DTU Elektro, Automation and Control

31300/31301 Reguleringsteknik 1

Forår 2018 rev 3/jca

Regbot og håndtuning - del 1
Øvelsesvejledning

Formål

Øvelsen understøtter følgende læringsmål:

1. Anvende analyse og simuleringsprogrammel (MATLAB og Simulink).

(Simulink kommer senere)

Indhold

Øvelsen er tilrettelagt til at give et indledende bekendtskab med kursets øvelsesrobot og hvordan data fra robotten overføres og behandles i MATLAB.

Der skal køres en firkant-mission hvor hjulene styres blot med en spænding, uden brug af regulering. Det forventes at hjulene kører ca. lige hurtigt når de får samme spænding, og det derfor blot med motorspændingen er muligt at styre robotten. Hvor godt det virker er det der skal undersøges i øvelsen.

Del 1: Intro og konfigurering af robotten

Opsætning af robotten

Del 2: Testmission og optagelse af data

Generering og kørsel af firkantmission

Del 3: Plot af data i MATLAB

Analyse af data.

Del 1: Intro og konfigurering af REGBOT

Se wiki siden http://rsewiki.elektro.dtu.dk/index.php/Regbot for for generel beskrivelse af robotten.

Kabler og batteri

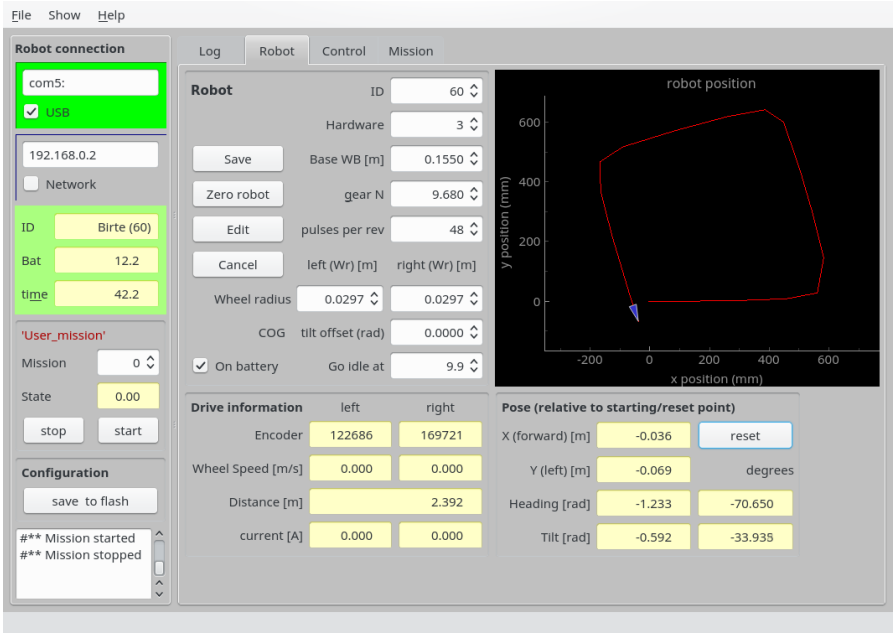
Forbind robotten med USB kabel - det er nok til at konfigurere robotten, for at robotten kan køre skal batteriet også være tændt (on).
Robotten kan også konfigureres over WiFi.

Batteriet oplades med den indbyggede lader. Det er gruppens ansvar at holde batteriet opladet. Batterispændingen er 12.4 V når det er fuldt opladet og ca. 10V når fuldt afladet, *så husk opladning og sluk når den ikke bruges*!).
Der er ca 3-5 timers brugstid pr opladning, det tager 1-2 timer for en fuld opladning.

Under kørslen logges data til processorens RAM. Så hold REGBOT tændt indtil de loggede data er overført.

Klientprogram REGBOT GUI

På PC'en startes REGBOT GUI (regbot.exe) som hentes fra campusnet under fildeling (det er et Python program, og kan også køres som python script *regbot.py* på PC'er hvor python og de rigtige Python biblioteker er installeret, se eventuelt wiki siden for detaljer).



Figur 1: Skærmbillede af klientprogrammet *regbot*. I venstre side vælges forbindelse over USB til en *comX* port, eller Wifi til en IP adresse. Grønne felter betyder forbundet. Gule felter bliver opdateret fra robot, og kan ikke ændres.

I Øverste venstre hjørne skrives den COM-port der oprettes når robotten forbindes (måske com3), og forbind ved at markere i "USB".

Hvis robotten er forbundet vises (bl.a) robottens navn og batterispænding.

Alternativt kan klient forbindes til robot over netværk, på DTU forbindes til adressen 10.16.166.XX:24001, hvor XX er robottens nummer. Wifi er dog betydeligt langsommere end USB og kan kan være mere ustabil. Check eventuelt om "port open" er markeret i fanen "wifi" (når forbundet over USB).

Undersøg lidt hvad der er tilgængeligt - eventuelt via "tool tip" når musen holdes over et felt.

GUIen kan gøres større for bedre at kunne læse tekst i visse felter.

Konfiguration af log

Under fanen "Log" styres hvad der skal gemmes under en kørsel. Der er kun ca. 30 kByte RAM reserveret til data, så jo mindre der logges jo længere tid kan der logges.

Sæt hak i "Allow logging". Logging "interval" styres når der oprettes en mission, det hurtigste er 1 ms, men så bliver logging tiden for kort til denne øvelse.
NB! hver gang der ændres i et felt sendes besked til robotten, og robotten svarer med den nye værdi, der så vises i feltet. Det kan af og til virke rigtigt irriterende, men når man ved det, er det til at leve med.

Marker i "log options" følgende f.eks. oplysninger:

- "Mission" er godt at have til at se hvilken linje missionen har gang i.
- "Motor voltage".
- "Motor current".
- "Wheel velocity" som er hjulenes hastighed på gulvet i $\mathrm{m\,s^{-1}}$.
- "Robot pose" er robottens position udregnet efter hjul-enkodere, x, y, h i forhold til start positionen, x,y i meter og h (heading) i radianer.
- "Battery voltage" da den kører lidt forskelligt ved forskellige batterispændinger.
- Fjern alle andre markeringer.

Konfigurer robot og deaktiver regulatorer

1. I fanen "Robot" er de vigtigste data:
    - "Wheel base" = 0.155m, som er afstand mellem hjulene (passer nok ikke helt).
    - "Gear N"= 9.68, som er udveksling i gerarkassen.
    - "pulses per rev" = 48, som er antal encoderpulser per motoromdrejning.

- ”Wheel radius“ = 0.03, som er hjulradius (ifølge hjul leverandør).
- ”go idle at“ må ikke være under 9.9 V (ellers er der fare for at ødelægge batteri).

Hvis der ændres i disse data skal de gemmes på robotten med ”Save“ efterfulgt af ”save to flash“.

2. I fanen ”Control“ konfigureres regulering af motorhastighed til kun at anvende ”feed forward“ (ingen regulator).
   - Klik i boksen ”Velocity“ for at konfigurere motor hastighedsstyringen.
   - aktiver ”Enable Controller” og sæt Kp=0 (ingen regulering).
   - aktiver “Feed forward” og sæt Kf=4 (hastigedsinput bliver direkte overført til motorspænding (i Volt). Enhed for Kf er således Volt per m/s (sættes en hastighed til 1 m/s fås en motorspænding på 4V – som ikke er helt galt med robottens gearing).
   - Sæt “output limit” til max 9 [V], som begrænser motorspændingen til +/- 9 V (det er en 6.5V motor).
   - Gem til robot med ”Apply“ eller ”OK“.
3. ”Heading“ regulatoren deaktiveres.
   - I fanen ”control“ vælges ”heading“ controller.
   - Her sikres at ”enable controller“ IKKE er valgt.
   - Gem til robot med ”OK“.
4. Den nye konfiguration er nu i robottens RAM, men for en sikkerheds skyld gemmes den også i robottens EE-prom (konfigurationsflash).
   - I venstre side i boksen ”Configuration“ klikkes på ”save to flash“, og robotten svarer med hvor mange bytes der er gemt (ud af de 2kByte der er plads til).

Del 2: Testmission og optagelse af data

Der skal oprettes og køres en mission, der skal forsøge at køre robotten i en firkant der burde ende samme sted som den startede.

Data fra mission skal gemmes og bruges til sammenligning i øvelse anden del (næste onsdag).

Der skal køres 2 firkanter en mod uret (CCV) og en med uret (CV).

1. Under fanen “Mission” oprettes missionen

   Missionen indføres i det hvide felt. Der oprettes en mission til at køre en firkant bestående af et lige stykke på 0.4 m og et 90 grader drej, alt sammen gentaget 4 gange, så start og slutposition burde være det samme sted:

   - Hastigheden skal give en motorspænding på $2\,\mathrm{V}$ (vel=0.5 $\mathrm{m\,s^{-1}}$ med en Kf=4)

- Det skal logges for hver 20 ms - eller langsommere, så hele missionen optages (log=20 skrives kun en gang i første linje).
- Sidelængden skal være 0.5 meter, brug (time=1) da der så burde køres 0.5 m på 1 sekund,
- Alle drej skal være 90° - positiv vinkel er CCV. Uden drejeregulering (som lige er disablet) kan der kun drejes på en måde (med fast drejeradius):
- Vælg en drejeradius på f.eks. 0.1 m (tr=0.1) og
- en drejevinkel på 90° (turn=90)

Kommandoerne som robotten forstår er beskrevet på
http://rsewiki.elektro.dtu.dk/index.php/Regbot under "Mission, how to write a mission".

Når missionen er indtastet og syntaks-checket, gemmes den til robottens RAM med "save to robot". (eventuelt kan den hentes tilbage fra robotten, for at se hvad robotten har opfattet, med "load from robot"..

2. Stil robotten med hjulene opad (det er derfor den har en flad overside) og prøv missionen (venstre panel "start" eller grøn knap på robotten).

   Det bør være synligt at hjulene køre fremad, at det ene hjul står (næsten) stille når der drejes, og at drej er afbrudt af ligeud-kørsler.

3. Marker en startposition på gulvet, så det er muligt at måle hvor langt der bliver fra start til slutposition.

   Afbryd forbindelsen til robotten, fjern USB ledningen og kør missionen på gulvet.

   Det vil formentlig ikke give en helt flot firkant, noter hvor langt der er fra start til slutposition i tabel 1.

4. Tilslut igen USB forbindelsen. Hent de loggede data fra missionen i "log" fanen "get log from robot" og gem dem til en fil. NB! knappen "save as" gemmer ikke data, men kan kun bruges til at vælge hvor data skal gemmes, tryk derefter på "save" (med mindre fejlen er rettet).

Del 3: Plot af data i MATLAB

Start MATLAB og skift til den folder hvor logfilen blev gemt.

Opret et nyt script, der loader filen og laver et x,y plot af missionen i både CV og CCV udgaven.

Endvidere skal hastigheden for venstre og højre hjul for en af missionerne plottes og vurderes.

Denne detaljerede beskrivelse er mest for dem der ikke har prøvet MATLAB før, dem der er erfarne med matlab kan gå direkte til resultat afsnittet.

Oprettelse af *script*

1. Start MATLAB.

2. Skift til den folder hvor logfilen er placeret, f.eks. ved at skrive “cd regbot” hvis folderen med filen hedder regbot, eller “browse” til den rigtige folder.

3. I “command window” kan skrives alle udregninger, men er bedre at lave et “script”. Klik på “new script” for at starte script editor.

4. I editor indtastes alle kommandoer så det er let at tilføje flere grafer i samme plot. Her antages at logfilerne hedder “square_CV.txt” og “square_CCV.txt”.

5. De første linjer af logfilen er kopieret ind som kommentar i scriptet for lettere at huske hvad talkolonner betyder.

```
close all
clear
%%
dataCV = load('square_CV.txt');
dataCCV = load('square_CCV.txt');
% Birte (60)
%  1    time 0.013 sec
%  2  3  4   (mission 0), state 2, thread 2, line 0
%  5  6 Motor voltage [V] left, right: 2.0 2.0
%  7  8 Motor current left, right [A]: -0.144 -0.052
%  9 10 Wheel velocity [m/s] left, right: 0.0000 0.0004
% 11 12 13 14 Pose x,y,h,tilt [m,m,rad,rad]: 0 0 0 0.515647
% 15    Battery voltage [V]: 12.22
%% plot path
figure(100)
plot(dataCV(:,11), dataCV(:,12), 'b')
hold on
plot(dataCCV(:,11), dataCCV(:,12), 'r')
set(gca,'FontSize',12)
grid on
grid MINOR
%title('Robot Birte (60), Square - no controller')
xlabel('X [m]')
ylabel('Y [m]')
legend('CV square', 'CCV square')
axis equal
```

6. Gem filen med et matlab-egnet navn, dvs ingen mellemrum, ingen ’+’ eller ’-’ og ingen danske bogstaver, og filen skal ende med et ’.m’.

7. Kør ’scriptet’ ved at taste *ctrl-shift-enter* for hver blok eller klik på *Run*.

8. Hvor godt ramte robotten samme position ifølge loggen?

   Det kan eventuelt udregnes som en del af scriptet således:

```
dx = data(end,7) - data(1,7);
dy = data(end,8) - data(1,8);
afstand = sqrt(dx^2 + dy^2)
```

Resultater

Noter resultater i tabel 1.

Tabel 1: Resultat af firkantkørsel, både efter robottens opfattelse og målt på gulv.

	Robot	CV	CCV
Afstand målt på gulv (start til slut)	Freja 4	X: 45, Y: -168 cm	X: 12, Y: -1 cm
Afstand optaget i log (start til slut)	Freja 4	X: -11, Y: -89 cm	X: 0, Y: -4 cm

1. Er der forskel på afstanden mellem start og slutposition på gulv og i log?
   I log kan afstand fra (0,0) til slutposition findes med MATLAB kommandoen `hypot(dataCCV(end,11),dataCCV(end,12))`. Overvej forskel (hvis der er forskel).
2. Plot motorspændingen som funktion af tiden
   (f.eks. `plot(dataCCV(:,1), dataCCV(:,5), 'c')`), kolonne 1 er tiden, kolonne 5 og 6 er motorspænding i denne logfil.
   Er motorspændingen på de 2 hjul konstant og ens, når der køres ligeud?
3. Plot hjulhastigheden på de 2 hjul fra den ene kørsel. Det yderste hjul skulle holde en konstant hastighed (på $0.5\,\mathrm{m\,s^{-1}}$). Hvor godt lykkedes det — og hvorfor ikke?
4. Kører de 2 hjul samme hastighed når der ikke er kommanderet drej?

(slut)
Næste gang skulle det gerne blive bedre med noget aktiv regulering.

De næste vejledninger bliver mindre detaljerede.

Matlab funktioner

Kort forklaring til udvalgte matlab kommandoer.

- `close all` closes all figures.

- `clear` clears all variables (as restart in Maple).
- `%` og `%%` er kommentartegn (resten af linjen) og `%%` angiver start på en ny kodeblok, der letter eksekvering (med *ctrl-enter* og *ctrl-shift-enter*).
- `data = load('logfil.txt');` loader en fil til en 2-dimentional matrix. Filnavnet gives som parameter, hver linje der ikke er en kommentar er en række og tal i kolonner er adskilt af mellemrum.
  Semikolon efter kommando gør at data ikke udskrives på skærmen (som kolon i Maple).
- `figure(100)` åbner/vælger en specifik (ny) figur (her med nummer 100).
- `plot(data(:,1), data(:,11))` plotter en kurve med data fra `data` matrisen, hvor x-aksen tages fra kolonne 1 i alle rækker (: betyder alle), og y-aksen fra kolonne 11.
- `hold on` gør at der kan plottes flere kurver i samme figur.
- `axis equal` bruges især til x-y plot hvor akserne har samme enhed.
- `grid on` viser gitterlinjer i plot (krav i dette kursus).
- `grid MINOR` Giver flere gitterlinjer.
- Brug også `xlabel`, `ylabel` og `legend` til at gøre figurer mere forståelige.
- `hypot(x, y)` er en almindelig matematikfunktion der giver længden af en x,y vektor (hypotenusen).
- `set(gca,'FontSize',14)` angiver fontstørrelse på aksemarkeringer, som kan være en fordel, hvis figuren skal bruges i en rapport (og aksemarkering læselig).
- `print('square_no_control','-dpng')` printer en figur til en fil i *png* format.
- `help hypot` *help* i kommandovinduet giver en hurtig kort hjælp til en kommando.

Brug MATLAB, selvom Maple kan meget af det samme, når vi kommer til Matlab-Simulink er Maple ikke et alternativ.