig med integral og derivasjonsledd.

Figuren under viser hvordan roboten kjører når settpunktet flyttes.



6.4 Utdrag fra koden

```
%% Initialisering av variabler i program

FartA       = 0;    % initialverdi motor A
FartB       = 0;    % initialverdi motor B

deltaT      = 1;    % initialverdi tid
LydverdiOld = 0;    % initialverdi gammel lysverdi
OneRound    = 0;    % initialverdi runder snurret

%% Initialisering av regulator

settpunkt   = 20;   % ønsket verdi
rP          = 10;   % Proporsjonal forsterking
regOutput   = 0;    % 0 ved f¯rste cycle


while ~JoyStopSwitch

    %% Lager logikk for knappene pÂ joystick.
    if JoyStepUp>0
        settpunkt=settpunkt-1;  % trekker 1 fra settpunktet ved Â bevege mini-joystick opp
    end
    if JoyStepDown>0
        settpunkt=settpunkt+1;  % legger 1 til settpunktet ved Â bevege mini-joystick ned


%% P-Regulator Del 1
%-------------------------------------------------------------------------------------------

    if JoyRegSwitch==1            % Regulator:

        rSp= settpunkt;           % regSetpoint     ønsket verdi
        rPv= Avstand(i);          % regProsessValue - faktisk verdi
        %
        rErr(i) = rSp - rPv;      % avviket mellom ønsket verdi og faktisk verdi
        regOutput= rErr(i) * rP;  % proporsjonal-leddet i regulatoren (P-regulator)

        FartA =-regOutput;        % adresserer signalet fra regulatoren som motorpÂdrag
        FartB =-regOutput;
    else
        % Kj¯res med joystick hvis regulator ikke er aktivert
        FartA = JoyForover(i)+ JoySideways(i);
        FartB = JoyForover(i)- JoySideways(i);
    end


%% Beveger seg i forhold til musikken Del 2
%-------------------------------------------------------------------------------------------

   % Filtrerer registrert lydsignal
   LydScaled(i)=(Lyd(i)/25);
    if i > 1
        LydFiltrert(i) = 0.4*LydScaled(i)+0.3*LydScaled(i-1)+0.3*LydScaled(i-1);
    else
        LydFiltrert(i) = Lyd(i);
    end

    % Starter sekvens
    if JoyDanceSwitch==1            % Danser etter musikk:
        if OneRound<1               % betingelse for at NXT skal snu
            motorA.TachoLimit=720;
            motorA.Power=70;        % snur NXT 360 grader med 70% pådrag til motor A
            motorA.SendToNXT();
            motorA.WaitFor();       % venter til NXT har snudd
            OneRound=1;             % forsikrer seg om at sekvensen ikke starter pÂ ny
            motorA = NXTMotor('A','SmoothStart',true);
        end
```

Fortsettelse på while-loopen

```
        % Deriverer registrert lydverdi
         j = j + 1;
         Derivert(j) = (LydFiltrert(i) - LydverdiOld)/deltaT/100;
         LydverdiOld = LydFiltrert(i);

         % Betingelse for at NXT skal bevege seg etter musikken
         % Konstant fart pÅ 10% +/- lydverdi med den deriverte som fortegn

         if Lyd(i)>150
             FartA =10+ round(Derivert(j))*(round(LydFiltrert(i)));
             FartB =10- round(Derivert(j))*(round(LydFiltrert(i)));
         else
             FartA=0;
             FartB=0;
         end
    end
```

6.5 Bevegelse etter musikk

Andre del av programmet bruker den deriverte til målt lydverdi for å bestemme retningen til roboten. Hvis lydnivået er på vei opp er den deriverte positiv og roboten beveger seg mot klokken (sett ovenfra). Den målte lydverdien filtreres ved hjelp av et FIR filter og deles på 25 for at resultatet skal ligge mellom 0 og 100.

6.6 Resultat

Dette var en morsom oppgave som både skapte engasjement og ønske om å forske videre på bevegelse i motsatt retning av måleverdi. Det ble en effektiv måte å visualisere virkemåten til en P-regulator. Den dansende roboten hadde derimot ikke vunnet noen dansekonkurranse.

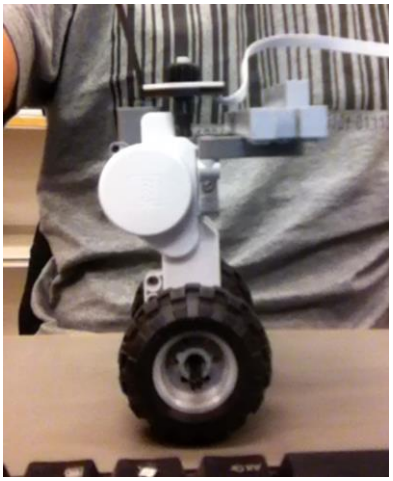
7 Program 6: Segway (Utført av Per-Ove)

Program: Gruppe1331_Program6_Segway.m

7.1 Problemstilling

Målet var å bygge videre på musikk. Ved å bruke akselerometeret og lyssensoren til å bestemme avstanden fra likestilling og beregne pådrag ut fra målt verdi. Dette krever en bedre regulering enn program_. Her kan avviket vokse fortere og er nødvendig med et D-ledd i regulatoren. Selv om vi på forhånd fikk vite at kommunikasjonen mellom Matlab og motorene var for treg var hovedmålet med oppgaven å eksperimentere med invertert bevegelse.

7.2 Utklipp fra kode



```
%% Regulator parametre eksternt

% P-ledd og Derivasjons-ledd
rP = 0.42;           % Proporsjonalledd
rD = 0.04;           % Derivasjonsledd

% Registrer f¯rste mÂlte lysverdi som settpunkt
nullpunkt = GetLight(SENSOR_3);

% Lager logikk for valgt mÂleverdi
lys=1;

% Regulator initialisering
rPdel = 0;     % Null ved f¯rste cycle
rDdel = 0;     % Null ved f¯rste cycle
ErrLast = 0;   % Null ved f¯rste cycle

%% PD-Regulator
%-------------------------------------------------------------------------------------------

if JoyRegSwitch==1

   rCycleT= deltaT;

   if lys==1                      % velger måleverdi
    rErr = nullpunkt-Lys(i);      % kalkulerer avviket (lysverdi første cycle - målt lysverdi)
   else
    rErr = IMUvinkelFiltrert(i);  % setter avviket til å være målt akselometer verdi
   end

 % Regulator

   rPdel = rErr * rP;                                % beregner P leddet

    if rCycleT > 0.001                               % hvis D leddet er aktivert
       rDdel = ((rErr - ErrLast) / rCycleT) * rD;    % Deriverer mÂlt verdi og ganger med D-ledd
    else
       rDdel = 0;
    end
        rTotcv = rPdel+rDdel;                         % totalt pÂdrag til motor

        if rTotcv > 100                               % begrenser pådraget til +/- 100
            rTotcv = 100;
        end
        if rTotcv < -100
            rTotcv = -100;
        end

        motorA.Power = (round(-rTotcv)); % sender resultat til motor
        motorA.SendToNXT()
 else
        motorA.Power = 00;               % sender 0 til motor hvis ikke regulering er aktivert
        motorA.SendToNXT();
 end
```

7.3 Resultat

Det viste seg at oppgaven ble tidkrevende, men veldig lærerik. Ut i fra tidligere erfaringer så er en unøyaktig måleverdi vanskelig å jobbe med. Uansett hvor god regulering man oppnår så vil resultatet være preget av målerverdien. Eksempel på dette kan være en filtrert måleverdi som i utgangspunktet har mye støy. Ved å filtrere signalet oppnår man myke bevegelser, men signalet kan ha store forsinkelser. I programmet for Segway anså vi målesignalene til å respondere relativt raskt, mens forsinkelsen lå i reaksjonsevnen til motoren. Dette viste seg å være feil og at problemet lå i kommunikasjonen mellom Matlab og roboten.

Vi prøvde å få ned syklustiden ved å fjerne unødvendig seksjoner som plot og deler av regulatoren som ikke var nødvendige. Siden avviket endret seg fort la vi til et derivasjonsledd som økte motorpådraget betraktelig ved høye rotasjonshastigheter. Vi kom frem til at et lite bidrag fra D-leddet førte til bedre reaksjon. Ettersom hovedmålet var å eksperimentere med inverse bevegelser var resultatet tilfredsstillende på flere måter til tross for at reaksjonene var forsinket.

For å oppnå en raskere kommunikasjon med motorene så vi på alternativet «DirectMotorCommand». Dette er den raskeste måten for å sende direkte kommandoer til roboten gjennom Matlab, men har større unøyaktighet på sendte signaler. Det viste seg også at programvaren i Lego roboten forsøker å korrigere på egenhånd i enkelte tilfeller. Ettersom DirectMotorCommand ble oppdaget seint i prosjektet valgte vi å prioritere andre programmer. Hvis muligheten skulle by seg senere vil det være av interesse å fullføre prosjektet. Da kunne det også vært lærerikt å se på hvordan man kunne korrigert for avstanden til massesenter og implementert en mer komplisert regulering.

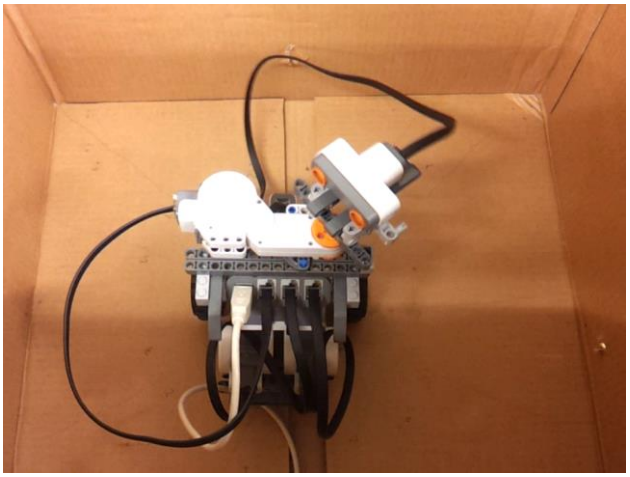
8 Program 7: Areal av eske (Utført av Patrick)

Program: Gruppe1331_Program7_Areal.m

Her plasseres roboten midt i en eske slik vi kan måle og registrere data nødvendig for å beregne dens areal. Målt data vil så bli behandlet og areal bli regnet ut ved hjelp av forskjellige metoder. Resultatet av disse metodene vil så bli sammenlignet med hverandre og det manuelt målte resultatet.

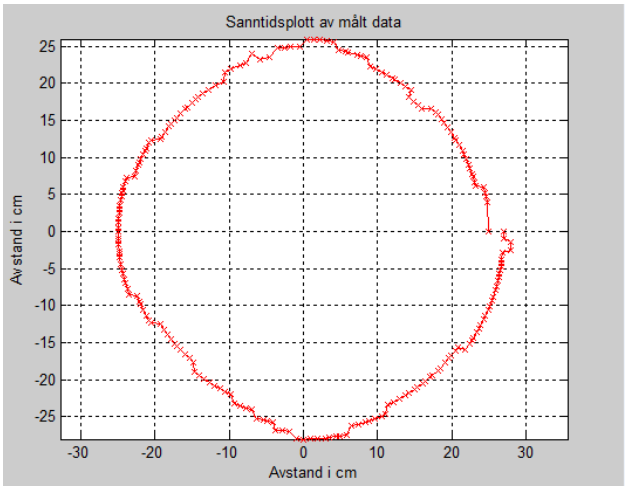
8.1 5.1 Måling av eske

For å måle esken roteres motoren, og dermed ultralyd-sensoren 360° samtidig som vinkel og avstand lagres i hver sine vektorer. Målt data blir også plottet samtidig med målingen. Dette for å lett kunne se en representasjon av hvordan målingen går.



Vi prøvde først å rotere motoren grad for grad og lagre data, men det førte til at det ble vanskelig å balansere nok kraft til å få den til å snu, og lite nok kraft for å unngå at ultralyd-sensoren ristet.

Metoden vi bruker gir varierende antall målinger, men flyten på motoren gir jevnere målinger på både vinkel og distanse da ultralyd-sensoren ikke rister frem og tilbake.



Vi starter med å nullstille motorposisjonen slik vi kan stoppe målingen når motorvinkelen er nådd 360°.

Funksjonen *pol2cart* oversetter polarkoordinater til kartesiske koordinater. Dette gir to vektorer X og Y med henholdsvis X og Y koordinater som vi bruker ved plotting og arealberegning.

For hver iterasjon av while-løkken plottes de sist beregnede koordinatene.

Utdrag fra kode:

```
%% Nullstiller motorposisjon
motorC.ResetPosition();

%% Bruker motorC og UL-sensor til å lagre vinkel og avstand i vektorer.
Stopper når vinkel er større enn 360 grader.
while vinkel <= 360
    %% Gir motor kraft
```

```
    motorC.Power = 11;
    motorC.SendToNXT();

    %%  Få tak i nye sensordata og lagre i vektor
    vinkelStruct = motorC.ReadFromNXT();      % Henter struktur med motordata
    Vinkel(i) = vinkelStruct.Position; % Henter motorvinkel
    vinkel = Vinkel(i); % Oppdaterer vinkeltelleren
    Distanse(i) = GetUltrasonic(SENSOR_4); % Distanse fra UL-sensor

    % Konvertere grader til radianer
    VinkelRad(i) = (((Vinkel(i))/180)*pi);

    % Polarkoordinater til kartesiske X,Y-vektorer
    [X(i),Y(i)] = pol2cart(VinkelRad(i), Distanse(i));

    %% Plotting av målt data
    figure(1)
    plot(X,Y,'-rx')
    hold on
    axis([-35 35 -35 35])
    grid
    title('Plott ved kartesiske koordinater')
    xlabel('Avstand i cm')
    ylabel('Avstand i cm')
    drawnow

    %% Oppdatering av tellere/variabler
    i = i + 1;
end
```

8.2 Beregning av areal

Program: Gruppe1331_deloppg3_Areal.mat

8.2.1 Matlab funksjonen polyarea

Funksjonen `polyarea` regner ut arealet av et gitt polygon, her gitt ved koordinatvektorene X og Y.

```
% Areal vha polyarea-funksjonen.
% polycount beregner arealet av polygonet spesifiset av vektorenes kanter.
areal1= polyarea(X,Y);
```

Da dette er en funksjon tilrettelagt til vårt spesifikke ønske kan bruke resultatet som en referanse for de følgende metodene.

8.2.2 Matlab funksjonen trapz

Funksjonen `trapz` bruker trapesmetoden til å tilnærme seg integralet av kurven. Trapesmetoden består av å estimere integralet til en funksjon med *n* antall trapeser.
Da dette er et polygon kan vi si at det består av to kurver som sammen blir et polygon. Integralet mellom dem kan man da få om man trekker integralet av den nederste kurven fra integralet av den øverste kurven.

For å benytte funksjonen trenger vi en indeks slik vi kan benytte metoden nevnt ovenfor. Dette for å «dele» polygonet i to kurver.

```
% Finner vektorenes maks-verdi og dens indeks vha max-funksjonen
[xMaksVerdi,xMaksIndeks] = max(X);
```

Deretter utfører vi selve beregningen og får arealet av det plottede polygonet.

```
% Areal vha trapz-funksjonen
areal2 = trapz(X(1:xMaksIndeks),Y(1:xMaksIndeks)) -
trapz(X(xMaksIndeks+1:end),Y(xMaksIndeks+1:end));
```

8.2.3 Trigonometri og Herons Formel

Da målingene kan tolkes som trekanter, kan vi bruke en generell formel for en trekants areal for å regne ut det målte arealet.

$$\Delta = \frac{1}{2} bcsinA$$

Areal

Utdrag fra kode:

```
m = length(X);
l = 1;
% Areal ved generell trigonometrisk formel
% Areal3(1) vil bli arealet av trekanten mellom siste og
% første element i Distanse-vektoren.
Areal3(l)= (Distanse(m) * Distanse(l) * sin((2*pi -
VinkelRad(m))+VinkelRad(l)))/2;
for l = 2:(length(X))
    Areal3(l) = (Distanse(l) * Distanse(l) * sin(VinkelRad(l)-
    VinkelRad(l-1)))/2;
end
areal3 = sum(Areal3);
```

Til slutt summeres arealet til alle trekantene og vi får arealet til det målte polygonet.

Vi kan også bruke Herons formel for å regne ut arealet av hver trekant. Herons formel forutsetter at vi kjenner alle trekantens sider. For å finne den siste bruker vi cosinus-setningen.

$$a^2 = b^2 + c^2 - 2bccos(\alpha)$$

Cosinus-setningen

Utdrag fra kode:

```
m = length(X);
l = 1;
% Finner lengden på motstående side vha cosinus-setningen.
% LengdeBC(1) vil bli lengden mellom siste og første element av
% Distanse-vektoren
LengdeBC(l) = sqrt(Distanse(m)^2 + Distanse(l)^2 -
2*Distanse(m)*Distanse(l)*cos((2*pi - VinkelRad(m))+VinkelRad(l)));
for l = 2:1:(length(X))
    LengdeBC(l) = sqrt(Distanse(l-1)^2 + Distanse(l)^2 -
    2*(Distanse(l-1)*Distanse(l))*cos(VinkelRad(l)-VinkelRad(l-1)));
end
l = 1;
```

Videre bruker Herons formel semiperimeteret til trekantene, noe som vil si halve trekantenes omkrets.

$$s \equiv \frac{(a+b+c)}{2} \qquad \Delta = \sqrt{s(s-a)(s-b)(s-c)}$$

Semiperimeter *Herons formel*

Utdrag fra kode:

```
% Finner semiperimeteret til trekantene
SemiPerimeter(l) = (LengdeBC(l) + Distanse(l) + Distanse(m))/2;
for l = 2:(length(X))
   SemiPerimeter(l) = (LengdeBC(l) + Distanse(l) + Distanse(l-1))/2;
end
l = 1;
```

Vi bruker så semiperimeteret i Herons formel og finner arealet til hver trekant.

Utdrag fra kode:

```
% Bruker Herons formel for utregning av areal
Areal6(l) = sqrt(SemiPerimeter(l)*(SemiPerimeter(l)-
LengdeBC(l))*(SemiPerimeter(l)-Distanse(l))*(SemiPerimeter(l)-Distanse(m)));
for l = 2:(length(X))
    Areal6(l) = sqrt(SemiPerimeter(l)*(SemiPerimeter(1)-
LengdeBC(l))*(SemiPerimeter(l)-Distanse(l))*(SemiPerimeter(l)-Distanse(l-
1)));
end
areal6 = sum(Areal6);
```

Til slutt summeres arealet til hver av trekantene og vi får arealet til det målte polygonet.

8.2.4 Koordinat regning

Arealet til et irregulært polygon kan utregnes ved par av XY-koordinater. For å gjøre dette kan vi benytte oss av følgende formel, kalt «Shoelace formula» da man multipliserer på kryss og tvers.

Metoden kan ligne på integralmetoden ovenfor ved at man trekker i fra arealet under polygonet mens man går rundt det.

$$\Delta = \frac{1}{2}\left(\begin{vmatrix} x_1 x_2 \\ y_1 y_2 \end{vmatrix} + \begin{vmatrix} x_2 x_3 \\ y_2 y_3 \end{vmatrix} + \cdots + \begin{vmatrix} x_n x_1 \\ y_n y_1 \end{vmatrix}\right)$$

$$\Delta = \left|\frac{1}{2}\left((x_1 y_2 - y_1 x_2) + (x_2 y_3 - y_2 x_3) + \cdots + (x_n y_1 - y_n x_1)\right)\right|$$

«Shoelace formula»

For å oppfylle formelens krav om alternerende multiplikasjon og å til slutt multiplisere x_n med y_1 og omvendt, settes variabelen m først slik at m er ved siste hjørne og så at variabelen videre er 1 lavere enn l.

Utdrag fra kode:

```
% Areal vha koordinatregning
% Modifisert kode fra http://alienryderflex.com/polygon_area/
% For å tilfredsstille alternerende multiplikasjon og siste ledd av
% formel defineres m (der length(X) er antall hjørner) slik:
m = length(X) - 1;
for l = 1:length(X)
    Areal4(l) = (X(m) + X(l)) * (Y(m) - Y(l));
    m = l;
end
areal4 = sum(Areal4) / 2;
```

8.2.5 Sektorer

For å teste tapet av informasjon regnet vi ut arealet ved hjelp av sektorer. For å få sektorer istedenfor trekanter tok vi medianen av to hosliggende målinger og beregnet arealet til den påfølgende sektoren.

$$A = \frac{r^2 \theta_{rad}}{2}$$

Areal av sektor

Utdrag fra kode:

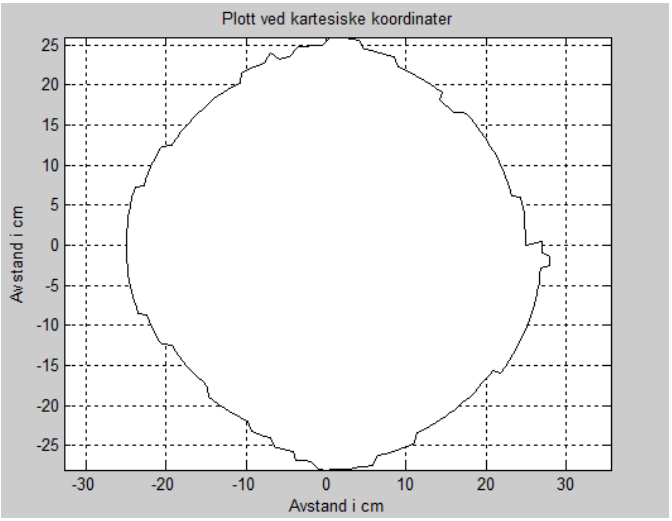
```
% Areal vha sektorer
% Tar gjennomsnittet av to distanser og beregner arealet av
% sektoren de og vinkelen lager.
k = 1;
vinkelRad=(2*pi - VinkelRad(m))+VinkelRad(k);
ArealSektor(k) = (vinkelRad*(median(Distanse(m),Distanse(k))^2))./2;
for k = 2:length(Distanse)
    vinkelRad = VinkelRad(k) - VinkelRad(k-1);
    ArealSektor(k) = (vinkelRad * (median(Distanse(k-1),
    Distanse(k))^2))./2;
end
areal6 = sum(ArealSektor);
```

Summert ga dette et tilnærmet areal av polygonet.

8.3 Plotting og presentering av data

Resultatet blir presentert i kommandovinduet. Her valgte vi å også vise et plott om lettere viser det målte polygonet.

```
Areal målt ved forskjellige metoder:

Polycount           = 1963.26 cm^2
Trapes              = 1963.95 cm^2
Gen.trekant formel  = 1963.47 cm^2
Herons formel       = 2033.60 cm^2
Koordinatregning    = 1975.51 cm^2
Sektorer            = 1963.50 cm^2

Reelt areal er 43cm * 46cm = 1978 cm^2
>>
```



Utdrag fra kode:

```
%% Plotter målt data
figure(2)
plot(X,Y,'-rx')
fill(X,Y,'w')
axis equal
grid
title('Plott ved kartesiske koordinater')
xlabel('Avstand i cm')
ylabel('Avstand i cm')

%% Presenterer resultat i kommandovinduet
disp(sprintf('\nAreal målt ved forskjellige metoder: \n'))
disp(sprintf('%s \t\t\t= %.2f cm^2', 'Polycount', areal1));
disp(sprintf('%s \t\t\t\t= %.2f cm^2', 'Trapes', areal2));
disp(sprintf('%s \t= %.2f cm^2', 'Gen.trekant formel', areal3));
disp(sprintf('%s \t\t= %.2f cm^2', 'Herons formel', areal4));
disp(sprintf('%s \t= %.2f cm^2', 'Koordinatregning', areal5));
disp(sprintf('%s \t\t\t= %.2f cm^2\n', 'Sektorer', areal6));
disp(sprintf('Reelt areal er 43cm * 46cm = 1978 cm^2'));
```

Vi bruker `sprintf` for å letter kunne formatere den viste teksten.

8.4 Resultat

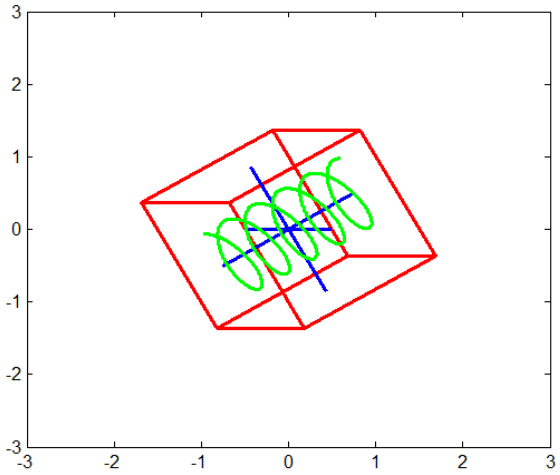
Det viste seg at alle metodene ga ganske gode tilnærminger av de reelle målene. Variasjonene kan skyldes flere ting. Koordinatregning fungerer for eksempel ikke dersom koordinatene «krysser» eller overlapper hverandre.
Sektorregning er i seg selv usikkert da målingene kan tolkes som trekanter og ikke sektorer.
Overraskende nok var resultatet ved sektorer nært det `polyarea` (og det reelle målet) viste. Mulig at gjennomsnittet av to målinger som kompenserer for tap når ultralyd-sensoren målte hjørnene av boksen. Alt i alt var ikke LEGO utstyret godt nok til å få helt presise målinger med metoden vi brukte, men det behandlede resultatet ga en god estimering.

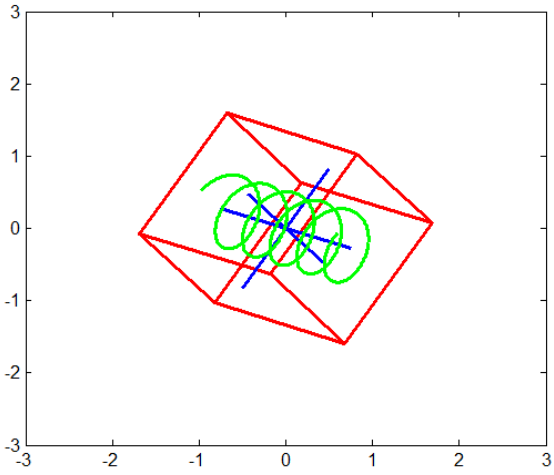
9 Program 8: Plotting av 3D kube i 2D plot: (Utført av Kjell)

Program: Gruppe1331_Program8_3Dkube.m

9.1 Problemstilling

Lage et program som tegner en kube i et plot. Anvende en rotasjons matrise til å rotere kuben om alle aksene. Bruker joystick som input for å styre rotasjonen.





Utdrag av kode

```
%% Initialiserer joystick
.
.
%% Initialiserer scalarer
xRotate  = 0.0;
yRotate  = 0.0;
zRotate  = 0.0;

%% While loop
while ~JoyStopSwitch
    joystick        = joymex2('query',0);       % spør etter data fra joystick
    JoyStopSwitch   = joystick.buttons(1);

%% Bruker aksene på Joystick til å rotere kuben.
    if (joystick.axes(1)/32768.0) > 0
        xRotate = xRotate - joystick.axes(1)/32768.0;
    end
    if (joystick.axes(1)/32768.0) < 0
        xRotate = xRotate - joystick.axes(1)/32768.0;
    end

    if (joystick.axes(2)/32768.0) > 0
        yRotate = yRotate - joystick.axes(2)/32768.0;
    end
    if (joystick.axes(2)/32768.0) < 0
        yRotate = yRotate - joystick.axes(2)/32768.0;
    end

     if (joystick.axes(4)/32768.0) < 0
        zRotate = zRotate + joystick.axes(4)/32768.0;
    end
    if (joystick.axes(4)/32768.0) > 0
```

```
        zRotate = zRotate + joystick.axes(4)/32768.0;
    end

%% Utskrift til skjerm
    fprintf(' X = %i , Y = %i,  Z = %i  \n', xRotate,yRotate,zRotate)

% Finner sinus og cosinus til akse vinklene når disse roteres
    % 0-360 grader.
    sx = sind(double(xRotate));  % xRotate = 0-360
    cx = cosd(double(xRotate));
    sy = sind(double(yRotate));  % yRotate = 0-360
    cy = cosd(double(yRotate));
    sz = sind(double(zRotate));  % zRotate = 0-360
    cz = cosd(double(zRotate));

    % Hentet informasjon fra:
    % http://www.songho.ca/opengl/gl_anglestoaxes.html
    %    Rx           Ry          Rz
    % |1  0   0| | cy  0 sy| |cz -sz 0|   | CyCz        -CySz          Sy  |
    % |0 cx -sx|*|  0  1  0|*|sz  cz 0| = | SxSyCz+CxSz -SxSySz+CxCz -SxCy|
    % |0 sx  cx| |-sy  0 cy| | 0   0 1|   |-CxSyCz+SxSz  CxSySz+SxCz  CxCy|

    rmx1 = cy * cz;
    rmy1 = -sz * cy;
    rmz1 = sy;

    rmx2 = cz * -sy * -sx + sz * cx;
    rmy2 = -sz * -sy * -sx + cz * cx;
    rmz2 = cy * -sx;

    rmx3 = cz * -sy * cx + sz * sx;
    rmy3 = -sz * -sy * cx + cz * sx;
    rmz3 = cy * cx;

    % Setter opp matrisen som danner "buret"
    %       x  y  z
    Bur = [-1 -1 -1
            1 -1 -1
            1  1 -1
           -1  1 -1
           -1 -1 -1
           -1 -1  1
            1 -1  1
            1  1  1
           -1  1  1
           -1 -1  1
           -1  1  1
           -1  1 -1
            1  1 -1
            1  1  1
            1 -1  1
            1 -1 -1];

    % Kalkulerer matriser med roterte "Bur" verdier
    X =  Bur(:,1) * rmx1 + Bur(:,2) * rmy1 + Bur(:,3) * rmz1;
    Y =  Bur(:,1) * rmx2 + Bur(:,2) * rmy2 + Bur(:,3) * rmz2;

    % Setter opp matrisen som danner "aksene"
    Akse = [-1  0  0
             1  0  0
             0 -1  0
             0  1  0
             0  0 -1
             0  0  1];

    % Kalkulerer vektorer med roterte "Akse" verdier
    AkseXx =  Akse(1:2,1) * rmx1 + Akse(1:2,2) * rmy1 + Akse(1:2,3) * rmz1;
```

```
    AkseXy =  Akse(1:2,1) * rmx2 + Akse(1:2,2) * rmy2 + Akse(1:2,3) * rmz2;
    AkseYx =  Akse(3:4,1) * rmx1 + Akse(3:4,2) * rmy1 + Akse(3:4,3) * rmz1;
    AkseYy =  Akse(3:4,1) * rmx2 + Akse(3:4,2) * rmy2 + Akse(3:4,3) * rmz2;
    AkseZx =  Akse(5:6,1) * rmx1 + Akse(5:6,2) * rmy1 + Akse(5:6,3) * rmz1;
    AkseZy =  Akse(5:6,1) * rmx2 + Akse(5:6,2) * rmy2 + Akse(5:6,3) * rmz2;

    % Bygger spiral, hentet ide fra læreboka
    % "MatLab for engineers" kapitell 5.4.1
    Xspiral = linspace(0,10*pi,10000);
    Yspiral = cos(Xspiral)*0.5;
    Zspiral = sin(Xspiral)*0.5;
    Xspiral = Xspiral/15-1;

    % kalkulerer vektorer med roterte "Spiral" verdier
    SpiralX = Xspiral * rmx1 + Yspiral * rmy1 + Zspiral * rmz1;
    SpiralY = Xspiral * rmx2 + Yspiral * rmy2 + Zspiral * rmz2;

    % Plotter vektorenee i samme plot
      plot(X,Y,'r','LineWidth',4)
      axis ([-3 3 -3 3])
      hold on
      plot(AkseXx,AkseXy,'b','LineWidth',4)
      plot(AkseYx,AkseYy,'b','LineWidth',4)
      plot(AkseZx,AkseZy,'b','LineWidth',4)
      plot(SpiralX,SpiralY,'g','LineWidth',4)
      drawnow
      hold off
end
```

9.2 Resultat

Programmet gir en illusjon av å se en tredimensjonal kube. Programmet bruker mye matrise håndtering. Det gir en økt forståelse av hvordan Sinus og cosinus henger sammen.

10 Konklusjon

Vi i gruppen har alle ulik bakgrunn, dette har ført til at vi alle har lært mye av hverandre.

Gjennom prosjektets gang, har vi vært opptatt av at alle skulle forstå programmene som ble utviklet. Dette har vi brukt forholdsvis mye tid på. I de individuelle oppgavene har alle gruppens medlemmer vært tilgjengelige for råd og veiledning når problemer og utfordringer oppstod. Dette har vært et interessant og utviklende prosjekt. Vi har fått øvet oss på samarbeid.

Lego NXT og Matlab har vært en genial kombinasjon. Det har vært lett å teste ut forskjellige ideer og fysiske sammenhenger med dette utstyret. Dokumentene som vi fikk til veiledning var veldig nyttige og lærerike. Integrasjon, derivasjon og filtrering har nå blitt visualisert på en god og intuitiv måte. Vi har nærmest blitt tvunget til å forstå sammenhengene.

Veiledning samtalene har vært nyttige. Vi har fått gode innspill, både utfordringer og forslag til løsninger. Samtalene har vært både lærerike og motiverende.

Arbeidet på laben har vært veldig sosialt. Vi har blitt bedre kjent med mange av våre medstudenter.

11 Referanser

[1] Lewis & Lofthus: "Java Software Solutions". 6th edition. Addison Wesley.

[2] http://www.songho.ca/opengl/gl_anglestoaxes.html

[3] H. Moore: "MATLAB for Engineers". 3rd edition. Pearson, 2013

[4] Wolfram Mathworld – Triangle Area. http://mathworld.wolfram.com/Triangle.html

[5] Wolfram Mathworld – Polygon Area. http://mathworld.wolfram.com/PolygonArea.html

[6] Darel Rex Finley "Polygon Area Algorithm With C". http://alienryderflex.com/polygon_area/

[7] #4 Numerisk integrasjon, numerisk derivasjon og filtrering.pdf. Its Learning – UiS

[8] PID kontroller: http://cs.brown.edu/~tld/courses/cs148/02/sensors.html

[9] DirectMotorCommand: http://www.mindstorms.rwth-aachen.de/documents/downloads/doc/version-4.03/help/DirectMotorCommand.html

[10] NXT toolbox Matlab: http://www.mindstorms.rwth-aachen.de/trac/wiki/Documentation

12 Vedlegg : Arbeidslogg

Dato	***Beskrivelse***	***Antall Timer***	***Hvem***
201309XX	Hentet Lego robot, telte opp delene, bygde roboten, testet software	3 3 3	Patrick Per-Ove Kjell
20131004	Laget programmet Gruppe1331_20131004.m Kjørte robot manuelt	4 3	Per-Ove Kjell
20131007	Laget programmet Gruppe1331_20131007_KR_Seint på kvelden.m Gjort ferdig integrasjon og filter samt plott av disse (Kreativt: NXT stopper ved høy lyd (for eksempel klapp)	4 4	Per-Ove Kjell
20131008	Laget programmet Gruppe1331_20131008.m Gjort ferdig numerisk derivasjon med plott. Fullført PI-regulator (mangler gode parametere) Kjørte robot automatisk	4 4 4	Patrick Per-Ove Kjell
20131009	Laget programmet Gruppe1331_20131009.m Gjort ferdig automatisk kjøring med valg av auto/manuell Plottet deler av PID-regulator. Plottet integrert avvik. Plottet akselerometer. Testet Ultralydsensor og Akselerometer. Sett på GUI.	4 4 4	Patrick Per-Ove Kjell
20131010	Laget programmet Gruppe1331_20131010.m Testet forskjellige parametere i PID-regulator Testet og dokumentert fremgang Skrevet statusrapport (Kreativt: NXT stopper når man holder foran ultralyd sensor)	4 4	Patrick Kjell
20131016	Påbegynt Strekkodeleser	1	Patrick
20131021	Laget programmet Gruppe1331_deloppg1_Strekkode_20131021.m Lest og tolket lys til binære verdier. Tolket verdier til brukbar data Oversatt Tolket Data til ASCII	5 4 4	Patrick Per-Ove Kjell
20131022	Laget programmene Gruppe1331_deloppg1_Strekkode_20131022.m Gruppe1331_deloppg2_Fart_20131022_PH.m Gruppe1331_deloppg3_Pappeskeareal_20131022_PH.m Dokumentert strekkode-program Begynt *veldig* smått med beregning av fart Plottet «rommet» ved hjelp av distanse og vinkel Gjort distanse og vinkel om til kartesiske koordinater Kode for lagre UL- distanse og vinkel fra motorC i vektorer.	4	Patrick
20131023	Laget programmene Gruppe1331_deloppg1_Strekkode_20131023.m Gruppe1331_deloppg2_Fart_20131023.m Gruppe1331_deloppg4_Musikk_20131023_PO Gjort ferdig og filmet Strekkodeleser Kodet foreløpig ide om beregning av fart, tid og strekning	3.5 6.5 3.5	Patrick Per-Ove Kjell

	Kodet foreløpig ide om avstandsregulator og musikkdans		
20131024	Laget programmene Gruppe1331_deloppg2_Fart_20131024 Gruppe1331_deloppg3_PappeskeAreal_20131024_PH Gruppe1331_deloppg4_Musikk_20131024_PO Fikk ok resultat når vi leste av farten, gjenstår å printe ut med medium avstand mellom svart og hvit og filme.	5 3 2	Patrick Per-Ove Kjell
20131025	Veiledningsmøte 2 Laget programmene Gruppe1331_deloppg5_HeaveComp_forskning_20131025PO.m Gruppe1331_deloppg4_Musikk_20131025_PO.m Ryddet opp i deloppgave 4 og forklarte regulatoren - Dette kan videreføres til andre prosjekter Bygde om en motor så den kan brukes som vinsj Brukte IMU- tilt verdi til å kjøre hastigheten på motor A Lagde program som logget måledata fra lyssensor til vektor for Lysverdier og Tid . Verdiene lagres i en mat fil for videre bearbeiding mtp. Integrering, Derivasjon og Filtrering. Lagde et program for Integrering og et for Derivasjon. Produsere plot og klippet disse sammen inn i et dokument som kan brukes i sluttrapporten	1 7 6	Patrick Per-Ove Kjell
20131026	Laget programmene Gruppe1331_deloppg5_HeaveComp_forskning_20131026PO.m Regulerer motorA.power mot motorA.TachoCount, med IMU som setpoint. Funker tålig bra, trenger litt mer jobb.	3,5	Per-Ove
20131027	Laget programmene Gruppe1331_deloppg6_Segway_20131027_PO.m Regulerer motorA.power med IMU som error. Deriverer error og lar pådraget øke proporsjonalt med en ny og høyere faktor. Kan forskes mer på!	6	Per-Ove
20131029	Laget programmene Gruppe1331_deloppg6_Segway_20131029_PO.m Prøvde meg på pendelregulering, men det ble mest bygging av lego og prøving av forskjellige parametre til regulatoren.	2	Per-Ove
20131030	Laget programmene Gruppe1331_deloppg3_PappeskeAreal_20131030_PH Gruppe1331_deloppg6_Segway_20131029_PO.m Bygget om NXT og testet areal-programmet. Testet forskjellige parametere for motorC. Får ujevn rotasjon (mye pga. Kabel). OK med 180 målinger på 2° . Lagret målte data og sammenlignet målt areal med virkelig areal. Manuell arealberegning trenger mer arbeid. Gir negativ verdi. PO: Bygde en bedre pendel som måler avviket til loddrett posisjon med tachocount og IMU. Hvis avviket er for stort i	5.5 7 1	Patrick Per-Ove Kjell

	forhold til hastighetsbegrensning på motor, kjører pendel (motorC) med liten kraft mot senter. 1.prioritet er å få bra regulering på ved relativt lite avvik.		
20131031	Laget programmene Gruppe1331_deloppg3_PappeskeAreal2_20131031_PH Skrev om programmet til å kjøre en kontinuerlig runde og hente data, istedenfor grad for grad vha TachoCount. Kjørt tester og lagret data til arealberegning. Innebygget arealberegning til ca reelt areal.	5 2	Patrick Kjell
20131102	Laget programmet Gruppe1331_deloppg3_PappeskeAreal2_20131102_PH Dokumentert kode Implementert to metoder for arealberegning (sektorer og koordinater)	3	Patrick
20131104	Laget programmet Bygget LEGO-modell av heave-kompensatoren. Kodet «heave-generator-ting» Regnet ut vinkel, fart og bruker dette til å kompensere med en annen motor. Konstruerte maskin i Lego Designverktøy hjemme. Vi får se om vi har nok lego til å få satt den sammen. Denne maskinen sørger for at det kun blir vertikal bevegelse på loddet.	2 2 5	Patrick Kjell Kjell
20131105	Bygde maskinen i henhold til konstruksjonen i designverktøyet. Måtte bruke litt andre legoklosser pga. begrenset utvalg i NXT kittet. Testet softwaren filmet og tok bilder.	4 4	Patrick Kjell
20131106	Startet på prosjektrapporten. Laget programmene Gruppe1331_deloppg6_Segway_1Motor_20131106_PO Gruppe1331_deloppg5_Pendel_utenMotorC_20131106_PO Brukte tid til å se på egenskapene til hvert ledd i regulatoren. La inn lyssensor som settpunkt til regulatoren	2 2 6	Kjell Patrick Per Ove
20131108	Veiledningsmøte Begynt med videoredigering	1 1 1	Kjell Patrick Per Ove
20131109	Jobbet med sluttrapport og ferdigstilt program for innlevering Jobbet videre med Gruppe1331_deloppg3_Areal.m	6 4	Kjell Patrick
20131110	Jobbet med sluttrapport og ferdigstilt program for innlevering Disse filene er klar nå: Gruppe1331_deloppg1_Strekkode.m Gruppe1331_deloppg2_Fart.m Gruppe1331_deloppg7_HeaveComp.m Gruppe1331_DeriveringAvLysVerdi.m Gruppe1331_IntegreringAvLysVerdi.m Gruppe1331_LesInnLysVector.m	8 5	Kjell Patrick
20131111	Jobbet med sluttrapport og ferdigstilling av program. Gruppe1331_deloppg4_Musikk_20131112_PO	8 6	Patrick Per-Ove

	Jobbet Gruppe1331_rotate.m	4	Kjell
20131112	Jobbet Gruppe1331_rotate.m Jobbet med å ferdigstille program for innlevering Gruppe1331_deloppg4_Musikk_20131112_PO	3 5 3	Kjell Patrick Per-Ove
20131113	Jobbet med dokumentering og ferdigstilling av program for innlevering Jobbet Gruppe1331_rotate.m	6 3.5 9	Patrick Kjell Per-Ove
20131114	Jobbet med dokumentering og ferdigstilling av prosjekt.	4 5	Patrick Per-Ove
20131115	Ferdigstilling av alle program og rapport. Levert prosjekt.	10 10 6	Patrick Per-Ove Kjell