Introduktionsdokument

Erik Frisk <erik.frisk@liu.se>

16 augusti 2016

Version 1.1

Status

Granskad	Viktor Leek	
Godkänd		

Styrning och optimering av bilbana

PROJEKTIDENTITET

Beställare:	Fordonssystem, ISY, LiTH, 581 83 Linköping.
Kontaktperson beställare:	Erik Frisk, tel: 013-28 5714, <erik.frisk@liu.se>
Handledare:	Viktor Leek <viktor.leek@liu.se>

INNEHÅLL

1 INLEDNING

I fordonssystems undervisningslaboratorium finns en bilbana där två bilar kan styras antingen via en dator eller ett körhandtag. Alla som kört bilbana vet att det är ganska enkelt att köra ett varv snabbt, men svårt att hela tiden köra snabbt och säkert. En dator kanske inte kan köra lika snabbt som en människa, men kan istället köra med stor precision. I projektet bilbanestyrning strävar vi efter att utveckla programvara i Matlab för att styra bilbanan med en dator. Bilarna ska köras med målet att hålla, en på förhand given, konstant varvtid.

För att hålla en konstant varvtid behövs olika gaspådrag vid olika områden i banan beroende på bil, friktion och kontakt med de elektriska ledarna. Därtill förändras bilbanans och bilarnas egenskaper under körning, vilket leder till olika varvtider för samma gaspådrag. För att hålla en given varvtid krävs därför en *adaptiv* styrning av bilarna, dvs. gaspådraget måste anpassas efter bilen som befinner sig på banan och hur länge den har körts. Den algoritm som styr gaspådraget är en s.k. regulator och implementeras i detta projekt i Matlab.

I datorn som är kopplad till bilbanan finns ett instickskort som registrerar digitala insignaler och genererar utsignaler, d.v.s. styrsignaler. Utmed bilbanan finns ett antal givare placerade, vilka signalerar att en bil passerat. Givarna är kopplade till datorn som registrerar när en givare signalerar och räknar ut hur lång tid som förflutit sedan förra givaren signalerade. På banan finns separata givare för att signalera att ett nytt varv påbörjats. Till hjälp i projektet finns programvara utvecklad i Matlab för att styra bilarnas gaspådrag och få information om att en givare passerats eller ett nytt varv påbörjats.

I avsnitt 2 beskrivs var Matlab-programvaran finns tillgänglig. En beskrivning av mjukvaran finns i avsnitt 3. Displayen beskrivs i avsnitt 4. Appendix A ger en kort introduktion till några standardfunktioner i Matlab som kan visa sig vara användbara i detta projekt. Sist i dokumentet är en teknisk specifikation av displayen bifogad.

2 ÅTKOMST AV PROJEKTFILER VIA VERSIONSHANTERINGSSYSTEMET GITLAB

Versionshanteringssystem är i det närmaste ett måste då mjukvaruprojekt genomförs. Versionhanteringssystem hanterar dokument, som källkod och textfiler, genom att spara både gamla och nya versioner av dokumentet i en databas. Det medför att det är lätt att gå tillbaka till äldre versioner av ett dokument om detta skulle behövas.

I detta projekt får varje grupp tillgång till aktuella projektfiler via ett versionshanteringssystem som heter Git[1]. I Windows är detta versionshanteringssystem enkelt att använda, man högerklickar med musen i exempelvis utforskaren och väljer vad som skall göras. En lokal kopia av det som finns i databasen sparas då på användarens hårddisk. När dokumentet sedan har skrivits klart skickar användaren en uppdaterad version av dokumentet till databasen genom att högerklicka och välja *Commit...* samt *Push...*. En ny användare kan nu spara ner den senaste versionen av dokumentet till sin hårddisk.

Viktigt: Om två personer checkar ut samma dokument kan det uppstå en versionskonflikt när dokumentet sedan ska checkas in. Systemet kan hantera sådana situationer, men för att undvika problem så kan det vara bra att försöka undvika att parallellt skriva i samma dokument.

Varje grupp tilldelas ett git-projektkonto vid projektstart och får undervisning i hur det används. Det uppmuntras starkt att använda detta konto för alla projektfiler under kursens gång.

1 En klient för Windows är TortoiseGit som kan laddas ner på: `https://tortoisegit.org`. Det finns även klienter för Linux och Mac OS

3 BEFINTLIG MJUKVARA

Mjukvara för att styra bilbanan från Matlab har utvecklas och består av ett antal funktioner som anropas för att kommunicera med bilbanan.

- `initialize_counters(track)`

  SYFTE: Konfigurerar räknare på kortet PCI6602.

  INARGUMENT TRACK (DOUBLE SCALAR): ’1’ för bana 1, ’2’ för bana 2.

- `start_race(track)`

  SYFTE: Första anropet startar tidsräkningen, d.v.s. alla räknarna börjar att räkna. Efterföljande anrop återställer alla räknare till ursprungsvärdet och återstartar räkningen.

  INARGUMENT TRACK (DOUBLE SCALAR): ’1’ för bana 1, ’2’ för bana 2.

- `set_car_speed(track,speed)`

  SYFTE: Sätter bilens hastighet.

  INARGUMENT TRACK (DOUBLE SCALAR): ’1’ för bana 1, ’2’ för bana 2.

  INARGUMENT SPEED (DOUBLE SCALAR): Antar värden mellan 0 och 100. Värde 0 ger min och 100 ger max pådrag.

- `[add_lap,add_check_point,elapsed_time] = get_car_position(track)`

  SYFTE: Tar reda på om bilen passerat någon varv- eller checkpoint-sensor (alt. antalet passerade sensorer om flera hunnit passeras sedan senaste anropet). Funktionen returnerar även värdet på tidräknaren samt nollställer tidräknaren om någon sensor passerats.

  ATT TÄNKA PÅ: Denna funktion bör anropas minst en gång efter det att en sensor passerats. Anropas denna funktion först efter det att två eller fler sensorer passerats sedan föregående anrop måste hänsyn tas till detta med tidangivelsen i åtanke.

  INARGUMENT TRACK (DOUBLE SCALAR): ’1’ för bana 1, ’2’ för bana 2.

  UTARGUMENT ADD_LAP (DOUBLE SCALAR): Anger om nytt varv har påbörjats sedan förra anropet. Om så är fallet sätts parametern till ’1’, annars till ’0’. Om ett nytt varv påbörjats så nollställs även tidräknaren.

  UTARGUMENT ADD_CHECK_POINT (DOUBLE SCALAR): Anger hur många givare längs med banan som har passerats sedan förra anropet. Om minst en bansensor har passerats så nollställs även tidräknaren.

  UTARGUMENT ELAPSED_TIME (DOUBLE SCALAR): Anger tid i millisekunder. Om en givare har passerats sedan föregående anrop så läses tiden av från räknaren och räknaren nollställs. Tiden returneras. Notera att räknaren inte nollställs då sensorn passeras utan då anropet sker. Dock, det föreligger ingen risk att det totala felet, dvs det fel man får om man lägger ihop tiderna för alla delsträckor, växer med tiden om en givare missas eller om funktionsanropet blir fördröjt. Med en genomsnittlig loop-tid på 0.1 sekunder i huvudprogrammet så blir maximala totala felet således 0.1 sekunder.

- `config_IOs()`

  SYFTE: Konfigurera IO-pinnar för mottagning av manuell hastighet.

- `manual_speed = get_manual_speed(track)`

  SYFTE: Erhålla manuell hastighet.

INARGUMENT TRACK (DOUBLE SCALAR): ’1’ för bana 1, ’2’ för bana 2.

UTARGUMENT MANUAL_SPEED (DOUBLE SCALAR): Antar värden mellan 0 och 127 där 0 är noll i motstånd genom handtaget och 127 är 100 procent motstånd genom handtaget. Notera att detta är omvänt mot den hastighet som matas ut till bilarna. Det kan också vara klokt att notera att gashandtagen har en liten tröskel i början vilket behöver kompenseras för för att få en behaglig körupplevelse.

- `terminate(track)`

  SYFTE: Stoppa bilen och frigör alla räknarresurser. En återinitiering av banan är nödvändig för att åter köra bilen.

  INARGUMENT TRACK (DOUBLE SCALAR): ’1’ för bana 1, ’2’ för bana 2.

4 SYSTEMBESKRIVNING AV LCD-DISPLAY

I bilbanelaboratoriet finns en LCD-display med modellnamnet eDIP320-8 som säljs av Electronic Assembly. LCD-displayen är pekkänslig (touch) och har en upplösning på 320x240. Eftersom displaymodulen är ansluten med en seriell USB-kabel behövs ingen extern strömförsörjning. LCD-displayen är fullständigt grafisk vilket betyder att individuella pixlar kan tändas och släckas. Förutom den grafiska funktionaliteten så är det även möjligt att skriva text direkt på displayen i en terminalmod via en textmarkör. Pekfunktionaliteten är dock inte kapacitiv vilket medför att man kan behöva beröra displayen något hårdare än vad man är van med från exempelvis en mobiltelefon.

En mer fullständig beskrivning av displayen och dess funktionalitet finns beskrivet i den tekniska specifikationen (Assembly, 2011). Detta dokument bör studeras väl av ansvarig person för delsystem 4.

4.1 Datakommunikation

LCD-displayen kan kommunicera med extern hårdvara genom någon av de tre integrerade interfacen; RS-232, SPI, eller I^2C. Används RS-232 interfacet överförs data seriellt och asynkront. Trots att det är vanligt med andra interface som är seriella, exempelvis USB, brukar RS-232 associeras med den 9-poliga D-SUB-kontakten som finns på äldre persondatorer även kallad COM-port. Eftersom denna port blir mer och mer sällsynt på dagens datorer så är det möjligt att göra en omvandling mellan RS-232 och USB via en speciell adapter. Denna adapter är inbyggd i displaymodulen och på så sätt är det möjligt att ansluta displayen direkt via USB-porten. Däremot dyker denna USB-port upp som en COM-port i datorns hårdvaruinställningar. Denna COM-port kan ges ett ledigt nummer mellan 1 och 256.

4.1.1 *Högnivåspråk*

Displayens funktionalitet kan styras genom att skicka ett antal högnivåkommandon via serieporten. Dessa högnivåkommandon gör det enklare att utföra vissa grafiska operationer. En av dessa operationer kan exempelvis vara att rita en linje mellan två punkter. Det räcker då med att ange att en linje skall ritas och att denna linje ska ritas mellan två givna koordinater. Genom att använda högnivåkommandon blir det även relativt enkelt att skriva enstaka tecken, eller strängar som består av ett antal tecken, på displayen.

4.1.2 *Datapaket*

För att skicka/ta emot data och instruktioner till/från displayen samlas dessa instruktioner och data i ett paket, ett s.k. datapaket har bildats. I den tekniska specifikationen (Assembly, 2011) benämns dessa datapaket *small protocol packages*. I detta fall så innehåller datapaket instruktioner och/eller data och består av fyra delar; (1) ett kommando

(<DC1> = 17) eller (<DC2> = 18), (2) längden av instruktioner/data (len), (3) aktuella instruktioner/data (data), samt (4) en checksumma (bcc). En schematisk bild för ett datapaket som skickar instruktioner/data till displayen visas i Figur 1. Checksumman (bcc) beräknas genom att summera (<DC1>), (len), och (data). Blir denna summa

<DC1>	len	data	bcc

Figur 1: Datapaket för att skicka instruktioner/data till displayen.

större än 255 tas modulo 256 på summan. Detta betyder att checksumman alltid har värden som ligger i intervallet [0, 255]. Efter att displayen har mottagit datapaketet kommer den att svara med ett (<ACK>) eller ett (<NAK>). Är datapaketet korrekt överfört kommer (<ACK> = 6). Blev det något fel under överföringen kommer displayen att svara med (<NAK> = 21).

För att mottaga instruktioner/data från displayen skickas ett datapaket som har samma struktur som i Figur 1, men där (<DC2>) används istället för (<DC1>). Instruktionen för att initiera displayen att skicka den data som finns lagrat i det interna minnet är S. Datapaketet som används för att initiera displayen att skicka data kan symboliseras enligt Figur 2. Displayen kommer att svara med (<ACK>) eller (<NAK>). Mottas ett (<ACK>) kommer displayens interna

<DC2>	1	S	bcc

Figur 2: Datapaket för att meddela att instruktioner/data från displayen skall mottagas.

minne att läggas ut på COM-porten och det är möjlig att läsas av innehållet utifrån. Innehållet i displayens interna minnet kommer att paketeras enligt datapaketet som visades i Figur 1.

4.2 Instruktionsupsättning

De instruktioner som displayen kan utföra presenteras väl i den tekniska specifikationen (Assembly, 2011) och återupprepas ej i detta dokument. Instruktionerna kan delas upp i fyra huvudgrupper; (1) terminalkommandon, (2) grafiska kommandon, (3) pekkommandon, och (4) responskommandon. Utseendet för data i respektive huvudgrupp presenteras i nedanstående tabell:

- **TERMINALKOMMANDON** visas i tabellen på sidan 11. Dessa kommandon används för att exempelvis flytta, slå på, och slå av textmarkören. I terminalläget behöver inga instruktioner skickas till displayen när text ska skrivas ut, vilket betyder att data i Figur 1 är aktuell textsträng.
- **GRAFISKA KOMMANDON** visas i tabellerna på sidan 14–15. Dessa kommandon används för att exempelvis rita linjer, rita mönstrade områden, skriva text på en given position.
- **PEKKOMMANDON** visas i tabellen på sidan 16. Dessa kommandon används för att definiera områden på displayen som kan associeras med ett givet nummer (dow code/up code). Dessa nummer kan väljas i intervallet [1,255], men nummer i början av intervallet är associerade med makron. De nummer som är associerade med makron kommer inte kunna användas när displaytryckningar ska registreras i Matlab.
- **RESPONSKOMMANDON** visas i tabellen på sidan 17. Dessa instruktioner används för att exempelvis detektera att ett område på displayen är berört. Dessa instruktioner och data skickas inte till displayen utan motas efter att en läsning av displayens interna minne är initierat. Initieringen genomförs genom att skicka det datapaketet som presenterades i Figur 2.

I så gott som samtliga fall börjar data med ESC = 27, och sedan kommer själva instruktionen följt av efterkommande data.

Det kan påpekas att terminalläget och övriga lägen är helt fristående. Detta betyder att det är möjligt att radera allt innehåll i terminalläget samtidigt som det grafiska innehållet som är ritat i de andra lägena bibehålls och vice versa.

Det finns även ett antal fördefinierade fonter och bakgrundsmönster att välja mellan, se sidan 11–13 i (Assembly, 2011) där även specialtecken såsom å,ä, och ö visas.

4.3 Klient/server-kommunikation i Matlab

Matlab har inbyggt stöd för att kommunicera genom serieporten (COM-porten) och ett s.k. handtag (handle) till porten kan skapas. Matlabkommandona `fwrite` och `fread` kan sedan användas för att skriva och läsa till/från porten. För att stänga kommunikationen användas `fclose`. Nackdelen med att låta Matlab sköta kommunikationen är att Matlab låser sig tills datapaketet är skickat. Detta kan leda till att allt annat som ska göras inte hinns med, trots att seriekommunikationen mest består av väntande för processorn. Därför kan det vara önskvärt att låta en annan process ta över arbetet med att skicka/ta emot datapaket till serieporten och på så sätt utnyttja operativsystemets möjligheter till s.k. multi-tasking.

4.3.1 *Delat minne*

För att underlätta kommunikationen mellan Matlab och displayen har två små program skapats. Dessa två program är en server och en klient. Servern körs i Windows kommandotolk (`cmd.exe`) och klienten körs direkt i Matlabs kommandotolk. Det smidiga med dessa två program är att de har läs- och skrivåtkomst till ett gemensamt minne (shared memory). Klienten kan således skriva datapaketet till det delade minnet och sedan avslutas klienten omedelbart. Samtidigt triggas ett *event* att servern ska läsa från det delade minnet och sedan skicka innehållet till displayen. När instruktionerna displayen ska utföras är färdiga läser servern från displayens interna minne och placerar detta i det delade minnet. Klienten kan sedan läsa innehållet i det delade minnet och returnera detta i Matlab. Eftersom det tar tid att skriva/läsa till/från displayen bör man vänta ca 0.5 s innan något nytt ska skriva/läsa till/från displayen.

4.3.2 *Server*

Servern starts genom att anropa `Server.exe` i Windows kommandotolk `cmd.exe`. Servern tar en flagga `Pcomx` där `x` är aktuell COM-port. Om displayen är ansluten till COM1-porten körs kommandot:

```
Server.exe -Pcom1
```

i aktuell sökväg där `Server.exe` finns. I kommandofönstret kommer innehållet i det delade minnet att skrivas ut i promten. Innehållet kommer att skrivas ut innan datapaketet skickas till displayen samt efter det att displayens interna minne skrivits tillbaka till det delade minnet.

4.3.3 *Klient*

Klienten anropas från Matlab och har följande funktionsbeskrivning:

- `[out, shm] = Client(flag,dataPackage)`

  SYFTE: Klient för att skicka/ta emot datapaket till/från servern från Matlab.

  INARGUMENT `FLAG` (`DOUBLE SCALAR`): Flaggan kan anta 3 värden.

  - Om `flag` = 1, Ett datapaket skickas till det delade minnet. Datapaketet `dataPackage` måste specificeras.
  - Om `flag` = 2, Klienten läser av det delade minnet och sparar innehållet i `out`. Datapaketet `dataPackage` behöver ej anges.
  - Om `flag` = 3, Servern avslutas. Datapaketet `dataPackage` behöver ej anges.

- Om `flag` = 4, Matlabklientens version skrivs ut i promten.

INARGUMENT DATAPACKAGE (DOUBLE VECTOR): Innehåller det datapaket som ska skickas till det delade minnet.

UTARGUMENT OUT (DOUBLE SCALAR): Argumentet används som en enklare kontrollvariabel, d.v.s. att korrekta flaggor har angivits samt att checksumman stämmer. Argumentet antar värden enligt:

- Om `out` = 0, Följande kontroller är godkända; (1) första elementet är <DC1> eller <DC2>, (2) längden på `dataPackage` stämmer med `len`, samt (3) checksumman stämmer. Datapaketet kopieras till det delade minnet.
- Om `out` = 1, Datapaketet är inte korrekt, eller så kan felaktiga flaggor angivits.
- Om `out` = 2, Servern är inte startad om ett datapaket försöker skickas.
- Om `out` = 4, Matlabklientens version skrivs ut i promten.

UTARGUMENT SHM (DOUBLE VECTOR): Om `flag` = 2 så innehåller argumentet en kopia av det delade minnet, annars används inte detta argument.

OBS: Bara för att `out` = 0 behöver detta inte leda till att en instruktion på displayen sker. Det är möjlig att felaktig instruktion har angivits. Displayen kommer i så fall att svara med en (<NAK> = 21) eller skriva ut konstiga tecken på displayen samt i kommandotolken. För att radera de konstiga tecknen på displayen kan dessa raderas genom att använda terminalkommandot för radering beskrivet på sidan 11 i (Assembly, 2011).

4.4 Grundläggande displayexempel

I detta avsnitt presenteras två exempel hur datapaket kan skickas och ta emot från displayen. De två exemplen är direkt hämtade från den tekniska specifikationen (Assembly, 2011).

4.4.1 *Radera och rita en linje på displayen*

För att radera och sedan rita en linje på displayen mellan koordinat (0,0) och (319,239), dvs diagonalt över hela skärmen, kan nedanstående Matlabkod användas. Datapaketet som ska skickas till displayen konstrueras enligt Figur 3. Som syns i figuren initieras en instruktion med `ESC` varpå instruktionen kommer. I detta fall har instruktionen för att

<DC1>	len	ESC	D	L	ESC	G	D	0	0	319	239	bcc
$11	$0E	$1B	$44	$4C	$1B	$47	$44	$00 $00	$00 $00	$3F $01	$EF $00	$9F

Figur 3: Datapaket för att radera och sedan rita en linje på displayen mellan koordinat (0,0) och (319,239).

radera displayen bokstäverna `DL` och instruktionen för att rita ut en linjen bokstäverna `GD`. I det senare fallet kommer koordinaterna direkt efter instruktionen. Eftersom displayen har 320 punkter i sidled räcker det inte med 1 byte = 8 bitar = $2^8 = 256$ för att ange koordinaten, vilket leder till att 2 bytes måste användas. I detta fall kommer koordinaten att anges med den **mest signifikanta byten till höger** och den **minst signifikanta byten till vänster**, vilket är ovanlig representation. Dock så är den interna representationen i varje byte som vanligt, dvs den mest signifikanta biten är till vänster och den minst signifikanta biten är till höger.

Matlabkoden får utseende enligt:

```
% Starta servern manuellt genom kommandotolken cmd.exe Syntax:
% Server.exe -Pcom1 (förutsatt att COM-porten är 1)
%
% Datapaket in hexadecimalt:
```

```
strHex = ['11'; '0E'; '1B'; '44'; '4C'; '1B'; '47'; '44'; '00'; '00'; ...
     '00'; '00'; '3F'; '01'; 'EF'; '00'; '9F'];

     % Anropa Matlabklienten
flag = Client(1, hex2dec(strHex)');

% Kontrollera så att textsträngen skickas till det delade minnet.
if flag ~= 0
    display('Send text string to the server failed!')
else
% Vänta 0.5s för att displayen ska rapportera att datapaketet är
% mottaget korrekt
    pause(0.5)
    [out,shm] = Client(2)
end
```

där en pause på 0.5 s görs för att säkerhetsställa att displayen hinner med att skriva tillbaka (<ACK>) eller (<NAK>) till det delade minnet. Oftast är denna kontroll inte nödvändig att utföras men om allt gått bra kommer ovanstående Matlabkod att returnera följande:

```
shm =

     6
```

4.4.2 *Fråga efter displayens internt lagrade data*

Så fort ett datapaket skickas till det delade minnet kommer servern att läsa av detta och vidarebefordra detta datapaket till displayen. Displayen kommer först att meddela ifall datapaketet överfördes korrekt och sedan utföra given instruktion. Displayen har ett internt minne där exempelvis knapptryckningar lagras. Genom att skicka datapaketet som presenterades i Figur 2 kommer displayen att kopiera innehållet i sitt interna minne till det delade minnet. Eftersom datan nu ligger i det delade minnet kommer Matlabklienten att kunna läsa av detta minne och returnera tillbaka innehållet till Matlab. Matlabkod för att utföra detta kan ha följande utseende:

```
% Starta servern manuellt genom kommandotolken cmd.exe Syntax:
% Server.exe -Pcom1 (förutsatt att COM-porten är 1)
%
% Datapaket in hexadecimalt:
strHex = ['12'; '01'; '53'; '66'];

% Anropa Matlabklienten
flag = Client(1, hex2dec(strHex)');

% Kontrollera så att textsträngen skickas till det delade minnet.
if flag ~= 0
    display('Send text string to the server failed!')
else
% Vänta 0.5s för att displayen ska rapportera att datapaketet är
% mottaget korrekt
    pause(0.5)
    [out,shm] = Client(2)
end
```

Om allt går som det ska kommer Matlabkoden returnera följande:

```
shm =

     6    17     0    17
```

vilket är på samma form som datapaketet i Figur 1, där `<ACK>` $= 6$, `<DC1>` $= 17$, `len` $= 0$, `data` $= []$, och `bcc` $= 17$. Eftersom displayens interna minne är tomt kommer `data` vara en tom mängd som har längd 0.

A KORT MATLABINTRODUKTION

Matlab är ett interaktivt tekniskt beräkningsverktyg som ursprungligen utvecklades för numeriska vektorberäkningar. Matlab har idag mycket högt utvecklat stöd för grafisk visualisering och programmering. Genom att komplettera med fler verktygslådor kan bl.a. regler-, simulering-, optimeringstekniska problem angripas och lösas.

I bilbanestyrningsprojektet används primärt Matlabs programmerings- och informationshanterande funktioner samt ett hårdvarugränssnitt mot bilbanan. För dokumentering och utvärdering kan grafiska visualiseringsfunktioner användas.

A.1 Funktioner

I detta avsnitt nämns några av de funktioner som kan behövas för att implementera ett styrsystem för bilbanan. Vissa av funktionerna är speciellt framtagna för att kommunicera med den specifika hårdvaran, medans andra är standardiserade Matlabfunktioner. Vid implementering får naturligtvis andra funktioner användas.

För mer information om en specifik funktion ges kommandot
`» help <funktion>`
vid Matlabpromten.

CLOCK: Returnerar nuvarande tid.

ETIME: Beräknar tidsskillnader.

FOR: Programmeringsfunktion som används exempelvis när ett antal liknande beräkningar ska genomföras.

IF: Programmeringsfunktion som kontrollerar ifall ett givet villkor är sant/falskt och utför därefter en given operation.

PAUSE: Stoppar programmet för en av användaren vald tid.

PLOT: Representerar data grafiskt. Se även *figure, clf, legend, xlabel, ylabel* och *axis* som är hjälpfunktioner till *plot*.

ROUND: Avrundar givet värde till närmaste heltal.

SWITCH: Programmeringsfunktion som fungerar på liknande sätt som en *if*-sats.

TIC/TOC: Mäter mellan kommandot *tic* till *toc*.

WARNING: Skriver ut en varning till Matlabpromten.

WHILE: Programmeringsfunktion som utför en viss operation så länge villkoret är giltigt.

TYPE: Visar Matlabkoden för en given funktion.

CLEAR: Tar bort en Matlabvariabel ur *workspace*.

A.2 Variabel och datahantering

I Matlab tilldelas variabler med *likamed*-operatorn =, exempelvis genom
`» a = 1 + b`
där `a` tilldelas värdet `1 + b`.

För att på ett smidigt sätt hantera data kan listor (vektorer), tabeller (matriser) eller strukturer (träd) användas. Strukturer är speciellt bra för informationshantering och används i styrsystemet. Strukturvariabler byggs upp som en trädstruktur där grenarna kan vara flervariabla. För mer information skriv `» help struct` i Matlabpromten.

I Matlab kan flera funktioner ha samma namn, s.k. överlagrade funktioner. När en sådan anropas avgör Matlab vilken funktion som passar det givna syftet. När hjälpkommandot används visas dock bara hjälptexten för en av funktionerna. För att erhålla hjälptext till en specifik funktion kan följande kommando användas
`help <version>/<funktion>`
där Matlab talar om vilka versioner som finns om enbart kommandot `help <funktion>` används.

REFERENSER

Electronic Assembly. Control unit 320x240 with intelligence. Technical Specification, August 2011.