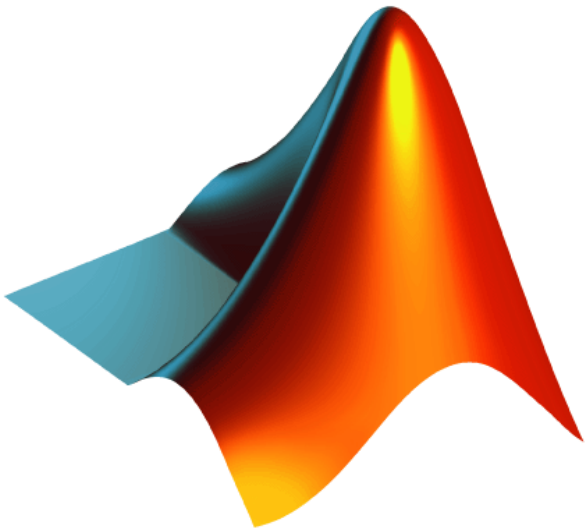
Teknostart 2024, Kybernetikk og robotikk

August 2024

1 Hva er Teknostart?

Teknostart er ment å være en liten forsmak på ting som kan komme til å møte dere under studiet, blant annet programmering, gruppearbeid, og bratte læringskurver. Selve oppgaven går ut på å bygge og programmere en robot av typen LEGO Mindstorms EV3®. Ved hjelp av sensorer, motorer og programmeringsverktøyet Simulink skal roboten klare å navigere seg gjennom en løype bestående av hindringer, og søke seg frem til mål. Etter prosjektet sitter man forhåpentligvis igjen med en litt bedre forståelse av hva kybernetikk egentlig er, og en demonstrasjon av hva det kan brukes til i form av en fungerende Legorobot.

Om man lurer på noe, både når det gjelder prosjektets innhold og praktisk informasjon, er det bare å spørre læringsassisten som er tilstede på rommet man sitter. Ellers kan det meste av praktisk info finnes på https://www.ntnu.no/studier/mttk/studiestart.



Figur 1: Prosjektet er basert på bruk av MATLAB og Simulink fra The Mathworks.

2 Autonome kjøretøy

Biler blir stadig mer autonome, og ifølge de mest optimistiske spådommene kommer det til å selges biler uten ratt og pedaler om kun få år [1]. Allerede er det selvkjørende busser i rute på norske veier [2]. Kjøretøy som selv tar seg av kjøringen er imidlertid ikke bare en praktisk måte å komme seg fra A til B på, de utgjør også en interessant problemstilling som for tiden sysselsetter mange kybernetikere. For å løse Teknostartoppgaven kan det derfor være nyttig å se på hvordan virkelighetens selvkjørende biler og busser fungerer.

Figur 2: Fremtiden er elektrisk.

2.1 Hvordan fungerer et autonomt kjøretøy?

Et autonomt kjøretøy er et stort og komplisert system, men kort oppsummert kan virkemåten til en selvkjørende bil deles inn i fire deler. Disse er illustrert i figur 3, 4, 5 og 6.

Del 1: Persepsjon/sansing

Først må man skaffe seg et bilde av omverdenen. Ved hjelp av RADAR, LIDAR eller vanlige kameraer skaffer kjøretøyet seg en oversikt over verden rundt.

Del 2: Lokalisering

Når man har et bilde av omverdenen, må man finne ut hvor i dette bildet man befinner seg. Retningen man kjører, og avstanden til ulike punkter bestemmes av sensorer som f.eks. GPS.

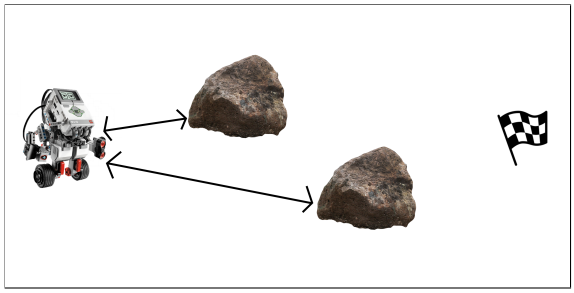
Del 3: Ruteplanlegging

Basert på den nevnte informasjonen kan man nå bestemme seg for hvor man har lyst til å kjøre. Man ønsker gjerne å unngå kollisjon samtidig som man ender opp der man skal fortest mulig. Å finne en rute som oppnår dette er det ruteplanlegging går ut på.

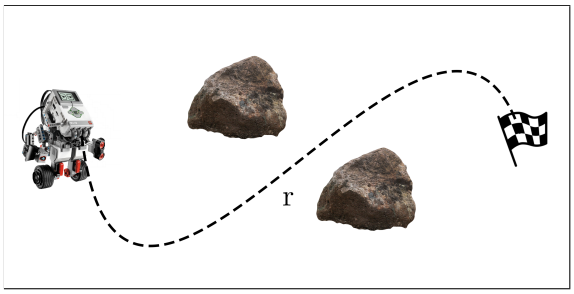
Del 4: Styring/regulering

Man vet hvor man er og hvilken rute man ønsker å følge, men hvordan får man kjøretøyet til å faktisk følge denne ruta? Dette er et reguleringsproblem, og slike problemer (der man ønsker å nå en **referanse** r ved hjelp av et **pådrag** u) utgjør kjernen av fagfeltet kybernetikk.

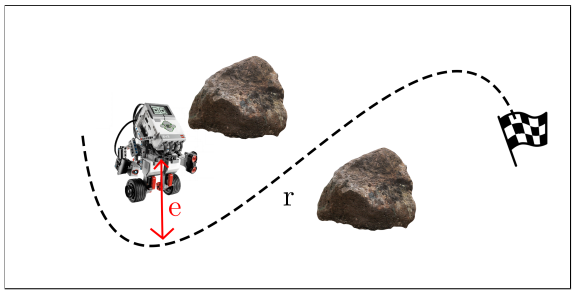
Figur 3: Persepsjon



Figur 4: Lokalisering



Figur 5: Ruteplanlegging



Figur 6: Styring

3 LEGO Mindstorms EV3

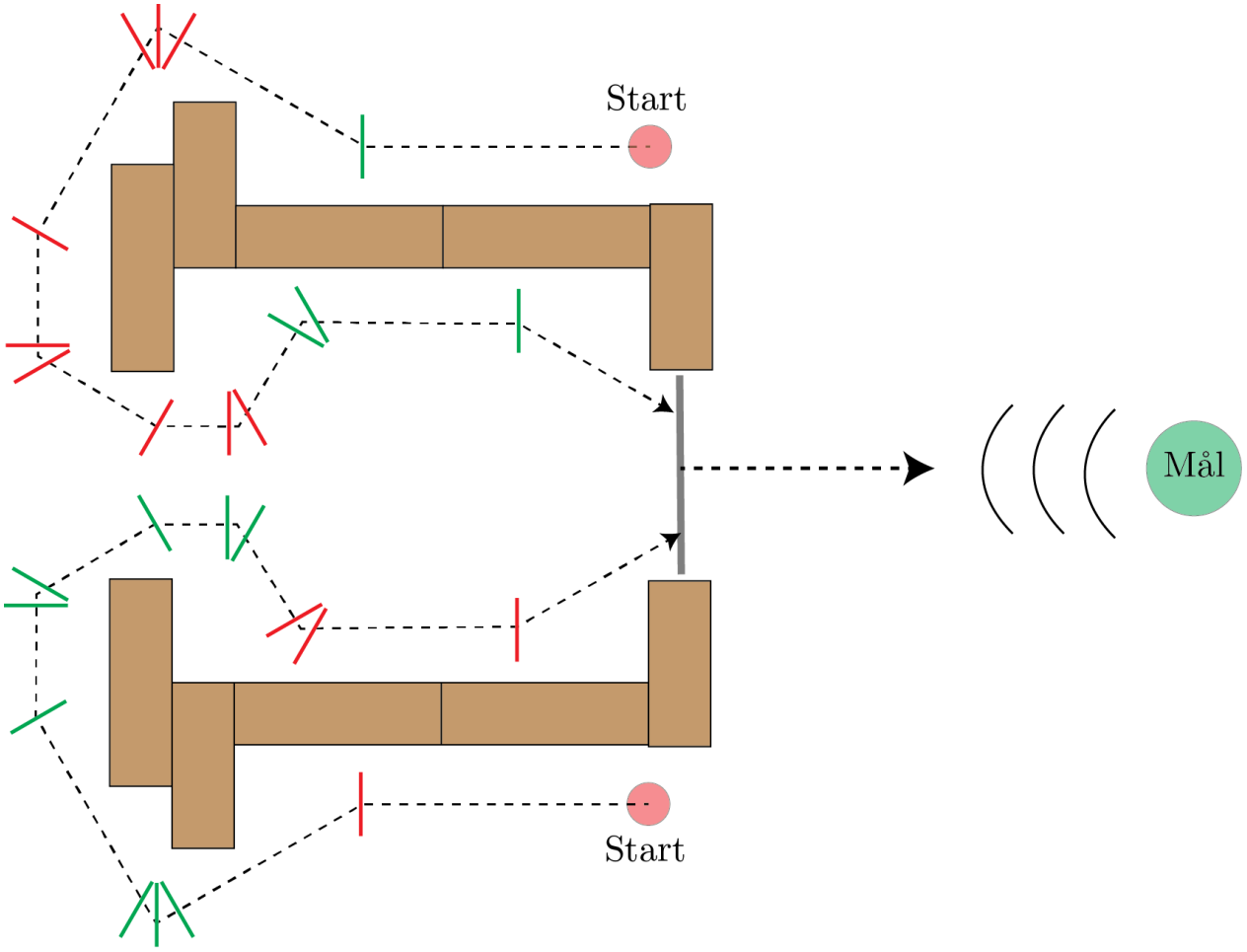
For å brukes til å løse oppgaven deles det ut et Legosett som består av følgende deler

- Tre motorer
- En gyrosensor som måler vinkelakselerasjon
- En fargesensor som kan registrere sju ulike farger
- En ultralydsensor som måler avstand
- En infrarød sensor som kan måle avstand og vinkel i forhold til en infrarød sender
- To berøringssensorer som registrerer når de er trykket inn
- EV3-brikken, som står for logikk og styring
- En hel haug med legoklosser

Hva man kan gjøre med disse delene er det kun fantasien som setter grensene for.

4 Oppgaven

Det vil etter hvert bli satt opp en løype i Glassgården utenfor EL5. Målet er å programmere roboten til å på egenhånd navigere seg gjennom løypa, uten å bli fjernstyrt eller motta annen ekstern hjelp mens den kjører. For å vise hvor man bør svinge er banen utstyrt med grønne og røde teipbiter. En grønn teipbit betyr at man burde svinge 30 grader til høyre, mens en rød teipbit betyr at man bør svinge 30 grader til venstre. Utover at man burde følge sunn fornuft er det ingen begrensninger på hvordan man løser oppgaven. Banen er illustrert i fig. 7 (men merk at den faktiske banen kan avvike fra illustrasjonen). De fargede strekene viser teipbitenes plassering. Banen på illustrasjonsbildet viser to speilvendte, men identiske baner: å kjøre fra start til mål på én side regnes som en hel runde. De stiplede linjene illustrerer hvordan robotene forhåpentligvis manøvrerer seg gjennom banen. Der de stiplede linjene møtes har man ikke lenger noen teipbiter å kjøre etter, men må sikte seg inn mot en IR-sender som sender ut signaler man kan sikte seg etter. Rundt denne går målstreken (illustrert som et grønt felt).



Figur 7: Grov skisse av banen som skal kjøres.

4.1 Hva har egentlig dette med autonome biler å gjøre?

Oppgaven kan på samme måte som virkemåten til en ekte selvkjørende bil deles inn i flere deler:

Del 0: Bygging av roboten

For å designe en velfungerende robot bør man i tillegg til de nevnte punktene *bygge* en god robot. Dette er kanskje delen av oppgaven med størst rom for kreativitet, så her er det bare å slå seg løs.

Del 1: Persepsjon/sansing

Dette gjøres av sensorene man velger å benytte seg av. Det er derfor viktig å velge sensorer som gir informasjonen man trenger i neste punkt.

Del 2: Lokalisering

Dette punktet går ut på å bruke målingene man gjør i del 1 til å finne ut hvordan roboten er plassert i banen. Posisjon, retning, og en oversikt over teipbitene man har kjørt over kan være nyttig informasjon.

Del 3: Ruteplanlegging

Dette krever sannsynligvis ikke så mye arbeid, siden teipbitene forteller hvilken rute man burde kjøre. Andre løsninger av ruteplanlegging er imidlertid også selvfølgelig lov

Del 4: Styring

Fra de tre første punktene har man forhåpentligvis en ønsket kjøreretning, og en måling av faktisk kjøreretning. Hvordan burde man bruke informasjonen om ønsket og faktisk retning (og avviket mellom dem) til å bestemme hva roboten sine motorer skal gjøre? Dette antakeligvis den vanskeligste delen av oppgaven, og om man klarer den er man godt i gang med å lære seg kybernetikk!

5 Blokkdiagram og simulink

For å programmere roboten skal vi benytte oss av **Simulink**. Simulink er et grafisk programmeringsverktøy, og kommer til å bli brukt mye senere i studiet. Man programmerer i Simulink ved hjelp av **blokkdiagram**, som er mye brukt i kybernetikken. Her er en kort introduksjon. Om man vil ha flere og mer detaljerte eksempler kan man sjekke tilleggseksemplene som ligger på Teknostart sine nettsider.

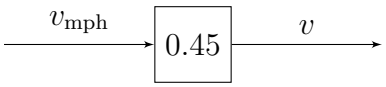
5.1 Blokkdiagram

Et blokkdiagram er en grafisk måte å representere likninger på. Når man jobber med store systemer med mange likninger er det ofte enklere å holde styr på en slik tegning enn et lass med likninger. Et blokkdiagram er et eksempel på en slik tegning. For å forstå hvordan et blokkdiagram kan brukes er det nyttig med et eksempel. I dette eksempelet er roboten vår en autonom taxi, og den har én oppgave: *Kjør i 100 meter, og stopp.*

Den hypotetiske roboten vår er utstyrt med et speedometer, slik at vi til enhver tid kjenner til hastigheten dens. Det er bare ett problem: Speedometeret viser hastigheten i $\frac{\text{miles}}{\text{time}}$! Vi vil åpenbart ha målingen oppgitt med en sivilisert måleenhet, f.eks. $\frac{\text{meter}}{\text{sekund}}$. Vi kaller hastigheten vist på speedometeret for v_{mph}, og hastigheten i meter per sekund for v. Da gir et raskt oppslag i leksikon at

$$v = 0.44704 \cdot v_{\text{mph}} \tag{1}$$

For enkelthets skyld bruker vi herfra at $v \approx 0.45 v_{\text{mph}}$. Da vil et blokkdiagram av omgjøringen vår se ut som følger



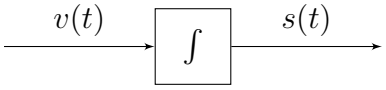
Figur 8: Blokkdiagram

Denne blokken gjør altså operasjonen "gang input med 0.45". Men det er ikke bare de grunnleggende regneoperasjonene man kan representere vha. blokkdiagram.

Roboten vår kjenner nå til hastigheten den kjører, men den trenger å vite hvor *langt* den har kjørt. For å finne avstand fra hastighet dytter man hastigheten inn i et integral. Integralet spytter da ut en posisjon:

$$s(t) = \int_0^t v(t)dt \tag{2}$$

Men roboten vår forstår bare blokkdiagram, derfor dytter vi hastigheten inn i en blokk i stedet:



Figur 9: Blokkdiagram

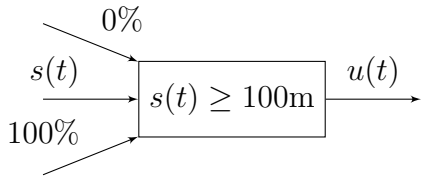
Nå vet vi hvor langt vi har kjørt, men vi har fortsatt ikke fortalt roboten hva den skal gjøre. Som nevnt er robottaxiens oppførsel bestemt av følgende likning

$$\text{Skal roboten gi gass?} = \begin{cases} \text{Nei}, & \text{hvis } s(t) \geq 100\text{m} \\ \text{Ja}, & \text{ellers} \end{cases} \tag{3}$$

Om vi skal være litt mer formelle kan vi si at robotens **pådrag** ved tidspunkt t, ofte kalt $u(t)$, er gitt av

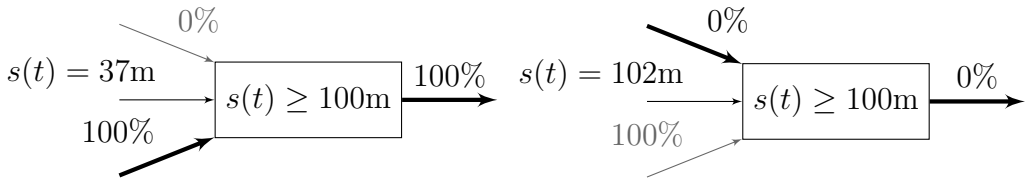
$$u(t) = \begin{cases} 0\%, & \text{hvis } s(t) \geq 100\text{m} \\ 100\%, & \text{ellers} \end{cases} \tag{4}$$

En slik "hvis-ellers"-funksjon representers i Simulink som en **switch**. Den har tre inputs, og om kriteriet i midterste input er oppfylt, vil øverste input bli blokkens output. Hvis ikke, så blir nederste input blokkens output. Om dette var forvirrende er blokkdiagrammet forhåpentligvis oppklarende



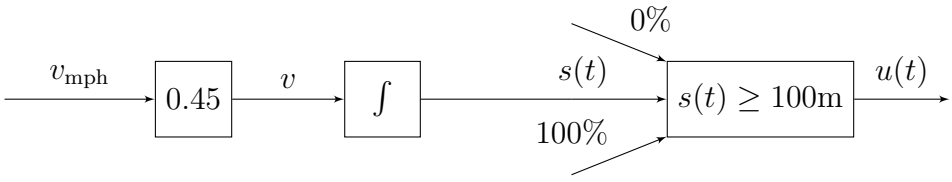
Figur 10: Blokkdiagram med flere inputs

To eksempler på mulige tilfeller er da

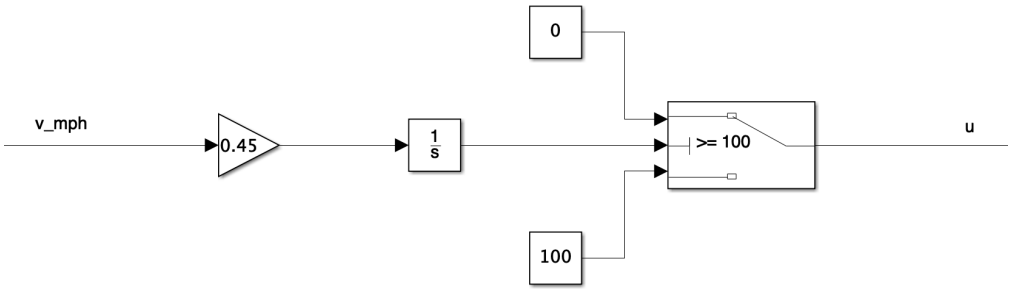


Figur 11: Switch-blokk

Om vi kobler sammen alle blokkdiagrammene våre ender vi opp med et system som tar inn en speedometermåling oppgitt med ubrukelige enheter, og gir ut hvor mye gass roboten skal gi!



Merk at blokkdiagrammene her bare er illustrasjoner. I Simulink vil systemet vi nettopp har designet se ut som vist i figur 12.

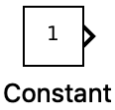


Figur 12: Ferdig system implementert i Simulink

5.2 Blokker

Det finnes en stor mengde blokker i Simulink, og mye av denne oppgaven går ut på å finne de blokkene som best løser problemet. Her følger noen blokker som *kan* være nyttige.

Constant

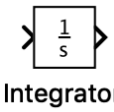


Gir ut en konstant verdi.

Gain

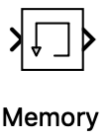
Multipliserer input med en bestemt verdi.

Integrator



Integrerer opp input.

Memory

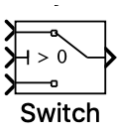


Husker på input, og gir den ut ved neste tidssteg.

Sum

Summerer opp alle inputs.

Switch

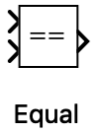


Som forklart i eksempelet over. Gir ut øverste input om midterste input oppfyller et gitt kriterie, gir ut nederste input hvis ikke.

Hit crossing

Gir ut 1 idet input overstiger en gitt threshold-verdi, 0 ellers. Om input f.eks. går fra 3 til 5, og threshold-verdien er 4, vil output i dette øyeblikket være 1. Men om input fortsetter å være 5 (som er større enn 4), vil output bli 0 igjen.

Equal



Gir ut 1 om begge inputs er like, 0 hvis de ikke er det.

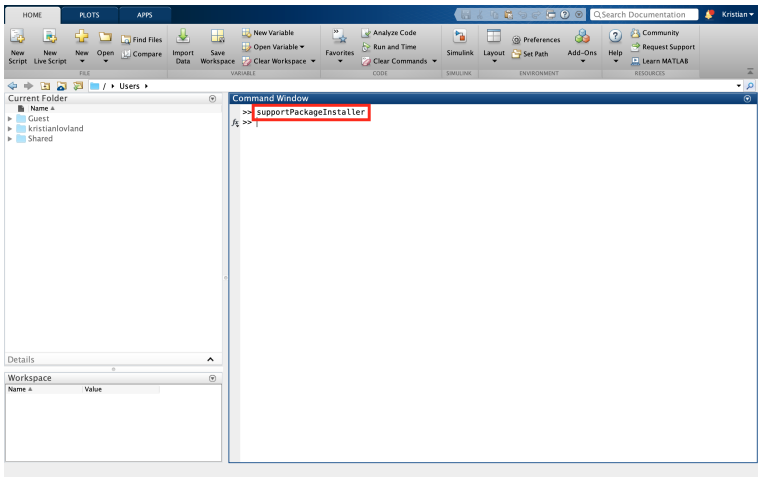
5.3 Installasjon av Matlab og Simulink

Simulink kommer inkludert med programmet Matlab, som man kan kjøre fra NTNUs programvaresider (`https://apps.ntnu.no/`). Følg installasjonsguiden for AppsAnywhere (ved å følge den forrige url-en), så søk på "Matlab" og trykk start. Merk: Et alternativ er å laste ned Matlab på maskinen: Hvis du vil gjøre dette kan du følge oppskriften på AppsAnywhere. Snakk med en læringsassistent hvis du får problemer.

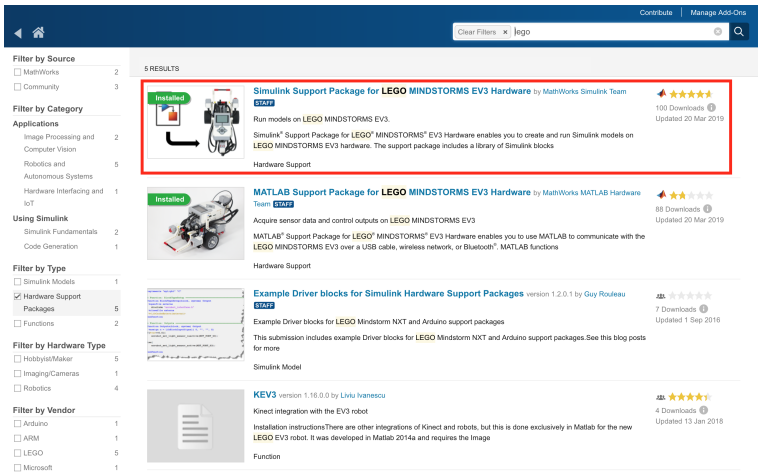
5.4 Installasjon av support package for LEGO Mindstorms EV3

Når man har installert MATLAB og Simulink må man installere programvare som gjør det mulig å programmere Legoen. Dette gjøres som følger:

1. Åpne Matlab.



Figur 13: Installasjon av Lego support package



Figur 14: Installasjon av Lego support package

2. Skriv "`supportPackageInstaller`" i det store, hvite terminalvinduet, og søk på "Lego" (som vist i fig. 13). Dukker ikke pakken opp, prøv å trykk krysset ved "Clear Filters" i søkefeltet.

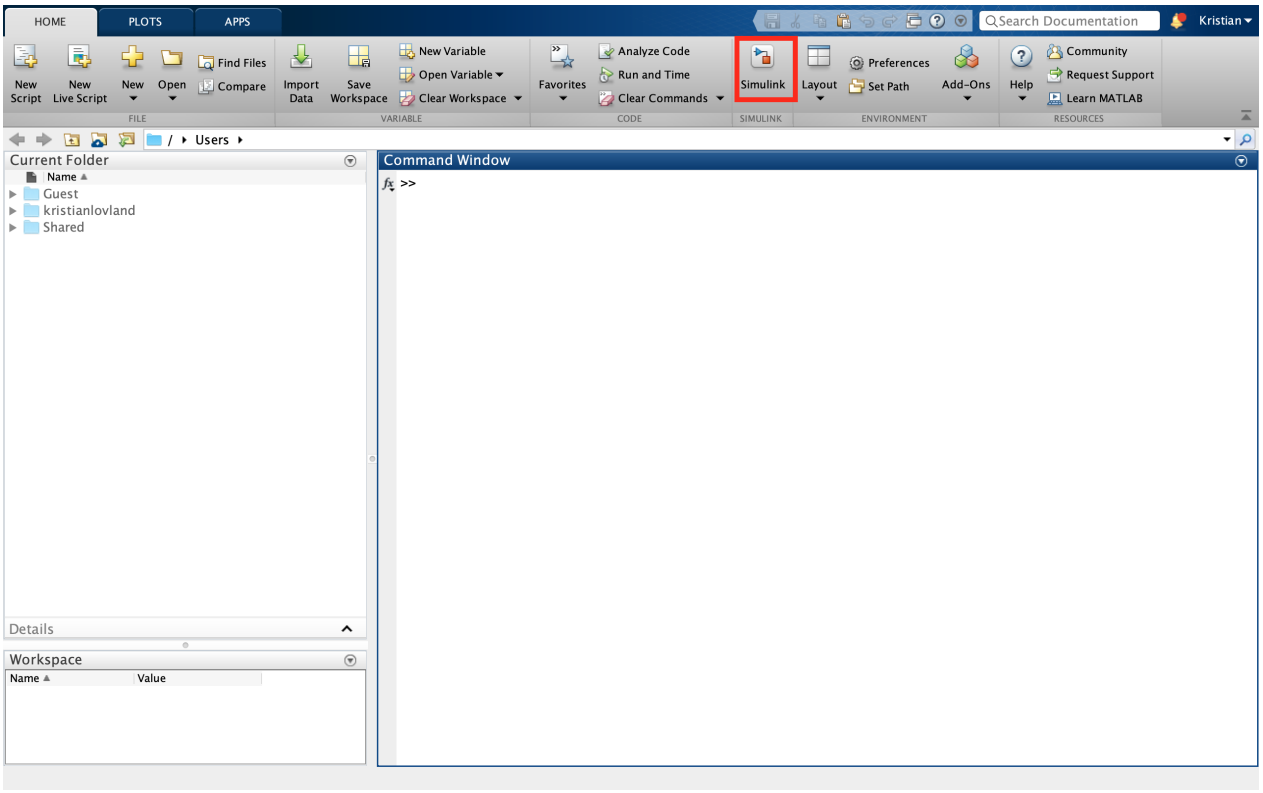
3. Du skal nå kunne finne pakken "`Simulink Support Package for LEGO MINDSTORMS EV3 Hardware`". Installer denne (illustrert i fig. 14)

Om man ikke finner frem er det bare å spørre læringsassistent om hjelp!

5.5 Programmering av LEGO Mindstorms EV3

Nå er man endelig klar til å programmere legoen! Her følger et eksempel på hvordan man gjør dette.

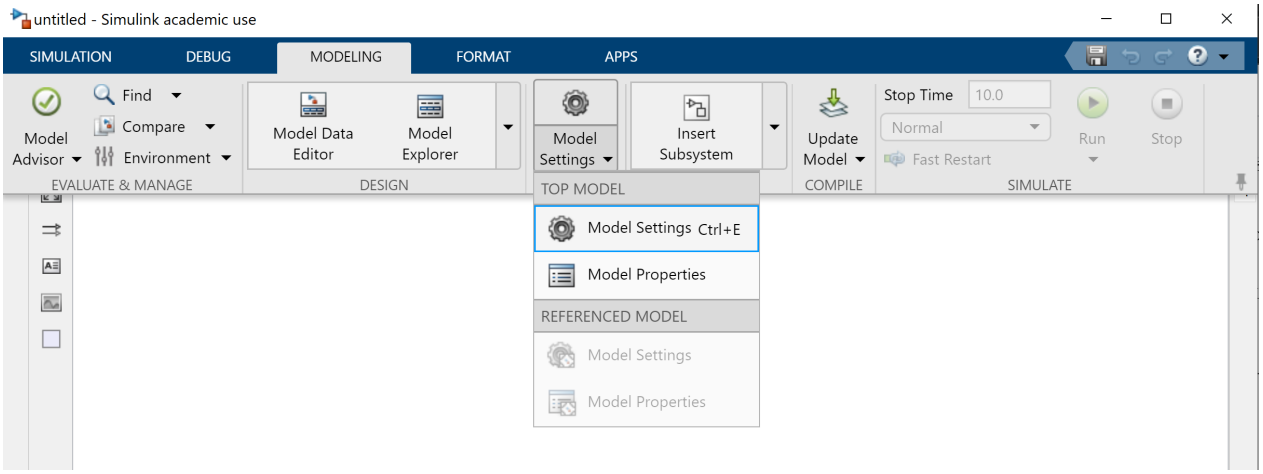
- Skru på EV3-brikken, og koble den til PC-en med USB-ledningen.
- Åpne Matlab, og trykk på "Simulink"-knappen oppe på menyen (figur 15).



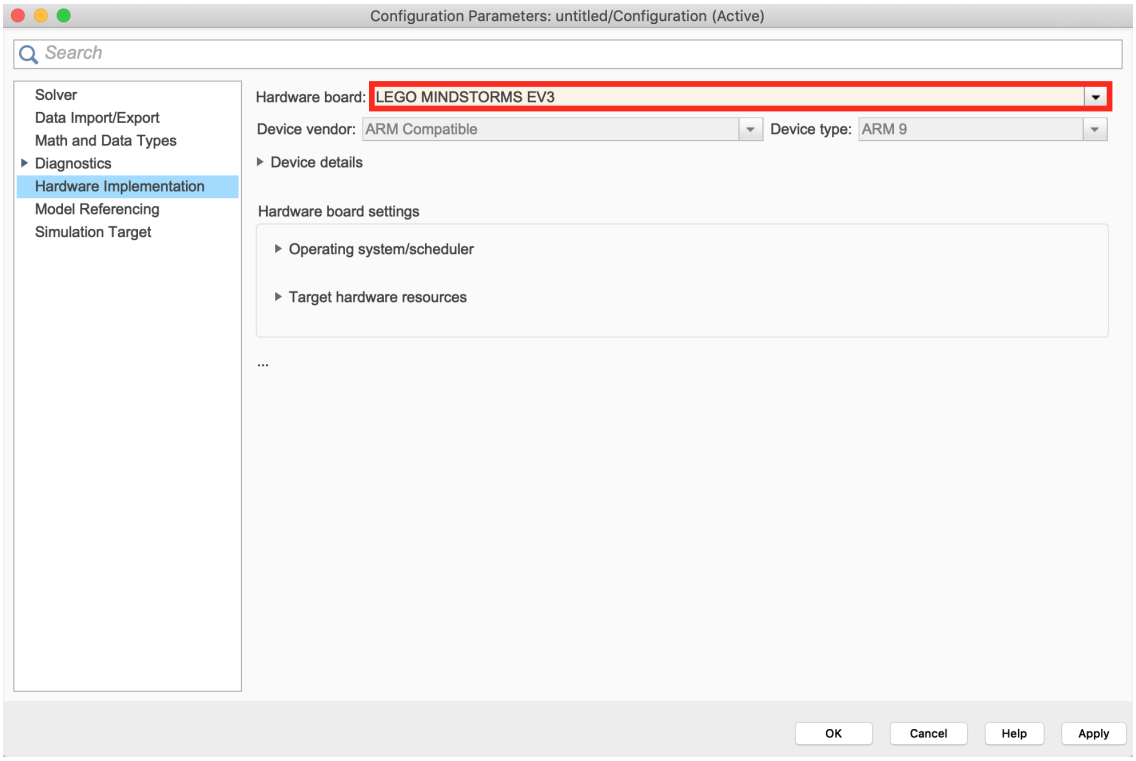
Figur 15: Åpne Simulink fra Matlab

- Når man lager en ny fil må man fortelle Simulink at den skal kjøres på en Lego-brikke. Dett gjør man ved å trykke på **Modeling** → **Model settings** (figur 16).
- I feltet **Hardware Board** velger man **LEGO MINDSTORMS EV3** (figur 17).
- Trykk på **Apply**. Dette trenger bare å gjøres første gang man bruker en fil.
- Lag programmet. For å finne blokker trykker man på knappen vist i fig. 18, eller man kan bare begynne å skrive navnet på blokka uten å trykke på noe som helst først.
- Når man har laget ferdig programmet (f.eks. det noe primitive systemet i fig. 19, trykker man på **Deploy to hardware**-knappen (fig. 20), eller trykker på **Ctrl + B**.

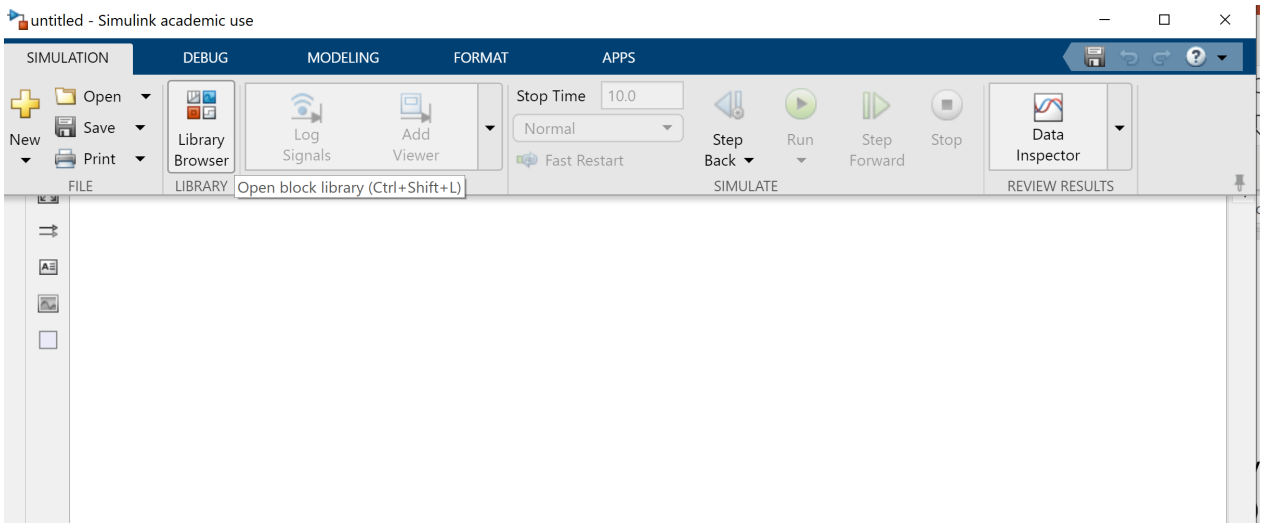
En ting som er verdt å merke seg er at om man forsøker å laste over et program flere ganger fra samme pc, kan man få en feilmelding (`An error ocurred while deploying hardware` eller noe i den duren). Dette er litt uheldig, men heldigvis finnes det en enkel løsning: Åpne Matlabvinduet (ikke Simulink), og skriv `clear java` i vinduet. Prøv så igjen. Hvis ikke dette fungerer, spør læringsassistent om hjelp.



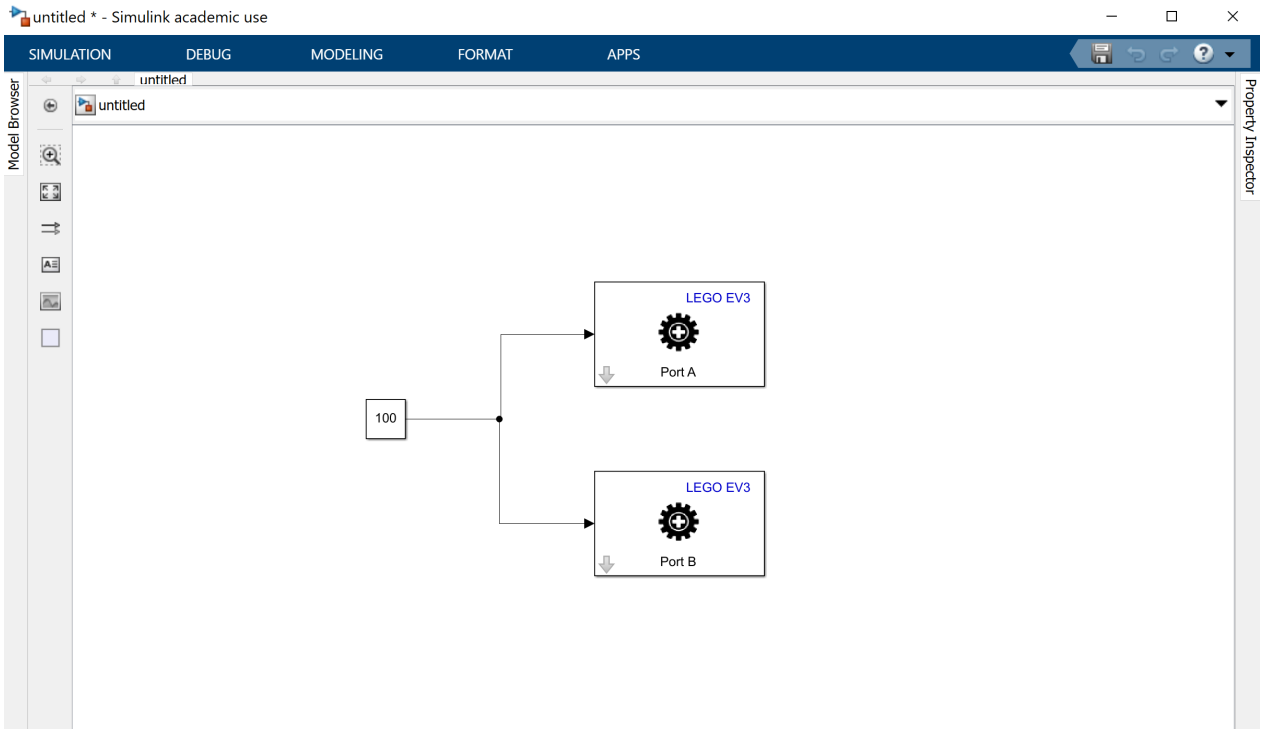
Figur 16: Forbered Simulink til å kjøre på LEGO-hardware ved å endre innstillingene.



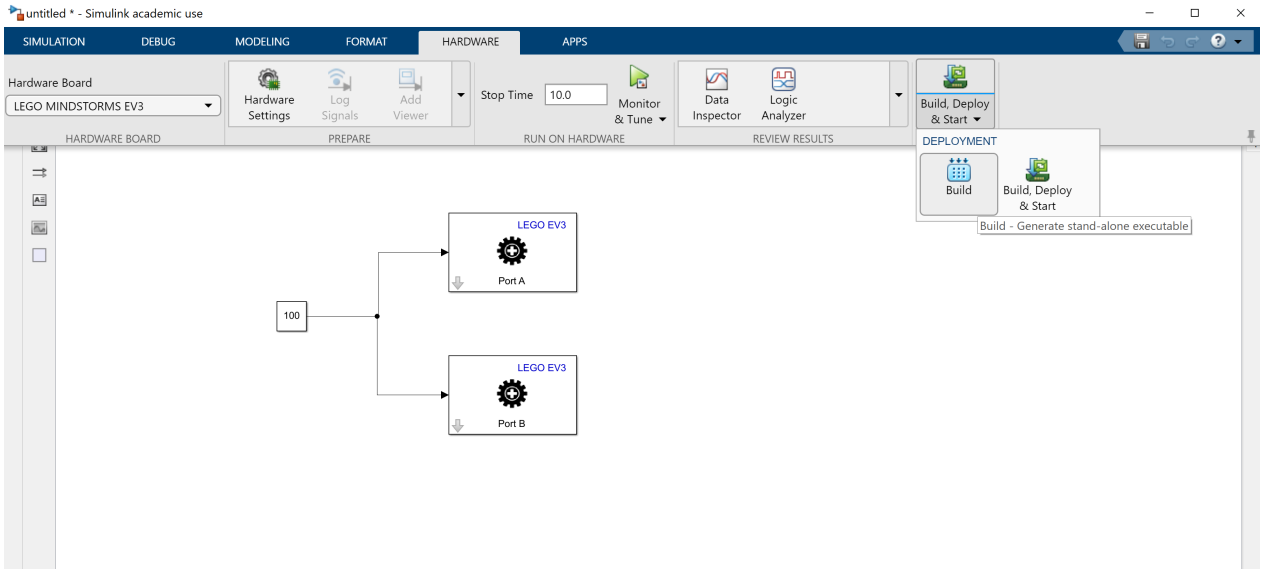
Figur 17: Innstillinger som forbereder Simulink for kjøring på Lego-hardware



Figur 18: Åpne Simulinkblokkmeny



Figur 19: Primitivt system



Figur 20: Send kjørbart program til Lego

6 Konkurransen

Konkurransen går ut på å gjennomføre en runde på banen på kortest mulig tid. Alle gruppene får mulighet til å kjøre banen i to tidsrom på konkurransedagen. Vinneren får i tillegg til heder og ære et gavekort på pizza. Det vil også deles ut en premie for mest kreative robot, så her er det bare å bruke Legoen godt!

6.1 Regler

§1. Tidspunkt

Konkurransen gjennomføres på fredag. Konkurransetidspunkt for de ulike gruppene vil bli lagt ut på hjemmesiden til Teknostart https://www.ntnu.no/studier/mttk/studiestart.

§2. Gjennomføring

Hver gruppe har fem minutter til rådighet, og kan gjennomføre så mange forsøk de vil i løpet av denne tiden. Etter en pause får hver gruppe nye fem minutter. Den beste tiden i løpet av de ti minuttene blir tellende.

§3. Autonomitet

Roboten skal være autonom, det vil si at man ikke har lov til å styre eller berøre den underveis. Om roboten kjører seg fast eller er fullstendig ute av kurs kan man stoppe den, og håpe på at den klarer seg bedre etter en omstart. Å berøre roboten vil imidlertid straffes med en milliard ekstrasekunder.

§4. Lumsk oppførsel

Alle gruppene skal ha like forutsetninger for å løse oppgaven. Dermed kan kun den utdelte Legoen brukes, og sabotasje av andre grupper er ikke lov.

7 Hva gjør vi nå?

Om man har kommet seg helt hit er man forhåpentligvis klar for å starte på prosjektet. Hvis ting virker uklart eller vanskelig er det ingen grunn til å bekymre seg, det er ikke uvanlig at prosjektet har en ganske bratt læringskurve. Heldigvis er det dyktige læringsassistenter tilgjengelig, og det er bare å stille dem spørsmål om man lurer på noe. Lykke til!

Referanser

[1] https://electrek.co/2019/04/23/tesla-design-without-steering-wheel-elon-musk/

[2] https://www.tu.no/artikler/lykkes-der-andre-feiler-setter-inn-forerlos-buss-pa-ordinaer-bussrute/463500