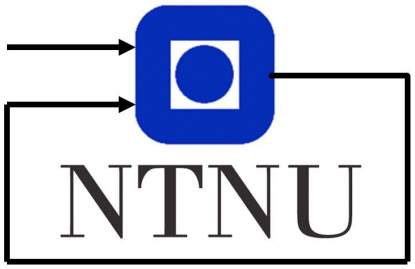
TTK4100: Legolab

Gyroskop og stabilisering av ståhjuling

September 2015



- Godkjenning av studass i timer for øvingsveiledning.
- Godkjenning krever fremvisning av ståhjuling og innlevering av svar på teorispørsmål.
- Godkjenning av denne oppgaven er obligatorisk for å gå opp til eksamen.

1 Introduksjon

Denne laboppgaven består av å bygge en ståhjuling (segway) av LEGO Mindstorms. På lik linje med teknostart skal denne ståhjulingen programmeres med MATLAB og Simulink. En ståhjuling er et eksempel på et ustabilt system. Ved hjelp av en tilbakekoplings-sløyfe skal vi se at dette kan stabiliseres slik at den holder seg oppreist. Dere vil lære å implementere en slik regulator i praksis. I tillegg vil dere lære å takle utfordringer ved bruk av sensorer som er utsatt for *bias* og *drift* slik som en gyrosensor. I denne oppgaven trenger dere ikke bruke mye tid på å bygge en egen robot; det følger med en designoppskrift dere skal følge. Før dere begynner å bygge skal dere lese igjennom teorien.

2 Grunnleggende Teori

2.1 Vinkelmåling med fast referanse

Ofte har vi tilgang til et fast referansepunkt som vi skal måle vinkelutslag i forhold til. Eksempel på dette er:

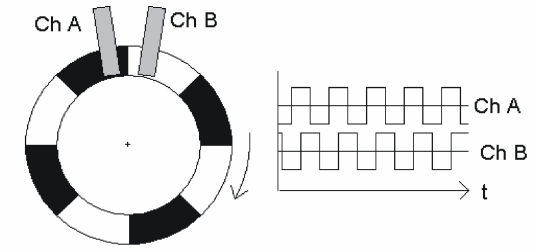
- Speedometer på bil, hjulets vinkelhastighet relativ til bilen
- Vinkelutslag på robotledd.
- Bevegelse til datamus

En rekke metoder er utviklet for å måle vinkelutslag eller vinkelhastighet i forhold til et fast referansepunkt. Potensiometre, resolvere og tachometer har alle vært mye brukt tidligere. Billig digital elektronikk, ønske om presisjon, og ikke minst ønske om digitale signaler, har imidlertid ført til at nevnte metoder i stadig større grad er blitt erstattet av encodere (pulsgivere).

Encodere Encodere består av en roterende kodeskive som avleses med en eller annen form for nærhetsdetektor. Tidligere ble ofte børster som gled over en kontaktskive benyttet. Figur 2.2 viser en slik mekanisk implementasjon. Begrenset oppløsning og kort levetid som følge av mekanisk slitasje har ført til at man i dag nesten utelukkende benytter optiske eller magnetbaserte nærhetsdetektore til å lese av kodeskiven. Vi kan i utgangspunktet skille mellom to typer encodere; inkrementelle og absolutte. **Den inkrementelle encoderen** består av to detektorer (kanaler). Når kodeskiven roterer gir disse ut hvert sitt pulstog med en frekvens proporsjonal med vinkelhastigheten. Ved å plassere detektorene som i figur 2.2, vil pulstogene ha en innbyrdes faseforskjell på 90°. Ved å se på fortegnet til denne faseforskjellen, kan man bestemme rotasjonsretningen.

Oppløsningen Oppløsningen i en slik encoder er gitt av hvor mange logiske skift pulstogene gir i løpet av en omdreining. Kodeskiven er vist i figur 2.2, med 4 pulser pr. omdreining og 8 logiske skift pr kanal. Målt over begge kanalene får vi 16 logiske skift pr. omdreining, som gir en oppløsning på $360°/16 = 22.5°/skritt$.

Figur 2.1: Inkrementell optisk encoder



Figur 2.2: Prinsipp for 2-kanals inkrementell encoder

Avlesning Frekvensen til pulstoget er proporsjonal med vinkelhastigheten og med oppløsningen på kodeskiven (pulser pr. omdreining). For at målingen med encoderen skal bli korrekt, må kanalene avleses (samples) minst en gang pr. logisk skift. Dersom kodeskiven i figur 2.2 roterer med 1 omdreining i sekundet, må signalet avleses minst 16 ganger i sekundet. Når vinkelhastighet og oppløsning øker, må ofte avlesning av encodere utføres i egen hardware, som tar seg av telling av pulser.
For å kunne bestemme absolutt vinkel, har ofte inkrementelle encodere et tredje spor (nullspor), som benyttes til å indikere nullpunktet til vinkelmålingen. Ved oppstart, eller ved feil i pulstelling som følge av støy eller for høy hastighet, kan absolutt posisjon kalibreres ved passering av nullpunkt.

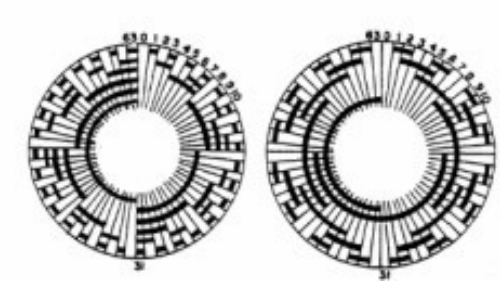
Figur 2.3: Absolutt, 4 kanals, binæ r-kodet encoder med glidekontakter

Den absolutte encoderen, eksemplifisert i figur 2.3, gir mulighet til å lese av vinkelen direkte, ved at hvert enkelt skritt gir et unikt signal på encoderens utgang. Dette medfører at encoderens oppløsning øker eksponentielt med antall kanaler.

Oppløsningen i grader er gitt som $360°/2n$, hvor n er antall kanaler. Eksempelvis vil en 16-kanals absolutt encoder som på figur 2.4, kunne kode 2 ˆ16 = 65536 skritt, som gir en oppløsning på 0.0055°.

Avlesning. I motsetning til den inkrementelle encoderen, som beregner vinkel ved å hele tiden telle pulspasseringer, kan vinkelen avleses direkte fra den absolutte encoderen. I overgangen mellom to skritt, vil unøyaktigheter i avlesningen føre til at verdiene på kanalene ikke skifter nøyaktig samtidig. Dersom kodehjulet kodes med vanlig binærkode, vil man ved gal avlesning av det mest signifikante bit (MSB) få en feil i vinkelmålingen på hele 180°. For å unngå dette benyttes såkalt gray-kode, hvor koden til to inntilliggende verdier alltid avviker med kun ett bit. En gal avlesning vil derfor aldri avvike med mer enn en *minst* signifikant bit (LSB), som gir en mye mindre feil[1]. Konvertering fra avlest gray-kode til vanlig binærkode kan utføres effektivt i hardware eller software ved eksklusiv-eller (XOR) operasjoner(se figur 2.6). Siden vinkelen kan leses direkte av den absolutte encoderen, kreves det ikke at signalet avleses for hvert logiske skift.

Figur 2.4: Absolutt 16-bits, gray-kodet encoder med optisk avlesning



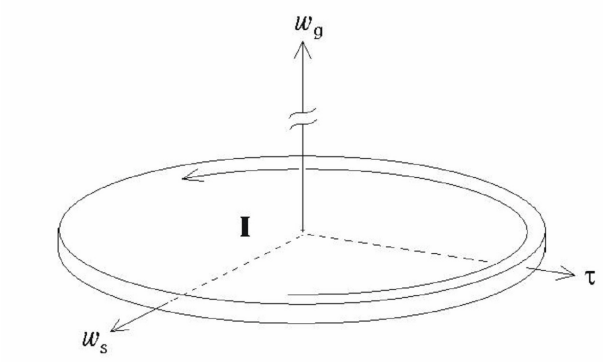
Figur 2.5: Kodehjul med vanlig binærkode til venstre, og med syklisk binærkode (gray-kode) til høyre

[1] XOR-operasjoner, MSB og LSB er tema som vil dukke opp senere fag, og det forventes ikke at disse uttrykkene er kjent på dette stadiet.

2.2 Vinkelmåling uten fast referanse

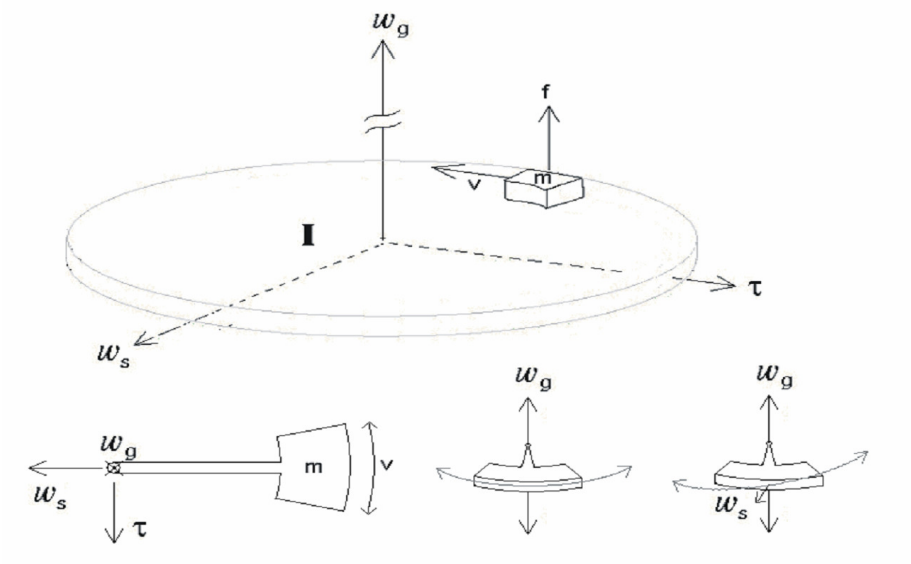
Noen ganger må vi kunne måle vinkel uten å ha tilgang på noe fysisk fast referansepunkt. For autopilotsystemet i et fly for eksempel, vil det være essensielt å ha en god måling av flyets vinkler (yaw-pitch-roll). Dette utføres av såkalte treghetsnavigasjons-systemer (INS) som ved hjelp av å måle vinkelhastighet og lineær akselerasjon langs tre akser, kan estimere fartøyets retning, hastighet og posisjon, med en mer eller mindre god nøyaktighet. Basiskomponentene i et slikt system er akselerometeret, som måler lineær akselerasjon, og gyroskopet, som måler vinkelhastighet om en akse. Tidligere var dette utelukkende mekanisk komplekse og svært kostbare instrumenter, med begrenset anvendelse hovedsakelig innen luftfart. De siste årene har imidlertid moderne produksjonsteknikker og masseproduksjon, åpnet en rekke nye markeder for slike systemer. I dette labheftet vil bare gyroskopet bli beskrevet.

Gyroskop Det finnes flere måter å konstruere gyroskop på. Vi skal her se nærmere på noen av de mest vanlige metodene. Dette temaet dekkes også i senere fag, som Navigasjonssystemer. **Roterende masse** Det mekaniske gyroskopet ble oppfunnet i 1852 av Léon Foucault, og det benytter seg av prinsippet om bevaring av vinkelmoment til å kunne måle statisk vinkelhastighet. Dersom man som i figur 2.12 har en skive med treghetsmoment I omkring en akse som den roterer om med vinkelhastighet ω_g, vil den for å rotere om en akse normalt på denne ω_s måtte utsettes for et statisk dreiemoment τ normalt på både ω_g og ω_s. (På vektorform kan dette skrives $\tau = I(\omega_g \times \omega_s)$) Rotasjon av gyroen normalt på spinn-aksen fører derfor til et målbart dreiemoment, proporsjonalt med vinkelhastigheten. Spinnende-masse-gyroer er med sine roterede deler lite robust og dårlig egnet til miniatyrisering. På den andre side er de svært presise, og benyttes fortsatt i en del navigasjonssystemer.



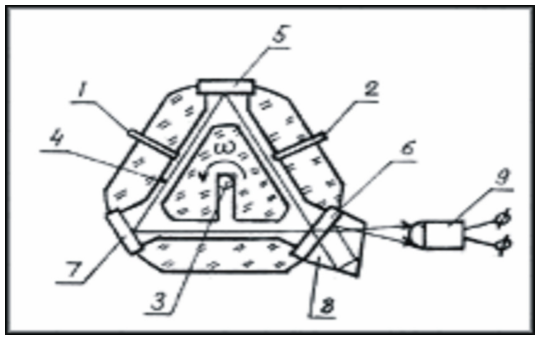
Figur 2.12: Gyro-effekt for roterende skive

Vibrerende masse I systemer med noe lavere krav til presisjon kan man ved å erstatte svinghjulet med en vibrerende masse redusere pris og størrelse betraktelig. Prinsippet er egentlig det samme (anta at man skjærer en liten del av den roterende skiven, og ser på denne individuelt), men siden man antar lineær hastighet i et rotert system, kalles den resulterende kraften ofte Coriolis-kraft. Corioliskraften kan uttrykkes som $f = 2mv \times \omega_s$. (Coriolis-kraft og ligninger på denne formen vil dukke opp i senere fag, som for eksempel Fysikk.) Også her er resultatet en målbar kraft proporsjonal med vinkelhastighet. Hastigheten v er gitt av hastigheten til en resonant vibrerende masse. Den spesifikke resonansfrekvensen (størrelsesorden 5-20 kHz), gjør disse gyroskopene spesielt følsomme for mekanisk støy på denne frekvensen.



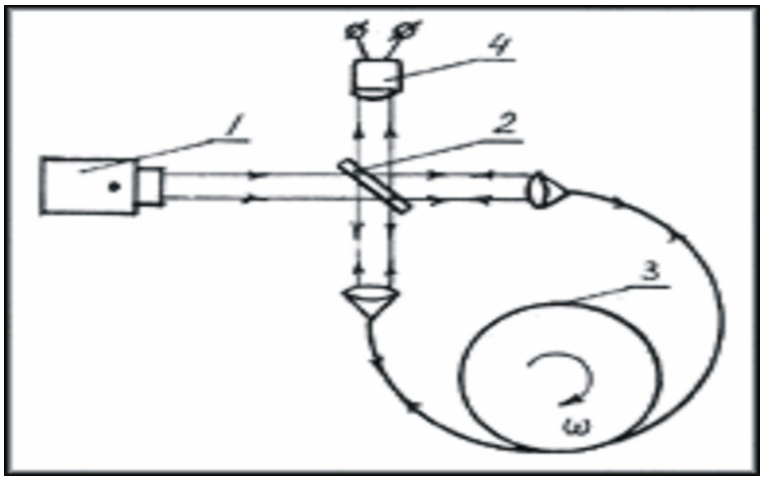
Figur 2.13: Coriolis-kraft for objekt i bevegelse i et rotert system

Optiske gyroskop I systemer med ekstrem krav til nøyaktighet og driftsikkerhet, som for eksempel undervannsnavigasjon, avionikk etc., benyttes ofte optiske gyroskop. **Ringlaser-gyroskop** består av en ring eller trekantformet laserkavitet som vist i figur 2.14. En slik konstruksjon kan gi opphav til to laserstråler; én som reflekteres med klokken og én mot klokken. Dersom gyroskopet ikke roterer omkring aksen normalt på figuren, vil begge strålene oppfatte distansen rundt som like lang, og får samme bølgelengde / frekvens. Når strålene mikses sammen gir de derfor opphav til en konstant lysintensitet. Dersom gyroen derimot roteres i retningen ω (mot klokken) vil strålen som går mot klokken oppfatte distansen rundt som litt lengre, og få større bølgelengde, mens strålen med klokken oppfatter lengden som kortere, og får kortere bølgelengde. Den resulterende forskjellen i frekvens, vil gi opphav til interferens (sveving) i den miksede lysstrålen. Lysdetektoren oppfatter derfor et oscillerende lys, med frekvens proporsjonal med vinkelhastighet.



Figur 2.14: Ringlasergyro. 1 og 2 : anoder, 3 : katoder, 4 : gassutladningskavitet, 5, 6, og 7 : speil, 8: strålesamler, 9 : lysdetektor. (National Instrument developer zone)

Fiberoptisk gyro består av en lang fiberoptisk kabel (opp til 5km) spolt opp i en sirkel. En laserstråle splittes i to og sendes hhv. med og mot klokken gjennom kabelen. Dersom kabelspolen ikke roterer, vil gangtiden for lyset gjennom spolen være den samme i begge retninger. Laseren kommer med samme fase ut fra begge retninger, og gir full konstruktiv interferens. Dersom spolen derimot roterer, vil den ene laserstrålen måtte gå lenger enn den andre. Dette resulterer i en faseforskyvning innbyrdes mellom strålene. Faseforskyvningen er proporsjonal med vinkelhastigheten, og kan måles (ved interferometri) for å bestemme rotasjonshastighet.



Figur 2.15: Fiberoptisk gyro. 1 = laserkilde 2 = strålesplitter 3 = spole med optisk fiber 4 = lyssensor. (figurer fra National Instrument developer zone)

3 Lego-settet

LEGO settet dere får utdelt til denne oppgaven inneholder flere deler:

- Byggeklosser til roboten.
- NXT kloss, datamaskinen som styrer motorene og leser fra sensorene.
- Tre motorer, kan styres av NXTen.
- Ulike sensorer, i denne oppgaven skal dere bruke gyroskopet.
- Ledninger og batterilader.

3.1 Oppløsning på encodere og sensorer

Oppløsningen på encoderne og sensorene er alle på 1 grad. Det betyr at feilen kan være like stor som denne størrelsen. For eksempel, kan den egentlige verdien være 0.49, mens den verdien målt av encoderen vil være 0.

3.2 Gyroskopsensor

Gyroskopet som kommer i LEGO settet er en XV-3500CB fra Epson. Dette er en variasjon av vibrerende masse gyroskop, der den vibrerende massen er et quartz krystall som produserer en elektrisk spenning når den blir utsatt for krefter. Denne spenningen måles, og blir dermed et mål på vinkelhastigheten.

4 Ståhjuling

En ståhjuling kan ses på som en invertert pendel. Dette er et ustabilt system som vil falle over ende om den blir overlatt til seg selv. En tilbakekoplingssløyfe er nødvendig for å kunne stabilisere ståhjulingen.
I en modell av ståhjulingen kan vi regne med fire tilstander,

1. θ og $\dot{\theta}$: Vinkel og vinkelhastighet.
2. x og $\dot{x}$: Lineær posisjon og hastighet

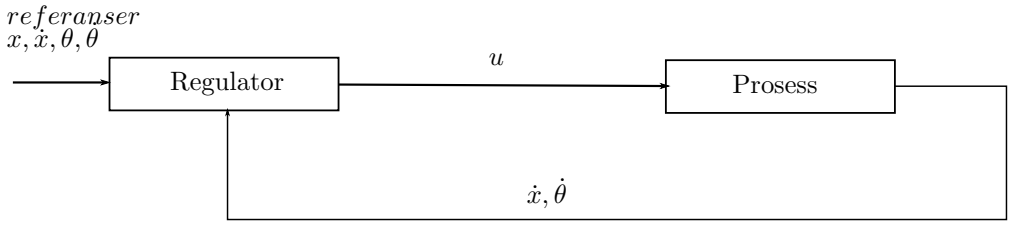
Vi ønsker å styre θ til 0 grader (eller 180 avhengig av hvordan man definerer referansesystemet), mens vi ønsker å styre x mot en kommandert posisjon. Kun vinkelhastighet $\dot{\theta}$ fra gyrosensoren og lineær hastighet $\dot{x}$ fra enkoderne er målte signaler. De to andre tilstandene kan dog finnes fra disse.

Vi skal bruke en PI-regulator med vinkelhastighet som referanse. I tillegg bruker vi en PI-regulator med hjulhastighet som referanse. Figur 1 viser det totale systemet.

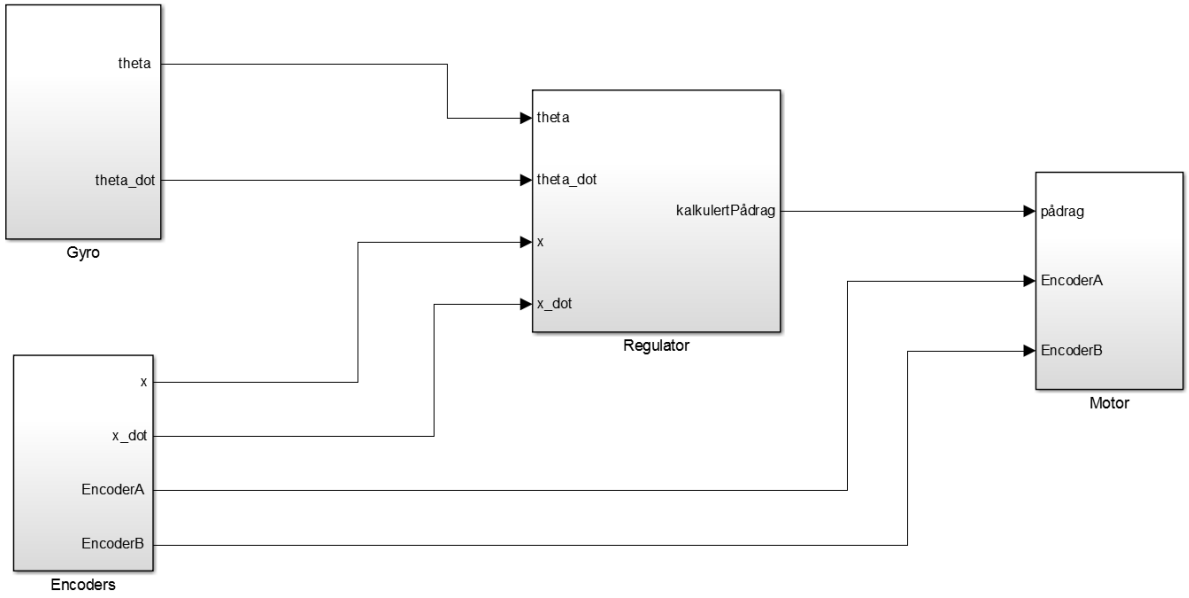
a) Tegn en enkel ståhjuling-illustrasjon med tilstandene x, $\dot{x}$, θ og $\dot{\theta}$.

5 Installasjon av MATLAB

Dersom dere har MATLAB med fungerende Lego Mindstorms NXT Toolbox for Simulink fra Teknostart er dette steget ikke nødvendig. For dere som ikke har hatt det, se installasjonsguiden fra Teknostart. Den ligger på it's learning og hjemmesiden til Teknostart for Kybernetikk og Robotikk.



Figur 1: Ståhjulingen sitt reguleringssystem



Figur 2: Hele programmet delt opp i subsystemer

6 Oppgaven

Når dere har fått i gang MATLAB og testet at dere får til å laste et program over på NXT'en, er det på tide å starte med selve oppgaven. I løpet av oppgaven skal spørsmål besvares. Svar til teorioppgavene skrives på et ark og vises til stud.ass ved godkjenning av lab.

6.1 Balansere en ståhjuling

Ståhjulingen skal programmeres i Simulink for MATLAB, slik dere gjorde i Teknostart. På side 14 ligger noen tips til simulink.

Styringssystemet kan deles opp i fire deler: gyro, enkodere, motorer og regulator, se Figur 2. Gyro- og enkoderdelen skal hente målinger, som behandles i regulatordelen, før det blir matet til motorene.

6.1.1 Bygg roboten

a) Det første man må gjøre er å bygge roboten i henhold til guiden som ligger vedlagt denne laboppgaven.

6.1.2 Gyroskop

a) Til å starte med skal vi skaffe målingene fra gyroskopet. Skriv ut verdien fra gyroskopet til LCD skjermen, last dette programmet opp til NXTen, og beveg på gyroskopet. Prøv å svare på spørsmålene:

Spørsmål: Hvilken verdi måler gyroskopet?

- Hva finner man ved derivasjon av gyroskopmålingen?
- Hva finner man ved integrasjon av gyroskopmålingen?

Hint: LCD-blokken kan brukes til å printe ut verdien til skjermen på NXTen. Dette er nyttig i testing og feilsøking.

Kalibrering av gyroskop Som dere sikkert har lagt merke til er ikke gyroskopet nødvendigvis kalibrert, altså at gyroskopet ikke viser null når det ligger i ro. For å kalibrere gyroskopet kan man lese av verdien som vises når gyroskopet ligger i ro, og summere denne verdien med offset-verdien som står i gyroskopblokken i simulink, før man laster over programmet til NXT brikken en gang til.

b) Kalibrer sensoren ved å trekke fra verdien den leser når den ligger i ro. Last opp et program med ny offsetverdi og bli sikre på at gyroskopet gir ut riktig verdi.

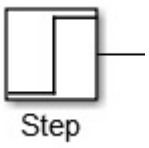
Spørsmål: Dersom man integrerer en ukalibrert sensor over lengre tid vil det føre til *drift* i målingen, hva tror du dette betyr?

Hint: Integratorblokken ønsker en double (desimaltall) verdi inn, mens gyroskopblokken gir ut en int16 (heltall) verdi. Dere må huske å konvertere signalet slik at verdiene blir korrekte.

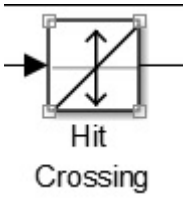
Automatisk kalibrering Det kan hende dere må rekalibrere sensorene flere ganger underveis, spesielt hvis det er en stund siden forrige gang dere kalibrerte den. For å slippe å gjøre dette manuelt flere ganger, kan NXTen programmeres til å lagre riktig offsetverdi selv.

c) Finn en måte å automatisk lagre offsettet på og trekk det fra gyroskopmålingen.

Hint: Ved å integrere verdien fra gyroskopet i et visst tidsintervall og deretter dele på tidsintervallet, får man en gjennomsnittlig verdi. Denne verdien kan lagres i en memory blokk og deretter trekkes fra gyroskopmålingen for å oppnå riktig kalibrering. Det er mange måter å implementere dette, en *Step*-blokk i kombinasjon av *Hit Crossing*-blokk kan være nyttig for å sende ut en puls etter et visst tidsintervall. Se Figur 3 og 4.



Figur 3: Endrer utgangen ved t_1



Figur 4: Hit Crossing-blokk

Oppløsning som feilkilde Gyroskopsensoren gir ut måling som heltall av typen *int16*. Dersom absoluttverdien til vinkelhastigheten som gyroskopet måler er mindre enn 1, vil gyroskopet runde av til nærmeste heltall. Dette gjør at gyroskopsensoren har en bias mot en retning. Konsekvensen er at vinkelmålingen kan drifte.

Spørsmål: Hva er forskjellen mellom feil fra støy og feil fra en ukalibrert sensor?

En metode for å hindre at vinkelmålingen drifter er å gi integratoren, som integrerer vinkelhastighet til vinkel, verdien 0 når absoluttverdien av korrigert måling er mindre enn 1.

d) Lag en løsning som hindrer integratoren fra å summere feilen som beskrevet ovenfor.

Hint: En *Switch* kan sende gjennom to forskjellige signal, avhenging av verdien til et tredje signal.

For å holde simulinkmodellen ryddig lager vi et subsystem for hver del. For å sende signaler inn og ut av subsystem må man bruke *Input*- og *Output*-blokker. Disse finner man i biblioteket *Commonly Used Blocks* sammen med subsystemblokken. Dobbeltklikk på en subsystemblokk for å gå inn i den, da vil innholdet i subsystemet vise seg. Trykk *ESC* for å gå ut av subsystemet.

e) Marker alle blokkene i simulink, høyreklikk og klikk *Create subsystem*. Sørg for at signalene for vinkelhastigheten og vinkelen går inn i *Output*-blokker slik at de er tilgjengelige på utsiden av subsystemet.

Forsikre dere om at offset blir trukket fra riktig. Dette er veldig viktig. Har man feil offset så kan man få mange unødvendige og frustrerende problem senere.

6.1.3 Enkodere

I tillegg til målingen fra gyroskopet kan enkoderne i motorene brukes til styring. Dersom roboten beveger seg horisontalt er dette en indikasjon på at den faller i kjøreretningen. Dersom pådraget fra gyroskopmålingene ikke klarer å rette opp roboten vil den kjøre med konstant fart, eller til og med falle. Motorhastigheten kan derfor brukes som pådrag for å kompansere for dette. Til å starte med er enkoderne konfigurert til å telle antall grader siden programmet ble startet. Ved å gå inn i enkoderblokken og endre reset-mode til *Reset at each sample time*, vil enkoderen gi ut grader siden sist sample. Ved å dele denne verdien på sampletiden får man grader per sekund.

a) Sett inn en enkoderblokk for hver av motorene i subsystemet til enkoderne. Konfigurer disse til å resette ved hver sample.

b) Send gjennomsnittet av motorhastigheten ut av subsystemet gjennom Output blokker.

c) For å stabilisere ståhjulingen bedre vil vi og bruke motorposisjonen, altså hvor langt motoren har kjørt. Dette hjelper og når man skal kjøre ståhjulingen bakover eller framover. Lag en ny utgang fra enkodersubsystemet som sender ut motorposisjonen. *Hint:* Integrasjon.

6.1.4 Motorer

Subsystemet for motorene er ganske enkelt, det skal motta pådraget fra regulatorsubsystemet.

a) I motorene sitt subsystem, legg inn en Saturation blokk før motorene, for å sørge for at motorene ikke får større verdi enn de klarer å kjøre. Den største verdiene motoren klarer er -100 eller 100.

b) Her kan det og være aktuelt å legge inn en cruise-control funksjon for motorene, slik som dere gjorde i teknostart. Bruk guiden fra teknostart som hjelp. NB: Dette er ikke nødvendig for å balansere ståhjulingen. Man skal kunne få en ganske god ståhjuling uten å gjøre dette leddet. Det kan dog være lettere å få ståhjulingen til å svinge eller å kjøre rett fram.

6.1.5 Regulator

I regulatoren skal selve styringen ta plass. Alle målingene fra gyroskopet og enkoderne skal være innganger i regulatorsubsystemet. Den eneste utgangen skal være pådraget, som går inn til motorsubsystemet.

a) Implementer regulatorene for gyro- og enkodermålingene.

b) Finn riktige parameter for regulatorene, slik at den står av seg selv.

I-leddet i PI-regulatoren(gyro): Dette leddet består greit av en konstant ganget med endring i vinkel fra nullposisjon. Et enkelt eksempel: dersom konstanten er 10 og endringen fra nullposisjonen er 3 grader så blir bidraget 30.

P-leddet i PI-regulatoren(gyro): Dette leddet består av en konstant ganget med vinkelhastigheten målt i [deg/s]. Dette gir et bidrag til pådraget da roboten begynner å falle fort nedover, men stopper da den står i ro. Skulle roboten være på veg "oppover", vil dette bidraget hjelpe med å få bidraget mindre slik at man ikke kommer over nullposisjonen. Et praktisk eksempel: Roboten begynner å falle fremover og du har en positiv vinkelhastighet. Du gir da på et positivt pådrag, og prøver å rette roboten opp. Du kommer til punktet den ikke faller lenger, men fortsatt er, si, 3 grader feil fra nullposisjonen. Da bidrar I-leddet slik at du fortsatt gir et pådrag, men nå vil jo vinkelhastigheten være negativ! Bidraget fra P-leddet vil da være negativt og gjøre pådraget mindre. Dette lager en form for dempeeffekt. Men man vil jo ikke at dempingen skal være så stor at man ikke kommer seg opp ... ;)

P-leddet i PI-regulatoren(enkoder): Robotens horisontale hastighet er tett knyttet opp mot vinkelhastigheten på motorene. Dersom roboten beveger seg indikerer dette at den lener seg mot den siden. Da ønsker man som oftest å gi ekstra pådrag til motorene, slik at den retter seg opp mot referansen sin. Eksempelvis vil en robot uten P-regulator for motorhastighet kunne kjøre i konstant fart i en retning, på grunn av at bidraget fra gyroskopmålingene ikke gir nok pådrag. Med P-regulator med motorhastigheten vil den konstante farten gi opphav til mer pådrag slik at roboten får rettet opp vinkelen sin. Dette gjør også at roboten blir mer robust mot at vinkelreferansen er feil.

I-leddet i PI-regulatoren(enkoder): Dette leddet øker lineært med hvor langt motoren har kjørt. En runde for motoren er 360 grader, to runder er 720 grader, osv. På den andre siden er en runde bakover -360, to runder -720 grader osv. Med andre ord, dersom du kjører fram 180 grader, og så bak igjen 180 grader, vil verdien her være 0. Dette vil hjelpe mot eventuell drift i gyrosensoren. Dersom nullvinkelen drifter mot en side vil dette i første omgang føre til at ståhjulingen vil rette seg mot og kjøre i den retningen. Dette gir utslag på den lineære posisjonen og skaper et pådrag som motvirker driften.

Hint: Man må tilpasse parameterverdiene etter hvilke enheter man velger å bruke. Parameterene burde ligge i følgende område når Gyro-blokka gir ut verdier oppgitt i grader og grader/s, og Enkoder-blokka gir ut signal med enhet meter og meter/s:

- 5-15 for vinkelen.
- 0.5-2 for vinkelhastigheten.
- 200-350 for lineær posisjon.
- 350-500 for lineær hastighet.

Dersom cruise-kontrolleren fra teknostart er brukt må parameterene skaleres. I stedet for å ta en referanse mellom -100 og 100, vil en cruise-kontroller ta for eksempel rotasjoner per sekund som referanse. Typisk vil ikke ståhjulingen sine motorer klare stort mer enn 3 rotasjoner per sekund. Dette tilsvarer en skalering på $u_{skalert} = 0.03u$.

Her er det litt prøving og feiling som må til. Systemet er ustabilt, så det er vanskelig å bruke kjente metoder slik som Ziegler–Nichols eller Skogestads.

Spørsmål: Hvordan kan man bruke de nåværende sensorene til å finne ut hvor langt man har kjørt? Kunne dette blitt brukt til å lage en bedre regulator?

6.2 Kjøre en ståhjuling

Nå som ståhjulingen klarer å stå er det på tide å få den til å kjøre fremover.

a) For å få ståhjuligen til å kjøre kan man endre vinkelreferansen til enkoderen.

b) Finn en måte å få roboten til å svinge, mens den holder balansen.

Hint: Se vedlegg B.

6.2.1 Konkurranse

For å motivere til å lage den beste ståhjulingen vil en *frivillig* test kjøres ved godkjenningen av laben. Testen går ut på å kjøre en forhåndsbestemt rute på kortest mulig tid. Testen er som følger:

1. Kjør en meter fram
2. Snu 90 grader til venstre og kjør en meter til
3. Rygg tilbake samme rute

Feilmarginer for å få godkjent tid er $\pm 10^\circ$ på vinkelen og ± 10cm på hver av avstandene.

7 Godkjenning

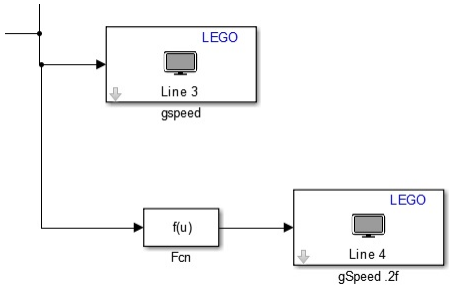
For å få godkjent lab skal fungerende ståhjuling etter seksjon 6 vises fram til studass. I tillegg må man vise fram svar på teorispørsmål, og til slutt pakke sammen og levere lego-settet.

Laboppgavene er nye fra og med i år. Dersom du har tilbakemeldinger om laben, lever gjerne disse til studass ved godkjenning. Dersom du lurer på noe, spør på diskusjonsforumet på it's learning eller send en epost til sondrewb@stud.ntnu.no eller magho@stud.ntnu.no.

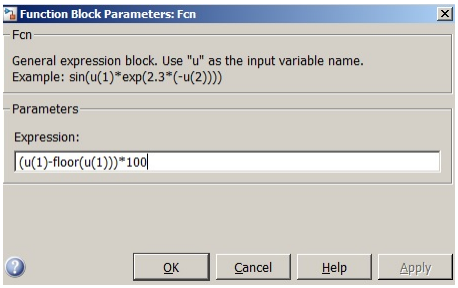
Vedlegg

A Printe ut desimaltall

Som sagt så er det veldig lurt å printe ut mange av verdiene på skjermen underveis. Dette gjør det mye lettere å debugge, så det kan spare deg for både tid og krefter. Et problem her er at man kan bare printe ut heltall på NXT-skjermen, mens verdiene i simulink er flyttall. For å komme rundt dette problemet kan man kjøre signalet igjennom en function-blokk, som i eksempelt under.

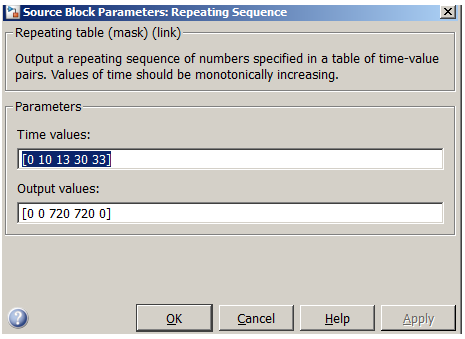


Her tar function-blokka og trekker signalet fra signalet nedrundet. Dette gjøres ved hjelp av gulvfunksjonen. Deretter gangest det med 100 for å få tallet som heltall. Ut kommer signalet inn sine 2 første desimalplasser. Med andre ord, har man et signal inn på 17.29 så kommer 29 ut. Sender man et signal på 78.557 så kommer 56 ut.



B Repeating Sequence blokken

I Simulink finner man en blokk *Repeating Sequence*. Denne blokken kan være til god hjelp når man skal få roboten til å kjøre fremover eller bakover. Under Time Values parameteren i blokken spesifiserer du, i sekund, hvor lenge det skal gå før signalet sender ut verdiene i Output Values.



Her vil det sendest ut 0 etter 0 s, fortsatt 0 etter 10 s, 720 etter 13 s , fortsatt 720 etter 30 s osv. Fra 10s til 13s stiger signalet lineært til 720. Det er med andre ord ikke et direkte hopp til 720 etter 13 s.