

PRH612 Bacheloroppgave

Kommunikasjon mellom mikrokontroller og industriell motorkontroller på autonom 'All Terrain Vehicle': Lone Wolf



IA6-5-20

Universitetet i Sørøst-Norge tar ikke ansvar for denne studentrapportens resultater og konklusjoner.

Emne: PRH612 Bacheloroppgave

Tittel: Kommunikasjon mellom mikrokontroller og industriell motorkontroller på autonom «All Terrain Vehicle»: Lone Wolf

Denne rapporten utgjør en del av vurderingsgrunnlaget i emnet.

Prosjektgruppe: IA6-5-20

Tilgjengelighet: Åpen

Gruppedeltakere: Johan Nicolai Barholt
Sindre Bøe

Veileder: Saba Mylvaganam, Håkon Viumdal, Hans-Petter Halvorsen, USN; Øivind Grønli, Kongsberg

Prosjektpartner: Kongsberg Gruppen, Land systems

Godkjent for arkivering: _____

Sammendrag:

Denne oppgaven handler om hardware/software integrasjon og redesign av kommunikasjon med og styring av motorkontrolleren (MK) i en autonom firehjuling kalt «Lone Wolf».

Grunnlaget for Lone Wolf er en bachelor-oppgave og et sommerprosjekt hos Kongsberg Gruppen, Protech. Her fikk den autonome ATVen sitt opphav i 2019.

To vesentlige utviklingspunkter for prosjektet var måten styresignalene ble overført til MKen og kalibrering av styresystemet.

Prosjektet er begrenset til styresystemet og dets funksjoner. Implementering av «homing mode» skulle sikre automatisk kalibrering og bruk av seriell kommunikasjon skulle gi nye muligheter for videre utvikling av ATVens styring.

Noe av styresystemet sitt hardware ble gjenbrukt, blant annet en Arduino MEGA som er en godt egnet plattform for prototyper og støtter de tiltenkte funksjoner, samt mulige utvidelser av disse. Et nytt kretskort for seriell kommunikasjon og styring ble designet og bygget ved hjelp av «Eagle PCB». Det ble gjort studie av dokumentene fra Festo samt «Festo Configuration Tool» (FCT) for å finne den beste protokollen for kommunikasjon, og den beste løsningen for styring.

Det ble valgt RS-232 som kommunikasjons protokoll som støtter «homing mode» og «profile positioning mode». En MAX232 IC ble implementert på kretskortet og Arduino kode ble skrevet i samsvar med CiA402 standarden.

Lone Wolf har fått et «turn key»-styresystem som kalibrerer seg selv og opererer med seriell kommunikasjon. Dette har resultert i en enklere oppstart prosedyre og bedre grunnlag for videre utvikling av ATVen. Resultater oppnådd vil være en god plattform for videre bruk i fremtidige prosjekter knyttet til ATVen «Lone Wolf».

Universitetet i Sørøst-Norge tar ikke ansvar for denne studentrapportens resultater og konklusjoner.

Course: PRH612 Bachelor Thesis

Title: Communication between microcontroller and industrial motor controller on autonomous «All-Terrain Vehicle»: Lone Wolf

This report forms part of the basis for assessing the student's performance on the course.

Project group: IA6-5-20

Availability: Open

Group participants: Johan Nicolai Barholt
Sindre Bøe

Supervisor: Saba Mylvaganam, Håkon Viumdal, Hans-Petter Halvorsen,
USN; Øivind Grønli, Kongsberg

Project partner: Kongsberg Group, Land systems

Approved for archiving: _____

Summary:

This bachelor thesis report deals with the hardware/software integration and redesign of communication system and control of the motor controller in an autonomous all-terrain vehicle (ATV).

The ATV "Lone Wolf's" corner stones were laid by one bachelor thesis group and one student summer project in then Kongsberg Group, Protech. Its steering system comprised of mainly industrial components. Two of the key areas within the context of our project was the realization of a homing system for calibration and the transfer of setpoints to the motor controller.

Our project is limited to controlling, communicating with, and calibrating the motor and motor controller. Signal processing and other system related functions in the existing version were implemented using Arduino MEGA. Therefore, a communication protocol suitable for the existing Arduino MEGA and the industrial motor controller was selected, viz serial communication. To accommodate this protocol, new communication circuitry had to be designed using "Eagle PCB" and was manually finished. The necessary information for this work was obtained from Festo's documentation for the industrial components and "Festo Configuration Tool" (FCT).

The RS-232 communication protocol was selected for its support of homing mode and profile positioning mode. Communication circuitry was built based on the MAX232 IC, and the CiA402 standard was implemented through Arduino code.

Lone Wolf has now become a turn-key solution with a self-calibrating steering system utilizing serial communication. We are sure that this will provide many interesting possibilities in future developmental projects involving "Lone Wolf".

The University of South-Eastern Norway takes no responsibility for the results and conclusions in this student report.

Forord

Denne rapporten har blitt skrevet som følge av en bacheloroppgave etter tre års studium innen Informatikk og Automasjon ved Universitetet i Sør-Øst-Norge institutt Porsgrunn.

Rapporten representerer studentenes kunnskap og evne til å tilpasse og konfigurere et styresystem på en Autonom ATV eid av Kongsberg Land systems.

Bildet på forsiden er tatt av Sindre Bøe under testing av Lone Wolf sommeren 2019.

Programmer som er brukt for å gjennomføre oppgaven er Arduino IDE, Microsoft (MS) Word, MS Excel, MS Visio, MS Project, MS Teams, Eagle, Festo Configuration Tool (FCT), Terra Term og Notepad++.

Vi vil takke Kongsberg Land systems for å ha stilt ATVen til rådighet for bacheloroppgaven. En spesiell takk til Øivind Grønli som har vært bindeledd mot Kongsberg og veileder for retning oppgaven har tatt.

Takk til veilederne våre Saba Mylvaganam, Håkon Viumdal og Hans-Petter Halvorsen som har veiledet oss til en effektiv og konkret arbeidsoppbygning.

Vi vil berømme veilederne våre og Øivind for god gjennomføring av tilpasningene som ble gjort i forbindelse med Koronautbruddet, der gode løsninger ble funnet slik at bacheloroppgaven fikk best mulig potensiale ut fra situasjonen.

Takk til USN Porsgrunn og Fredrik Hansen som bidratt med arbeidslokale og gode råd. Takk til tidligere arbeidere og samarbeidspartnere på Lone Wolf prosjektet og til Festo kundeservice for god kommunikasjon.

Porsgrunn – 19.05.2020

Nomenklaturliste

ATV	-	«All Terrain Vehicle» (Firhjuls motorsykkkel)
C/C++	-	Programmerings språk
CAN	-	«Controller Area Network»
CiA	-	«CAN in Automation» (CAN i Automatisering)
DA	-	Digital Analog
DC	-	«Direct Current» (Likestrøm)
DIN	-	Digital Inngang
FCT	-	«Festo Configuration Tool»
FHPP	-	«Festo Handling & Positioning Profile»
GPS	-	«Global Positioning System»
HMI	-	«Human-Machine Interface»
IDE	-	«Integrated Development Environment»
I/O	-	«Input/Output» (Inngang / Utgang)
IEC	-	«International Electrotechnical Commission»
I ² C	-	«Inter-Integrated Circuit»
IC	-	«Integrated Circuit» (Integreert Krets)
KP	-	Regulatorforsterkning
LS	-	«Limit Switch» (Grensebryter)
LIDAR	-	«Light Detection And Ranging» (Laserbasert avstands måling)
MK	-	Motorkontroller
NO / NC	-	«Normally Open / Normally Closed»

PCB	-	«Printed Circuit Board» (Kretskort)
PID	-	Proporsjonal Integral Derivat
PLS	-	Programmerbar Logisk Styring
PROFIBUS	-	«Process Field Bus»
PWM	-	«Pulse Width Modulation» (Puls Bredde Modulering)
RC	-	«Remote Controller» (Fjern kontrollert)
RFID	-	«Radio Frequency Identification»
RS-XXX	-	«Recommended Standard-XXX»
RTK	-	«Real Time Kinematics»
RX/TX	-	«Receive/Transmit» (Motta / Sende)
TTL	-	«Transistor-Transistor Logic»
USB	-	Universell Seriell Buss
USN	-	Universitetet i Sørøst-Norge

Innholdsfortegnelse

Forord	4
Nomenklaturliste	5
Innholdsfortegnelse	7
1 .. Innledning.....	11
2 .. Introduksjon	13
2.1 Rettledning til emner	14
2.1.1 <i>Kommunikasjons standard</i>	14
2.1.2 <i>Protokoll</i>	16
2.1.3 <i>RS-XXX fysiske krav</i>	17
2.1.4 <i>Manuell og autonom styring</i>	17
2.1.5 <i>Festo motorkontroller</i>	20
2.1.6 <i>Festo stepper-motor</i>	21
2.1.7 <i>Arduino Mikrokontroller</i>	23
2.1.8 <i>Fysisk forbindelse mellom Arduino og Festo motorkontroller</i>	24
2.1.9 <i>Grensebryter</i>	26
2.1.10 <i>«Homing»</i>	27
2.2 Programkode for styring av motorkontrolleren	30
2.2.1 <i>Organisering av koden</i>	30
2.2.2 <i>CiA402</i>	30
2.2.3 <i>Programspråk</i>	31
2.2.4 <i>Funksjoner for «homing» og styring</i>	31
2.2.5 <i>Arduino Innganger</i>	32
2.2.6 <i>Arduino utganger</i>	32
2.2.7 <i>Festo motorkontroller innganger</i>	32
2.2.8 <i>Festo motorkontroller utganger</i>	32
2.2.9 <i>Alternativer til RS-232 og RS-485</i>	33
2.2.10 <i>Programkode oppbygning</i>	33
2.3 Reguleringsystem	36

2.3.1 Festo innebygd regulator	36
2.3.2 Filter av inngangssignal	37
2.4 Sikkerhet av personell og materiale	38
2.4.1 Innledning	38
2.4.2 Fabrikantens Advarsler	38
2.4.3 Nødstoppbryter og dødmanns knapp.....	38
2.4.4 Fysisk sikkerhet	39
2.4.5 Programmert sikkerhet	39
2.4.6 Fremtidig sikkerhet	39
2.4.7 Eksosgass.....	40
2.4.8 Brann	40
3 ..Metode.....	41
3.1 Litteraturstudie	41
3.1.1 Beskrivelse av relevante dokumenter	42
3.2 Programvare beskrivelse	44
3.2.1 Festo Configuration Tool	44
3.2.2 Terra Term	44
3.2.3 Arduino Integrated Development Environment (IDE).....	44
3.3 Parametrisering med «Festo Configuration Tool»	45
3.3.1 «Configuration»	45
3.3.2 «Application data»	47
3.3.3 «Axis»	49
3.3.4 «Homing»	51
3.3.5 «Closed loop»	52
3.3.6 «Control interface»	53
3.3.7 «Fieldbus»	55
3.3.8 «Error Management»	56
3.4 Programmering	57
3.4.1 Ressurser for koding	57
3.5 Kretskort	58
3.5.1 RS-485.....	58
3.5.2 RS-232.....	58
3.5.3 Digital I/O	59
3.5.4 Fullstendig kretskort	59

3.6 Testtrigg	60
3.7 Tester	61
3.7.1 Fjernstyring av ATV	61
3.7.2 Kommunikasjon mot MK med RS-232	62
3.7.3 Test av «Homing» over RS-232.....	63
3.7.4 Test av «Profile positioning» over RS-232	63
3.7.5 Kretskort for TTL til RS-485.....	64
3.7.6 Kretskort for TTL til RS-232.....	64
4 ..Resultater	65
4.1 Programbeskrivelse	65
4.1.1 Variabler.....	65
4.1.2 Metoder.....	68
4.1.3 Spørringer, løkker og formatering	73
4.1.4 Gjennomkjøring av koden	75
4.2 Resultat av tester	78
4.2.1 Fjernstyring av ATV.....	78
4.2.2 Kommunikasjon mot MK over RS-232	79
4.2.3 Test av «Homing» over RS-232.....	80
4.2.4 Test av «Profile Positioning» over RS-232	80
4.2.5 Parametrisering til RS-485	81
4.2.6 Test av Arduino TTL til RS-232	82
4.3 Valg av kommunikasjons-protokoll	83
4.4 Grensebryter.....	84
4.5 «Homing»	85
4.5.1 Hardware.....	85
4.5.2 Parametere til motorens roterende akse	86
4.6 Kretskort for kommunikasjon mot motorkontrolleren.....	88
4.6.1 Digital I/O på motorkontrolleren	88
4.6.2 RS-485.....	89
4.6.3 RS-232.....	90
4.6.4 Kombinert kretskort for RS-232, RS-485 og digital I/O	91
4.7 Bygging av testtrigg	93
4.7.1 Design.....	93
4.7.2 Montasje.....	93

4.7.3	<i>Komponenter og verktøy brukt i bygging av testrigg</i>	94
5	Diskusjon	99
5.1	Korona situasjonen	99
5.2	Problemer med parametrisering	100
5.3	Videre forbedringer	101
5.3.1	<i>Transistor styring av reléer</i>	101
5.3.2	<i>Ny revisjon av kretskort</i>	101
5.3.3	<i>Avlesning av aktuell informasjon</i>	102
5.3.4	<i>Optimalisering av kommunikasjon og bevegelse</i>	102
6	Oppsummering	103
7	Referanser	105
	Vedlegg	110

1 Innledning

Oppgaven er basert på arbeidet gjort for Kongsberg Land Systems i en tidligere bachelor oppgave, «Guided Wolf», samt et sommerprosjekt ved navn «Lone Wolf» [1] [2].

Prosjektet består av en ATV med et autonomt styringssystem som erstatter føreren. Det fullstendige systemet består av en Arduino MEGA 2560, med tilhørende signalbehandling og flere aktuatorer, MKer og enkodere. For å styre ATVen finnes et system bestående av en påmontert stepper-motor med tilhørende enkoder, samt en Festo MK.

MKen er designet til å brukes med en Programmerbar Logisk Styring (PLS), men er i dette tilfellet styrt av et kretskort. Kretskortet er tilknyttet en Arduino igjennom Inter-Integrated Circuit buss (I²C).

Kretskortet består av en 24 V til ± 12 V likestrøms (DC) omformer, og kombinert med en 10-bit Digital-Analog-omformer (DA) og en inverterende forsterker danner det et ± 10 V analogt styresignal. Styresignalet går inn på en analog inngang i MKen.

Tilbakekobling kommer fra enkoderen til stepper-motoren og har en analog utgang igjennom MKen. Utgangen er 0-10 V DC, men ettersom dette er for høy spenning for de analoge inngangene til Arduinoen, må signalet først igjennom en spenningsdeler. Tilbakekoblingen blir brukt i en proporsjonal-regulator som igjen gir et nytt styresignal.

MKen er av typen FESTO CMMS-ST-C8-7-G2 med mulighet for parametrisering over RS-232, samt styring over blant annet RS-485, Controller Area Network Open (CANOpen) og ProfibusDP.

Parametrisering er metoden som brukes for å finne verdier og stille inn en modul før den tas i bruk for første gang.

Når man parametriserer MKen har man flere valg basert på oppsett av enkoder og stepper-motor, samt styringsmetode som blir benyttet. Velger man rett styringsmetode åpner muligheten seg for å bruke «Homing mode», samt «Profile positioning mode». Da sendes settpunkt direkte til MKen over feltbuss slik at den innebygde regulatoren tar seg av posisjoneringen.

Med analog styring av MKen må posisjonen til stepper motoren og enkoderen kalibreres manuelt for hver oppstart. Kalibreringen utføres ved å manuelt vri styret på ATVen slik at hjulene står i maksimalt utslag mot venstre før oppstart. Dette gir en komplisert oppstarts-prosedyre med mindre nøyaktig styring.

Rapporten omfatter en litteraturstudie av Festo sine manualer og dokumenter for den gjeldende MKen, for deretter å drøfte de forskjellige styremåtene. Oppgaven er å finne en ny metode for kalibrering av stepper-motoren og en ny kommunikasjons protokoll som kan implementeres mot en Arduino. Dette kan innebære utvikling av nye komponenter som må bygges og testes før de kan implementeres.

Bacheloroppgaven omhandler primært styringssystemet innenfor kategoriene hardware og software. Prosjektet skal ikke ta for seg fullstendig implementering av «Festo Handling Program Protocol» (FHPP) eller «CAN in Automation» (CiA402).

Kapittel 2 er en introduksjon til de forskjellige emnene som rapporten fordyper seg i, samt beskrivelse av det som er funnet i Festo sine dokumenter.

Kapittel 3 omhandler metodene som er brukt for å realisere hoved- og delmålene til prosjektet.

Kapittel 4 omhandler resultatene av de forskjellige testene som blir utført, samt resultatet av annet arbeid som har blitt gjort.

Kapittel 5 diskuterer forskjellige aspekter rundt oppgaven som problemstillinger utenfor oppgavebeskrivelsen og videre forbedringer.

Kapittel 6 sammenfatter hva som er blitt oppnådd i prosjektet.

Kapittel 7 inneholder referanser.

2 Introduksjon

I denne rapporten vil «Lone Wolf» og ATVen bli benyttet om hverandre. Lone Wolf er navnet på prosjektet satt i gang av Kongsberg, og består av en ATV med påmontert utstyr.

Som det står beskrevet i forordet og innledningen er ATVen «Lone Wolf» et prosjekt som er satt i gang av Kongsberg sin tidligere avdeling «Protech», nå «Land Systems». Beslutninger som omhandler bruk av komponenter, hardware og software er tatt av tidligere gruppedeltagere under sommerprosjekt og en tidligere bacheloroppgave. Gruppedeltagerne i denne rapporten har fått mulighet til å bytte ut ulike komponenter for å bedre styringen av ATVen.

For å nå målene som er beskrevet i innledningen forventer gruppen å måtte bytte blant annet mottakeren fra Remote Controlleren (RCen), kretskort og nesten alt av kabling. Dette medfører behov for omprogrammering av innsignalene på Arduinoen og valg av annen protokoll for dataoverføring.

Gruppemedlemmene har teknisk bakgrunn og interesse for elektronikk. Noe av det tekniske grunnlaget er basert på egen kunnskap. Underkapitlene i introduksjon gir et innblikk i komponenter, protokoller, standarder, programkode, reguleringsmetoder og sikkerhet.

Rapporten er til dels ment som en introduksjon til emnet for videre studenter i Lone Wolf prosjektet. Disse studentene vil ha varierende kunnskap innenfor emnene som blir presentert og det er derfor bevisst skrevet utfyllende for å gi bedre innsikt.

2.1 Rettledning til emner

Dette kapitlet rettleder leseren igjennom de forskjellige emnene til prosjektet og tar for seg hardware, software, Innganger/Utganger(I/O) og diverse koblinger mellom de to. Hoveddeler som blir forklart er MKen, Arduinoen, stepper-motoren, grensebryter, protokoller og programkode.

2.1.1 Kommunikasjons standard

Som vist i Figur 2.1 støtter MKen flere kommunikasjons standarder. For å ta en beslutning på hvilken man skal velge er det en fordel å se de forskjellige standardene sine fordeler og ulemper. Det er viktig å ha målene til prosjektet og budsjett i bakhodet.

Digital I/O kan brukes som standard, men da bare for å velge en allerede lagret posisjon. Disse posisjonene kan legges inn ved hjelp av Festo Configuration Tool (FCT), men det er en begrensning på antall posisjoner som igjen begrenser oppløsningen man kan bevege motoren med. Det positive med denne standarden er at den utelukkende krever digitale utganger som er både lettvinnt og meget robust.

Analog I/O bruker samme kontakt som digital I/O, altså «X1». Den består av et ± 10 V DC signal som er relativt til omdreiningen til motoren. Ulemper med styremåten er at den kun støtter «torque-» og «velocity-profile». Disse fungerer slik at +10 V er maks moment eller hastighet i en retning, og omvendt. «Homing» er heller ikke støttet av denne standarden. Denne standarden er også relativt enkel å implementere i kode, men krever eget design av kretskort ettersom Arduino ikke støtter høyere enn +5 V eller negative spenninger.

Synkronisering er ikke aktuelt da det krever en annen motorkontroller som den kan motta settpunkt fra.

Forskjellige feltbuss standarder er tilgjengelig for MKen. To av disse, **DeviceNet** og **ProfibusDP**, krever en ekstern modul. Disse er industrielle standarder, men med lite støtte mot Arduino.

DriveBus er ABB sin protokoll for kommunikasjon mot kontrollere. Denne krever eget ABB utstyr.

CANOpen er en protokoll med nettverks topologi, den baserer seg på Controller Area Network (CAN) protokollen. En fordel med denne protokollen er dens mulighet til å ha flere enheter på samme nettverk som alle kan kommunisere. Den er ofte brukt i kjøretøy og har støtte for hele OSI modellen. CANOpen krever en egen Integrated Circuit (IC) for å kommunisere mot Transistor-Transistor Logic (TTL) som Arduinoen støtter, men MKen støtter simulering av CANOpen over RS-232 standarden. Da brukes de samme meldingene, men de sendes i klartekst til «X5» kontakten. Ved bruk av CANOpen støttes «homing» profilen.

RS-485 er meget lik **RS-232**, men støtter høyere hastigheter over lengre avstander. Den er også halv-duplex i motsetning til RS-232 som er full-duplex. MKen bruker samme protokoll, CAN in Automation (CiA 402), for både CANOpen og RS-485, med unntaket at RS-485 har et node nummer som sendes med. RS-485 støtter TTL kommunikasjon igjennom for eksempel en «MAX485» IC. Disse er enkle å kjøpe og implementere med en Arduino. Ved bruk av RS-485 støttes «homing» profilen.

2.1.2 Protokoll

Valg av protokoll er avgjørende for arbeidet videre i prosessen. For at Arduinoen skal sende leselig data til Festo MKen, må dette komme i form av en protokoll. Som vist i Figur 2.1 som er hentet fra «Function and commissioning» manualen [3], bruker MKen «CAN-Interpreter (CiA 402, SDO)» for å kommunisere over RS-485. «Interpreter» er referert til som et dataprogram som utfører datakode direkte, uten at det kreves at det komprimeres til et programspråk [4]. I dette tilfelle er datakoden CiA 402. I kapittel 2.2.2 står det mer informasjon om CiA 402.

2.4.2 Positioning mode: Direct mode, individual record mode, record linking mode and interpolated positioning mode

Control interfaces → page 46

Inputs/outputs								
Digital inputs/outputs DIN/DOU, 24 V								
Analogue input AIN, ± 10 V								
Synchronisation (encoder input), 5 V								
Fieldbus								
DriveBus (Motion Control)								
CANOpen								
PROFIBUS DP								
DeviceNet								
RS485								
Device profile								
- C = CiA 402 (CANopen)								
- CI = CAN-Interpreter (CiA 402, SDO)								
- F = FHPP (Festo)								
				C	F/C	F	F	CI
Operating modes								
Positioning mode (position control) → page 112								
Direct mode → page 115								
Direct application								
					F/C	F	F	CI
Individual record mode → page 119								
Record selection (Positioning record 1...63)								
DIN					F	F	F	
Record linking mode → page 136								
Record selection (Positioning record 1...7)								
DIN								
Record selection (Positioning record 1...63)								
					F	F	F	
Interpolated positioning mode → page 152								
Direct application								
				C	C			

Tab. 2.13 Overview: Positioning mode "direct mode, individual record mode, record linking mode and interpolated positioning mode"

Figur 2.1 Sammenheng mellom protokoll og bevegeses profil [3]

2.1.3 RS-XXX fysiske krav

For å kunne benytte seg av overføring over RS-485 eller RS-232 krever MKen en D-sub 9 kontakt. Arduinoen har kun TTL tilkoblinger, men ved å benytte seg av en MAX485 IC eller MAX232 IC fra Maxim, kan man overføre data over Transmitt (TX) og Receive (RX). For å kunne koble inn en av disse ICene trengs det å produseres et kommunikasjons kretskort.

2.1.4 Manuell og autonom styring

Lone Wolf sin hensikt er å kunne bli styrt av et autonomt system basert på Light Detection and Ranging (LIDAR) og Global Positioning System (GPS).

Under kortere forflytning og funksjonstest vil det være til stor fordel å kunne flytte ATVen med en RC, uten aktivering av det autonome systemet.

Autonom styring

Store norske leksikon definerer ordet autonom som [5]:

«Autonom betyr selvkjørende»

Autonom stammer fra det greske ordet *auto* (selv) og *nomos* (lov) [6]. Utvikling av autonome systemer for kjøretøy har pågått siden 1930-tallet [7] [8]. I andre halvpart av 2010-tallet kom flere bilprodusenter med ulike sensorer som standard i bilene, dette for å støtte opp sikkerhetsfunksjoner som kollisjonsbremsing og varsling for vegmerking. Utviklingen av dette har gjort at flere bilmodeller har kommet med selvstendige styringssystemer og «cruise-control» med avstandsdeteksjon til bilen foran.

Lovgivende i Norge har føreren av Kjøretøyet fullstendig ansvar over hva kjøretøyet foretar seg i trafikken, men det er trolig at flere land vil se på en endring i lovverk i fremtiden når det kommer til autonome kjøretøy. Fordeler som er med å framskyve autonome systemer er blant annet billigere teknologi, «teknologisk reklame» for bilprodusentene, trafikksikkerhet og miljø. Ulempene kan være tap av arbeidsplasser for transportnæringen, personovervåking og skadeomfang ved hacking.

For Lone Wolf er det installert en LIDAR, gyroskop, magnetometer og GPS som sensorer, den har også en NVIDIA TX2 som tar seg av behandling av sensor data og navigasjonsønsker fra operatør.

Lidar sensoren sender ut laser pulser som manifesterer seg som punktskyer på alle objekter innenfor 360° og 100 m avstand [9]. Disse punktene inneholder en posisjon i 3D rommet som kan brukes til å måle avstand og posisjon i forhold til sensoren.

Gyroskopet gjør at ATVen har informasjon om akselerasjon og helling på ATVen mens den forflytter seg. Kombinasjonen av gyro og Lidar i samarbeid kan gi ATVen mulighet til å forflytte seg mot et mål dersom systemet skulle miste kommunikasjon fra GPSen og ordre utenifra.

Magnetometeret blir brukt til å finne kompassretningen til ATVen, som deretter reguleres med kontroll systemet og en Proporsjonal Integral Derivat (PID) regulator [2].

GPS i samarbeid med Real Time Kinematics (RTK) gir posisjonen i lengde og bredde grader med en potensiell nøyaktighet helt ned til \pm én cm [10].

Som det står beskrevet i Lone Wolf 2019 rapporten [2], er det i startfasen av prosjektet ment at ATVen skal navigere til en nærliggende GPS posisjon og stoppe for eventuelle hindringer som skulle oppstå. I påfølgende faser skal ATVen være i stand til å navigere til forskjellige GPS posisjoner der den styrer seg selv mot målet og unngår hindringer.

Dataen som går mellom Arduinoen og det autonome systemet vil primært være settpunkt for retning av ATVen, gasspådrag samt bremsepådrag. Tilbake til det autonome systemet vil informasjon om hjulhastighet og den reelle styreposisjonen bli oversendt. Dette er for å kunne muliggjøre mer med nøyaktig styring basert på tregheten i systemet.

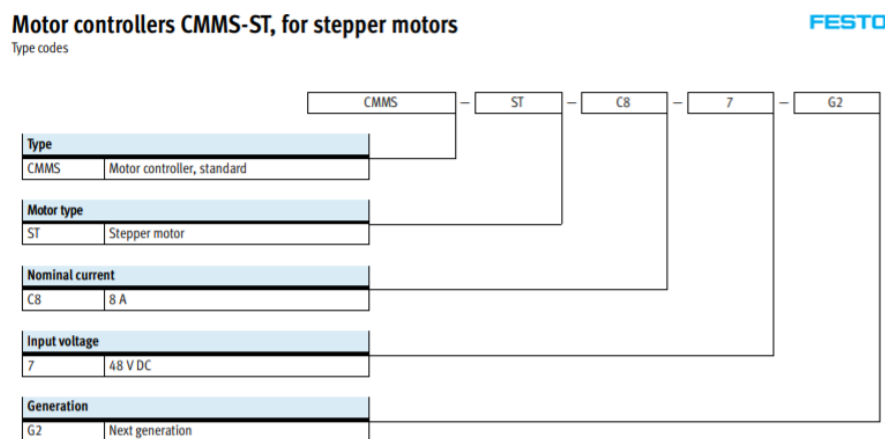
Manuell styring

En RC brukes for å kunne forflytte ATVen enkelt, med full manuell kontroll fra operatøren. Systemet består av en RC av typen «Spektrum DSMX DXe» og en «Spektrum 4 kanals» mottaker. Mottakeren konverterer RC signalet til et pulsbreddemodulert (PWM) signal. Arduino kan lese av PWM signaler og måle lengden mellom hver stigende flanke i millisekunder, denne verdien følger posisjonen til stikken på RCen.

For ATVen er det en separat kanal for styresignal og en annen for pådrag, som består av gass og brems. Detaljer om koden omdanner signalet står beskrevet i kapittel 4.1.

2.1.5 Festo motorkontroller

MKen som blir benyttet er en «CMMS-ST-C8-7-G2» fra Festo. På side 5 fra datablad om «CMMS-ST-C8-7-G2» [11] vises det til at det er en andre generasjons MK for stepper-motorer. Figur 2.2 og Figur 2.3 viser utklipp av hvordan informasjonen står oppført. MKen krever 8 A og står merket for å ha en innspenning på 48 V DC. I tabellen for Electrical data i samme datablad står det at den skal ha en «Load Supply – Nominal voltage» på 24 V til 48 V, se side 7. Under sommerprosjektet har MKen vært tilført 24 V DC fra en «Victron Orion 12/24V-20» omformer koblet til et eksternt 12 V batteri. Les mer om dette i Lone Wolf 2019 rapporten [2].



Figur 2.2: Kode for kontrollert navn [11]

Load supply	
Nominal voltage	[VDC] 24 ... 48

Figur 2.3: Spenning tilførsel [11]

2.1.5.1 Prosjektets utgangspunkt

MKen befinner seg inne i instrumentskapet på ATVens venstre side. Signal- og strømledning til stepper-motoren og enkoder er produkter som er kjøpt og designet av Festo. Ledninger mellom MKen og Arduinoen er en kombinasjon av en digital I/O kabel laget i et tidligere prosjekt samt en nullmodem-kabel laget i denne bachelor oppgaven. En nullmodem-kabel er en kabel der TX og RX i kontakten er krysset fra side til side.

2.1.6 Festo stepper-motor

Stepper-motoren som blir benyttet er en «EMMS-ST-87-L-SE-G2» fra Festo [12]. Figur 2.4 viser til at motoren er en annen generasjons stepper-motor, med rett tilkoblingspunkt og enkoder. Den er en lang versjon med 87 mm boltbredde for innfestning.

Type codes

001	Series	005	Electrical connection
EMMS	Motor	S	Straight plug
002	Motor type	006	Measuring unit
ST	Stepper motor ST		None
		E	Encoder
003	Flange size, motors	007	Brake
28	28		None
42	42	B	With brake
57	57		
87	87		
004	Length	008	Generation
S	Short		1st generation
M	Centre	G2	2nd generation
L	Long		

Figur 2.4: Kode for motor navn [12]

Det er montert en gir modul med 9:1 ratio. For hver 9. rotasjon av motoren så vil tannhjølet rotere en omdreining. Dette gir en enda mer nøyaktig styring av ATVen samt mer moment.

Motoren yter 65 N inn på gir modulen, multiplisert med en 9:1 gir ratio gir dette 585 N. Dette blir deretter overført igjennom et system som består av to tannhjul i samme størrelse og et kjede. Momentet som blir påført styrestaget er derfor 585 N, minus noe dynamiske tap.

Dette kan sees i Figur 2.5:



Figur 2.5 Bilde av de to tannhjulene på ATVen (privat foto)

Enkoderen som er på motoren er av typen «inkrementell» som gir en dynamisk posisjons endring, i motsetning av en «absolutt» enkoder som har ett fast referanse punkt. Dette betyr at motoren ikke husker posisjonen sin om den mister spenning og dermed må kalibreres på nytt for hver oppstart. Enkoderen er til for å signalisere posisjonen til rotoren i forhold til polene. Stepper motoren har en oppløsning på 4000 steg per omdreining når den brukes i kombinasjon med en CMMS ST MK [13]. Dette gir en oppløsning på $0,09^\circ$ mellom hvert steg. Når man vurderer en girkasse med 9:1 ratio i tillegg resulterer det i en oppløsning på $0,01^\circ$ per steg.

2.1.7 Arduino Mikrokontroller

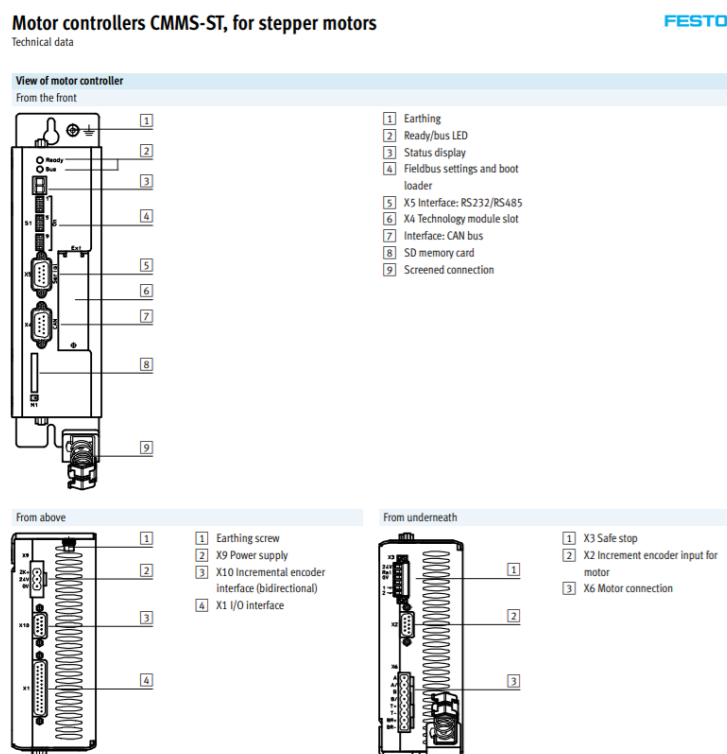
Arduino er en produsent av mikrokontrollere og utviklingsplattformer med flere ulike modeller. Modellen som blir benyttet i ATVen er en Arduino MEGA 2560, som skiller seg fra grunnmodellen Arduino UNO med at den har flere innganger og utganger, samt at UNOen bare har et sett med TX og RX porter der MEGAen har fire sett [13]. Arduino baserer mikrokontrollerne på «open-source» hardware og software, med kode i C++ programspråk [14]. Dette har bidratt til stor interesse i det private markedet som i dag har et stort nettverk med personer som legger ut prosjekter og koder.

Arduino ble valgt fra oppstart av prosjektet grunnet dens allsidige bruksområder, mulighet for utvidelse samt bruken av C++ programspråk. Det er også fra tidligere fase i prosjektet blitt brukt en Arduino for signal behandling til blant annet bremses, styring og gass.

2.1.8 Fysisk forbindelse mellom Arduino og Festo motorkontroller

På sommerprosjektet ble MKens «X1» port benyttet. Denne er reservert til digitale innganger og utganger. Festo MK har en seriell inngang som heter «X5 Serial interface: RS-232/RS-485». Den blir benyttet ved parametrisering og gir mulighet for kommunikasjon over RS-232 eller RS-485. Festo sin visuelle fremstilling av portene vises på Figur 2.6. Arduino har ikke mulighet til å sende ut på RS-232 eller RS-485 uten hjelpemidler. Hvis en MAX232 eller MAX485 IC blir koblet ut fra Arduinoen, vil denne kunne kommunisere med MKen.

Det er tre innganger på MKen som må aktiveres for å få bevegelse på motoren. Disse er på +24 V DC mens Arduinoen kun har +5 V DC. Reléer er tidligere brukt for å ta seg av dette problemet. Figur 2.7 viser en I/O liste for Festo sin MK.



Figur 2.6: CMMS-ST Motor kontroller tilkoblinger [11]

Pin	Value	Mode 0 – positioning (single record)
1	SGND	0 V Screening for analogue signals
2	DIN 12	– Mode bit 0 = "0"
	AI _{n0}	max. 30 V Differential analogue input (setpoint input 0) ²⁾
3	DIN 10	– Record selection bit 4 (high active)
4	+VREF	+10 V ±4 % Reference output for setpoint value potentiometer
5	–	–
6	GND24	– Reference potential for digital I/O modules
7	DIN 1	– Record selection bit 1 (high active)
8	DIN 3	– Record selection bit 3 (high active)
9	DIN 5	– Controller enable (high active)
10	DIN 7	– Limit switch 1
11	DIN 9	– Mode bit 1 = "0"
	DIN 9	– High-speed input (sample) ³⁾
12	DOU1	24 V 100 mA Motion complete (high active) ¹⁾
13	DOU3	24 V 100 mA Common error (low active) ¹⁾
14	AGND	0 V Reference potential for analogue signals
15	DIN 13	R _i = 20 kΩ Stop (low active)
	#AIN0	– Reference potential for setpoint input 0 ²⁾
16	DIN 11	– Record selection bit 5 (high active)
17	AMON0	0 ... 10 V ±4 % Output: analogue monitor 0
18	+ 24 V DC	24 V 100 mA Output: 24 V DC, looped through from [X9.2]
19	DIN 0	– Record selection bit 0 (high active)
20	DIN 2	– Record selection bit 2 (high active)
21	DIN 4	– Output stage enable (high active)
22	DIN 6	– Limit switch 0
23	DIN 8	– Start for the positioning procedure (high active)
24	DOU0	24 V 100 mA Output: Controller ready for operation (high active)
25	DOU2	24 V 100 mA Start acknowledged (low active) ¹⁾

1) Default setting, configurable in the Festo Configuration Tool (FCT).

2) Pin allocation with control via analogue input

3) Pin allocation for flying measurement

Tab. 4.5 Pin allocation of the I/O interface [X1], positioning (single record)

Figur 2.7: Digitalt I/O Skjema for motorkontrollerens «X1» kobling [15]

2.1.9 Grensebryter

For å kunne gjennomføre en «homing» prosedyre trenger MKen et referansepunkt på en fast posisjon som kan brukes til å kalibrere mot.

På side 117 i dokumentet som omhandler «Function and commissioning» [3] er det beskrevet de forskjellige «homing» metodene som støttes av MKen. I flere av metodene beveger motoren seg med konstant hastighet i en retning til den treffer grensebryteren. Deretter beveger den seg sakte tilbake igjen for å finne det eksakte punktet hvor bryteren aktiveres.

Bryterne finnes i mange former, og det vil være hensiktsmessig å finne en som ikke vil ta skade om stepper-motoren trykker for hardt. Den må også støtte spenningsnivået til MKen.

Bryteren har ofte to forskjellige aktiverings metoder, «Slow action» og «Snap action» [16]. «Snap action» sørger for at det er et klart og tydelig punkt der bryteren går fra av til på. Dette da bevegelsen fra av til på skjer kjapt, og bryteren hopper til en på posisjon. Dette skjer på grunn av bruk av en fjær mekanisme. Den samme oppførselen skjer på veg tilbake. En positiv effekt av dette er minimalt med gnister og et veldig klart og tydelig signal. En negativ effekt er at det ikke vil være en presis måte å overføre bevegelse til posisjonering, siden «hoppet» av bryteren kan ha hysteresis.

«Slow action» er en bryter uten en fjær mekanisme der de ytre påkjenningene påvirker bryteren direkte. Dette resulterer i en veldig presis, repeterbar posisjon der bryteren er på, og samme med av. Negative konsekvenser kan være støy på signalet som følge av den treige aktiveringen av bryteren.

2.1.10 «Homing»

«Homing» er metoden for å finne nullpunktet til en mekanisk aktuator med utgangspunkt i en vilkårlig posisjon i dens bevegelsesrom.

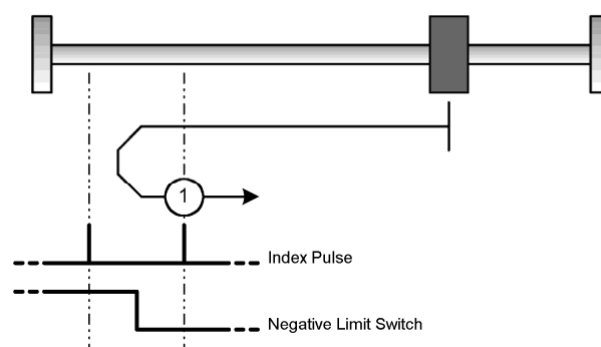
«Homing» gjennomføres typisk etter hver oppstart av en aktuator før den blir operativ. Involvert i «homing» prosedyren er ofte enkoder, med eller uten «zero pulse», og én eller flere grensebrytere. «Zero Pulse» er et punkt på enkoderen til motoren som markerer én omdreining [17].

Festo sin MK har flere forskjellige metoder for å gjennomføre «homing». Disse metodene fungerer for begge bevegelses retninger til aktuatoren og er beskrevet i CANOpen dokumentet på side 120 [18]. Noen aktuelle metoder er beskrevet nedenfor:

Negativ grensebryter med «zero pulse»

I Figur 2.8 vises det at aktuatoren beveger seg hurtig i negativ retning frem til den treffer grensebryteren. Deretter beveger den seg sakte i positiv retning til den finner første «zero pulse», dette blir referansepunktet.

Med denne metoden passer en grensebryter med «snap action» bra, ettersom det ikke trengs presisjon. Presisjonen kommer av «zero pulsen», bryteren er kun et klart signal om at grensen er nådd.

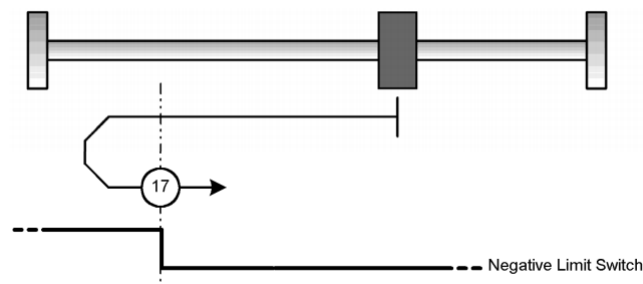


Figur 2.8 Negativ grensebryter med «zero pulse» [18]

Negativ grensebryter uten «zero pulse»

I Figur 2.9 vises det at aktuatoren beveger seg hurtig i negativ retning til den treffer grensebryteren. Deretter beveger den seg sakte i positiv retning til den finner den eksakte posisjonen til grensebryteren. Denne posisjonen blir referanse punktet.

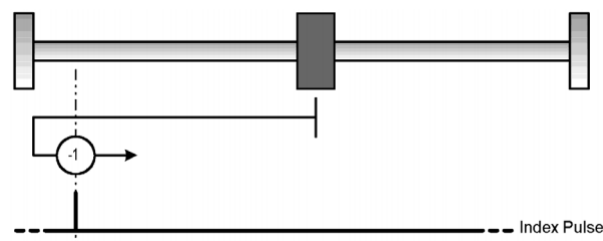
Her passer en «slow action» grensebryter bra ettersom det trengs presisjon.



Figur 2.9 Negativ grensebryter uten «zero pulse» [18]

Negativ stopp med «zero pulse»

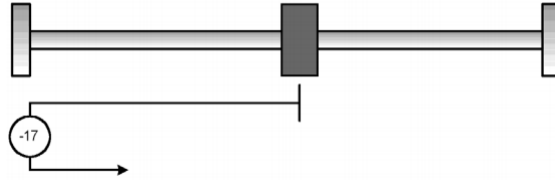
I Figur 2.10 vises det at aktuatoren beveger seg i negativ retning til den blir stoppet av en mekanisk grense. Integral verdien i regulatoren stiger deretter til 90 % av maksimal verdien før den beveger seg i positiv retning frem til første «zero pulse». Denne posisjonen blir brukt som referanse punkt. For at denne metoden skal kunne brukes må den mekaniske grensen være sterk nok til å kunne motstå kreftene til motoren uten å skades. Denne metoden sammen med «Negativ stopp uten «Zero pulse» passer dårlig til ATVen ettersom rattstammen ikke er dimensjonert for slike krefter.



Figur 2.10 Negativ stopp med «zero pulse» [18]

Negativ stopp uten «zero pulse»

I Figur 2.11 vises det at aktuatoren beveger seg i negativ retning til den blir stoppet av en mekanisk grense. Integral verdien i regulatoren stiger så til 90 % av maksimum verdien før posisjonen brukes som referanse verdi.



Figur 2.11 Negativ stopp uten «zero pulse» [18]

2.2 Programkode for styring av motorkontrolleren

Som beskrevet i kapittel 2.1.2 blir det benyttet kode knyttet til International Electrotechnical Commission (IEC) standarden av CiA402. Koden som sendes blir kalt «ordre». Ulike ordre som sendes får MKen til å gjennomføre oppstart, innstilling, kalibrering og operasjon MKen. Det er utarbeidet en mer omfattende beskrivelse av selve programkoden og oppsettet i kapittel 4.1.

2.2.1 Organisering av koden

I Festo sine dokumenter som omhandler «Function and commissioning» står det at ved oversending av kode over RS-485 brukes det samme ordre som RS-232, men med tillegg at den sender oversendingskode «XT» og MKens node-nummer [19] side 38. Ved bruk av «broadcast-adresse» «00» vil ordren gjelde for alle tilkoblede enheter.

2.2.2 CiA402

CiA er en «Nonprofit» organisasjon som ble startet i 1992 av 8 firmaer. Organisasjonen arbeider med å standardisere ulike aspekter rundt CAN. Organisasjonen CiA nesten 700 medlemmer verden over, der majoriteten av medlemmene er lokalisert i Tyskland. Deres hovedkontor i dag er plassert i Nuremberg, Tyskland [20]. CiA402 er delvis standardisert for IEC 61800-7 serien som omhandler «power drive systems». CiA402 er standarden som Festo benytter seg av ved kommunikasjon med RS-232 og RS-485, der informasjon om hvordan signalet overføres står forklart i MKen sine manualer på Festo sine hjemmesider [21]. Mer informasjon fra disse manualene blir beskrevet i kapittel 3.1.

2.2.3 Programspråk

Som det står beskrevet om Arduinoen i kapittel 2.1.7, bruker den en versjon av C++-programspråket. Programspråket gir programmereren mulighet til å lage egne funksjoner og metoder eller benytte seg av løsninger laget av andre.

Historien til programspråkfamilien C/C++ stammer tilbake til 1972 da Dennis Ritchie utviklet et programspråk for å kunne brukes i forskjellige systemer som UNIX operativsystem [22]. Programspråket har hatt en sterk utvikling siden den gang. Parter innenfor matematikk, teknologi, romfart og mange flere har bidratt for utviklingen av mange programspråk vi har i dag. C familien er en av de mest kjente i dag, da det har vært fordelaktig for data, IT og industri. Mikroprosessorer med brukervennlig grensesnitt har fanget interessen til mange for privat bruk. Dette har ledet til at valget av mikroprosessor falt på den svært populære Arduino MEGA2560.

Oppbygningen i programspråket til en Arduino er fordelt i tre hovedkategorier: metoder, verdier og struktur. Figur 2.12 viser Arduino sin formulering [23], før de har alle elementene kategoriene omfatter lengre ned på siden.

Language Reference

Arduino programming language can be divided in three main parts: functions, values (variables and constants), and structure.

Figur 2.12: Forklaring fra Arduino, om programspråk (C++) [23]

2.2.4 Funksjoner for «homing» og styring

Det er ønskelig at styresystemet som bytter ut dagens system blir så enkelt så mulig å integrere. Derfor må koden som skrives integreres med koden som allerede eksisterer for gass og brems. Forventede nye funksjoner blir en funksjon for start av MK, en funksjon for «homing» og en funksjon for styring.

2.2.5 Arduino Innganger

ATVen er fra før av satt opp for to typer styring. Et autonomt system over Seriell- og en PWM inngang fra en RCer. Denne funksjonaliteten ønskes det å beholde, gjerne slik at de kan brukes om hverandre. Den serielle inngangen må settes opp etter hvordan det autonome systemet ønskes å bygges i fremtiden. Eventuelle andre innganger vil være fra sensorer på ATVen som faktisk styrevinkel og faktisk hastighet.

Alle innganger må behandles slik at oppløsningen på signalet ikke reduseres og slik at det er så enkelt som mulig å bruke signalet i fremtiden.

2.2.6 Arduino utganger.

Arduinoen er et mellomledd for signalbehandling fra enten RCen eller det autonome systemet. Derfor vil alle utganger fra Arduinoen være en funksjon av en eller flere av disse inngangene. I all hovedsak vil alle utganger være TTL eller digital I/O. Noe mer signalbehandling kan være nødvendig for å bruke signalet videre, men dette må gjøres på det eksterne kretskort.

For å feil søke kan en USB-kabel benyttes. Denne kobles til Arduinoen for å lese relevante verdier via «Serial» som er «Serial0».

2.2.7 Festo motorkontroller innganger

MKen har en 25 pins digital I/O tilkobling ved navn «X1» som har designerte porter for hvert digitale signal som kommer til MKen. MKen er avhengig av tre innganger for å kunne aktivere rotasjon av motoren. Eventuelle settpunkter som sendes til MKen må sendes som et analogt signal over «X1» eller som seriell kommunikasjon over «X4» eller «X5».

2.2.8 Festo motorkontroller utganger

Utgangene til MKen brukes i all hovedsak til Human Machine Interface (HMI), alarmer eller tilbakekobling. Det eksisterer flere digitale utganger på «X1» for dette, ellers er alle andre utganger over den serielle protokollen.

2.2.9 Alternativer til RS-232 og RS-485

MKen har flere muligheter for å bli tilsendt settpunkt for hver av de mange bevegelses metodene. En oversikt over alle mulighetene finnes på side 40 i «Function and commissioning» Manualen [3].

Alle de forskjellige metodene støtter forskjellige sett med funksjoner. For eksempel vil ikke «homing» funksjonen være aktiverbar om man bruker analog input. De forskjellige serielle kommunikasjonene som går over kontakten «X5» er RS-485 og RS-232. Det er også mulig å simulere CANOpen og bruke alle dens funksjoner over RS-232.

2.2.10 Programkode oppbygning

Under koding av et program blir det brukt ulike verdier, funksjoner og metoder. Det finnes ingen lover på hvordan et program skal bygges opp, men å sette verdier og funksjoner i feil rekkefølge kan få store følgefeil. Derfor er det vanlig å legge inn biblioteker og globale variabler i toppen av programmet. Neste som kommer er koden i midten, før metodene kommer i slutten av programmet. Ved lengre kode, eller ved gjentakende prosesser i koden er det vanlig å benytte seg av ferdigkomponerte metoder som ligger i biblioteket eller skrive sine egne. Dette gjør programmet mer leselig og brukervennlig når programkoden blir lang. Hvis rekkefølgen ikke stemmer i programmet kan feil som «bruk av utdatert verdier» eller «finner ikke verdier» forekomme [24].

Verdier og formater

C++ har flere forskjellige formater for verdier. Relevante verdier for dette prosjektet er «Int» (heltall 16-bit), «long» (heltall 32-bit) og «String» (tekst). «Int» er i koden brukt som adressering og verdilagring. «long» kan brukes til verdilagring der «Int» sine 16-bit ikke er tilstrekkelig. Alternativer kunne vært å bruke «uint32_t», som er «Int» med 32-bit. Forskjellen er at «long» går fra $\pm 2\,147\,483\,647$, mens «uint32_t» går fra 0 til 4 294 967 295.

Når man skal opprette en verdi har man valget mellom å opprette den globalt for hele programmet eller lokalt i metoden eller funksjonen den omhandler. Globale verdier brukes gjerne når verdien skal kunne brukes gjennom hele programkoden. For verdier som inkrementeres som

eks «for-løkke» er det i de fleste tilfeller best å ha de lokalt, da de kan benyttes på nytt i andre metoder uten å skape problemer.

Benevning og «leading zeros»

For at programmet skal forstå om tallet er i binær-, desimal-, oktal- eller heksadesimal form, kreves det en benevning i forkant av tallet, med unntak av desimal som er standard tallformat. Binære tall starter med «B», oktale tall med «0» og heksadesimale med «0x». Ut fra oppgavebeskrivelsen er det ikke behov for å benytte seg av binære eller oktale tall. Når en tallverdi skal skrives som «String» godtar Arduinoen at man skriver inn desimale verdier. eks: *String = (desimaltall)*, da dette også gjør det lesbart ut på seriell utgang. For heksadesimale krever det først en benevning foran tallet og en konvertering i slutten av «stringen» eks. *String = (0xHeksadesimaltall, HEX)*.

«leading zeros» er et kallenavn på nuller som står foran et tall. For flere datatyper er det svært viktig at antall nuller foran tallet stemmer. Arduinoen har en innebygd funksjon som fjerner «leading zeros». Dette skaper problemer for ulike komponenter som krever fullstendige data-signaler. Festo sin MK krever over RS-232 fullstendige datasignaler for å forstå en ordre som blir sendt over. Årsaken til at «leading zeros» forsvinner er at tallet komprimeres og det kreves færre bit når «leading zeros» fjernes. Dette kan løses ved å legge inn kode som tar høyde for manglende nuller i programmet.

Metoder

Når metoder opprettes, velger man i benevninga retur funksjonen til metoden. Ved å skrive «void» returneres ingenting. Dersom metoden benevnes som «Int» returneres det en «Int» tallverdi når den er gjennomført.

For å lese inn verdier som brukes om inngangsparametere i metoden, benevnes disse med format i parentes eks. *void Multipliserings Metode(Int tallverdi)*.

Funksjoner

I Arduinoen finnes innebygde funksjoner som «map», «constrain» og «pulseIn». Disse og flere andre ligger som standard funksjoner som ikke har behov for å kalles opp gjennom et bibliotek. «map» bidrar til å fordele en tallverdi til en annen tallverdi eks. *map(0, 1000, 0, 10)*. Her blir

verdien 0 til 1000 fordelt mellom 0 til 10. Formelen som ligger til grunn for metoden er vist i formel (2.2-1) og hentet fra Arduino sine hjemmesider [25]:

$$x = \frac{(y - in_{min}) * (out_{max} - out_{min})}{(in_{max} - in_{min}) + out_{min}} \quad (2.2-1)$$

'y' er i dette tilfellet inngangsvariabelen og 'x' er utgangsvariabelen.

Dette gjør det mye enklere ved bruk av kompliserte tall eller ved invertering av verdier.

«constrain» funksjonen begrenser tallmengden den omhandler. Dette er for at ikke tall skal under- eller overskride ønsket verdi.

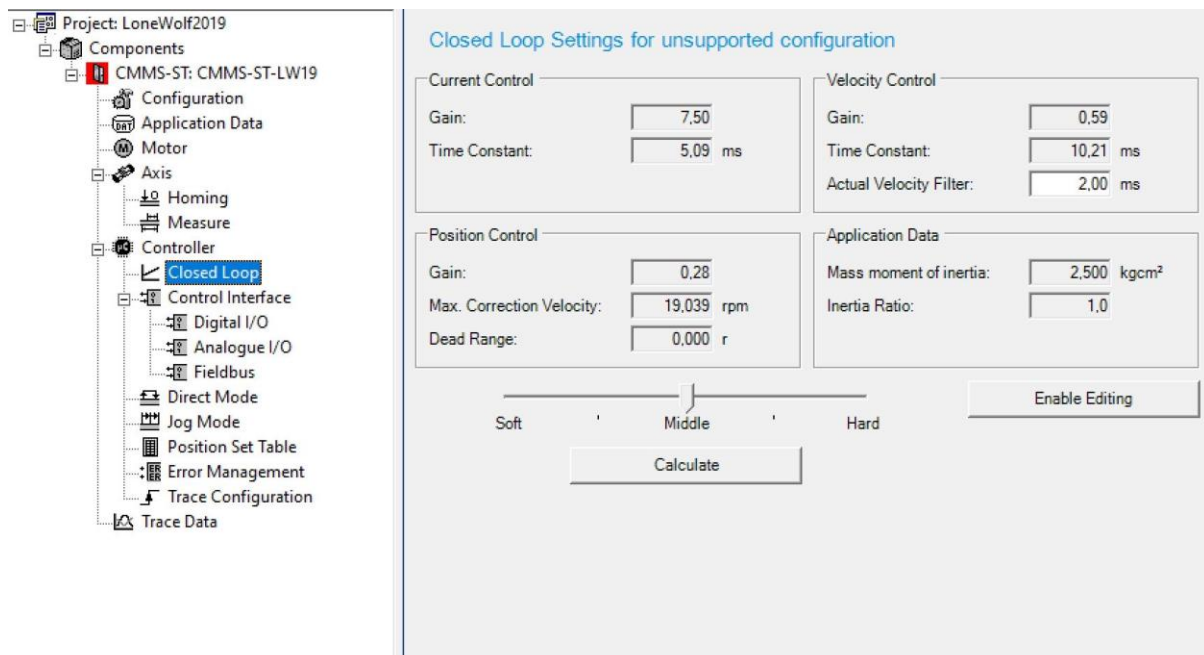
«pulseIn» brukes for å lese av en pulserende verdi. Dette gjøre den ved å lese av mellomrommet i millisekunder mellom hver gang signalet settes til bestemt høy eller lav.

2.3 Reguleringsystem

Reguleringsystemer bidrar til å kunne stabilisere utgangssignalet basert etter eget eller andres ønske. For å få til dette kan man benytte seg av forskjellige reguleringsmetoder, regulatorer eller filtre. Bacheloroppgaven om «Guided Wolf» i 2019 har beskrevet ulike relevante PID regulatorer i sin rapport [1].

2.3.1 Festo innebygd regulator

Den interne regulatoren i MKen har mulighet for manuell parametrisering. En enklere måte å stille den inn korrekt er å stille inn den ønskede oppførselen til stepper-motoren, ved hjelp av «slideren» i Figur 2.13, samt legge inn det mekaniske treghets momentet. Da vil FCT selv regne ut resten av parameterne. Dette blir beskrevet nærmere i kapittel 3.3.5.



Figur 2.13 Parametere CMMS-ST interne regulatorer (Skjerm bilde fra FCT)

2.3.2 Filter av inngangssignal

Et filter blir brukt for å fjerne støy eller å jevne ut en verdi. Det kan ofte føre med seg treghet, manglende nøyaktighet og dårligere oppløsning. Filtre kan være svært enkle med bare en enkel inngang med dempning. De kan også være svært komplekse med flere innganger fra flere punkter i en prosess. Spesielt for signaloverføring er riktig bruk av filtre viktig for å få et anvendelig signal ut. Støy kan oppstå ved overføring av radio frekvensen eller i kabling nær andre spenningsførende deler. Derfor er valg av filter svært viktig for å få en stabil prosess.

2.4 Sikkerhet av personell og materiale

USN, Kongsberg og gruppemedlemmene tar sikkerhet på alvor. Dette kapittelet skal opplyse om farer rundt ATVen, barrierer og brann.

2.4.1 Innledning

ATVen er fra før av sterkt modifisert med tanke på dens opprinnelige bakgrunn. Det ble montert flere aktuatorer, motorer og mekaniske innretninger som endret ATVens handlemåte. En operatør kan ikke påvirke styring eller bremses fysisk, kun via RC-kontrolleren og programkode. Eneste unntakene til dette er Nødstoppbryteren og dødmanns knappen som slår av forbrenningsmotoren og styresystemet.

Faremomenter:

- Klemfare
 - Drev og kjede ved styring
 - Brems aktuatorer
- Påkjørsel
- Brann og brannskader
- Eksosgass

2.4.2 Fabrikantens Advarsler

I brukermanualen hentet fra Fabrikantens hjemmeside [26] er det gjennomgått hvilke forhåndsregler man må foreta seg før man operer ATVen. Dette går ut på blant annet at man skal bruke godkjent hjelm, unngå å kollidere og sette seg inn i manualen før man tar ATVen i bruk.

2.4.3 Nødstoppbryter og dødmanns knapp

Det er montert en nødstoppbryter for styresystemet som bryter 12V tilførselen fra batteriet. Denne er separat fra ATVen sitt system som har en dødmanns knapp. Dødmanns knappen er av samme typen som påhengsmotor på mindre båter. Denne har som funksjon at hvis ATVen kjører fra den som er sikkerhetsansvarlig eller den går løpsk, kan man dra i snora og ATVen

vil skru seg av. Dødmanns knappen skal derfor alltid være festet til den som har sikkerhetsansvaret for ATVen når tenningen er aktivert. Under operering av ATVen vil en person gå ved siden av med snora festet til seg.

2.4.4 Fysisk sikkerhet

At operatøren aldri befinner seg på ATVen legger til et lag i sikkerhetsbarrierene. Ved oppstart vil ATVen kjøre en kalibrering som gjør at hjulene svinger fra side til side. Ved dette er det risiko for å komme i kontakt med drevet som tar seg av styringen eller hjulene.

For å unngå klemfare ved drev og kjede er det plassert et plastdeksel over dem. Dette hindrer at man kan henge seg fast dersom man skulle være uheldig.

Aktuatorene for bremsene befinner seg på en hensiktsmessig lokasjon i nærhet av opprinnelig posisjon for manuelle hendler. Hendelen til håndbremsen er fjernet, noe som bidrar til redusert klemfare.

2.4.5 Programmert sikkerhet

I oppstart av programmet kan det legges inn en tidsforsinkelse i Arduinoen slik at operatøren får tid til å komme seg i sikker avstand. Det er antatt at 5 sekunder er tilstrekkelig tid fra tenningen er skrudd på til hjulene starter å bevege seg.

2.4.6 Fremtidig sikkerhet

I rapporten fra Lone Wolf 2019 sitt sommerprosjekt står det at det autonome systemet vil benytte seg av LIDAR og radar til å detektere objekter som oppholder seg foran ATVen [2]. Dette vil medføre at ATVen stopper innenfor en sikkerhetsmargin. I fremtiden kan det også bidra til å øke sikkerheten ved bruk av RC, da det autonome systemet kan forhindre kollisjon ved menneskelig svikt.

2.4.7 Eksosgass

I alle kjøretøy med forbrenningsmotor er det en fare for forgiftning eller kvelning som følge av eksosgasser. Som fabrikanten beskriver i brukermanualen til ATVen skal man aldri benytte den i dårlig ventilerte rom, som f.eks. garasjer, carporter eller låver. Den største faren kommer fra Karbon Monoksid som er en fargeløs og luktløs gass.

2.4.8 Brann

ATVen inneholder 95 Oktan bensin. Dette er et medium som er svært brennbart, som gjør at man skal ta forhåndsregler når man skal fylle drivstoff.

3 Metode

Store deler av tidligere arbeid på Lone Wolf prosjektet har hatt et stramt tidsskjema, dette har resultert i mye bruk av «Prøve og feile» metoden. Med BCs oppgaven kan vi bruke mer tid på en mindre oppgave, og dermed bruke riktige metoder som litteraturstudier, skikkelig testing og planlegging.

3.1 Litteraturstudie

Kombinasjonen av «off the shelf» deler som Arduino, 12 V batteri, en sivil ferdig utviklet plattform, industrielle aktuatorer og MK gjør at det ikke eksisterer mange faglige artikler eller andre bachelor oppgaver med samme utgangspunkt. Det ble derfor tatt utgangspunkt i de tekniske dokumentene [21] som hører med Festo MKen.

Flere steder tar dokumentene utgangspunkt i kommunikasjon med en PLS. Det er nevnt ferdig utviklede programblokker som tar input som hastighet og retning, og deretter sender ut de riktige signalene. Siden vi benytter en Arduino var ikke dette en mulighet for oss.

Et arbeidskrav i bachelor faget var å bruke et akademisk oppslagsverk som universitetet disponerer, gjøre søk med avgrensninger og finne relevante artikler for vår oppgave.

Flere søk ble gjort og vi endte opp med to artikler [27] [28].

Den ene omhandler en robot som skal manøvrere et lager lokale og flytte på esker. Med en praktisk vinkling gav den mye innsikt i det elektriske og signalbehandlingen i en robot.

Den andre omhandler MIT sin autonome bil. Her var det et utelukkende matematisk fokus for å styre bilen utenfor dens normale operasjons parameterne. Det de kom frem til var ikke nødvendigvis veldig brukbart for denne bachelor oppgaven, men mye av det de kom frem til ligger i fremtiden til Lone Wolf prosjektet og vil kunne brukes for bedre styrings algoritmer samt utnyttelse av sensor suiten om bord.

3.1.1 Beskrivelse av relevante dokumenter

Dette kapittelet gir informasjon om innholdet i rapportens relevante dokumenter.

BRP XT 650 Operator Guide [26]

Brukermanual for ATVen, skrevet av BRP som er produsenten. Det beskriver sikkerhetstiltak en operatør bør foreta seg før betjening, spesifikasjoner, vedlikehold og mer. Dokumentet er lastet ned fra BRP sine gjemmesider [26].

Festo CANOpen (CMMS-CO_2010-12a_554352g1) [18]

Dokumentet beskriver kommunikasjon via CANOpen, med oppsett og funksjoner som kan benyttes hvis man velger denne protokollen. Dokumentet er utgitt av Festo i 2010 og er lastet ned fra Festo sin hjemmeside.

Festo Mounting and Installation (CMMS-ST-G2-HW_2014-04_8034456g1) [15]

Dokumentet er et informasjonsdokument som viser tabeller for diverse emner som I/Oer og feilmeldinger.

Den inneholder alle grensesnittene på MKen, hva de brukes til, samt detaljerte koblingsskjemaer.

Festo Assembly and Installation (CMMS-ST-G2_2010-08_573125g1) [19]

Gjennomgang av koblingsskjemaer for I/Oer, samt eksempel-metoder for tester og parametrisering. Den gir en oversikt over metoder, protokoller og styringsmåte for CMMS-ST MK.

Festo FHPP Description (CMM_-FHPP_2010-06a_555696g1) [29]

Dokumentet opplyser om hvordan FHPP involveres rundt MKen. Dette er generell protokoll som brukes på tvers av MKer fra Festo. Dokumentet har som hensikt å gi et innblikk i hvordan data behandles i MKen, hvilke feilkoder som finnes og hvordan registrene er bygget opp.

Festo Function and Commissioning (CMMS_D-FW_2014-04_8034520g1) [3]

Omhandler installasjon, parametrisering, testing og feilsøking av MKen med diverse utstyr, moduler og aktuatorer.

Festo Motor Controller Overview (CMMS-ST_EN) [11]

Dokumentet er hentet fra Festo sine hjemmesider, det omhandler ST motorkontroller serien til Festo, altså alle MKer for stepper-motorer. Det finnes en oversikt over moduler, oppsett, kabler og utstyr til MK og en grov oversikt over styremåtene.

Festo Stepper-Motor (EMMS-ST 13_1-18 – Stepper motors EMMS-ST_ENUS) [12]

Dokumentet er hentet fra Festo sine hjemmesider, det omhandler stepper-motoren og enkoderen. Det finnes informasjon om nøyaktigheten til stepper-motoren og enkoder, nominell effekt, moment samt mål og vekt.

Festo CiA402 (CMMS_D-C-CO_2014-04_8034536g1) [30]

Dokumentet er hentet fra Festo sin hjemmeside. Det inneholder alle informasjon om ordre som MKen kan motta eller svare med. Samt relevante pinner for ulike serielle innganger slik som RS-232 og RS-485.

MAX232 Datasheet [31]

Dokumentet er ett datablad som beskriver MAX232 ICen som brukes for RS-232 protokollen.

MAX485 Datasheet [32]

Dokumentet er ett datablad som beskriver MAX485 ICen som brukes for RS-485 protokollen

3.2 Programvare beskrivelse

For å kunne utføre alle innstillinger, programmering og testing kreves ulike programvare som støtter komponentene som blir brukt.

3.2.1 Festo Configuration Tool

FCT er et verktøy fra Festo som tillater parametrisering av alle Festo sine kontrollere. Det støtter kun Windows og det må lastes ned riktig versjon for kontrolleren det gjelder. Kommunikasjon mellom FCT og kontrollere kan gjøres over flere forskjellige protokoller som Ethernet, USB og RS-232, men vil variere ut ifra kontrolleren.

3.2.2 Terra Term

Terra Term er et gratis program for å lese og skrive seriell data. Programmet støttes kun av Windows og versjon 4.105 ble brukt. Programmet ble brukt for å skrive ordre til Festo MKen under kapittel 3.7 om tester.

Dette programmet er populært program for avlesing og skriving av protokoller og data. En av gruppas medlemmer har også erfaringer av å bruke programmet fra tidligere.

3.2.3 Arduino Integrated Development Environment (IDE)

Arduino IDE er en programvare som blir brukt for å programmere ulike Arduino komponenter. Programmet lastes ned gratis fra Arduino sine hjemmesider og støtter både MAC, Linux og Windows. Nyeste versjon 1.8.12 på Windows 10 ble brukt til å skrive programmet. I Arduino IDE skriver man i C++ kodespråk og har mulighet til å benytte seg av innebygde funksjoner uten å kalle opp biblioteker.

Programmet er svært mye brukt av dem som programmerer Arduino. Siden mars 2015 har programmet blitt lastet ned over 40 millioner ganger [33]. Det finnes et stort nettverk rundt Arduino og flere forskjellige forum har mange innlegg skrevet av andre brukere åpent på internett.

3.3 Parametrisering med «Festo Configuration Tool»

MKen krever parametrisering for å stille inn utstyret den er tilkoblet og bruksområdene dens. For å gjennomføre parametriseringen brukes «Festo Configuration Tool» (FCT). Det er viktig at det brukes riktig versjon og «plugin» som støtter valgt MK. I denne oppgaven sitt tilfelle kan riktig versjon finnes her: [34].

Dette kapittelet vil ta seg for alle innstillinger i FCT som har blitt justert for å gjennomføre målet til bacheloroppgaven.

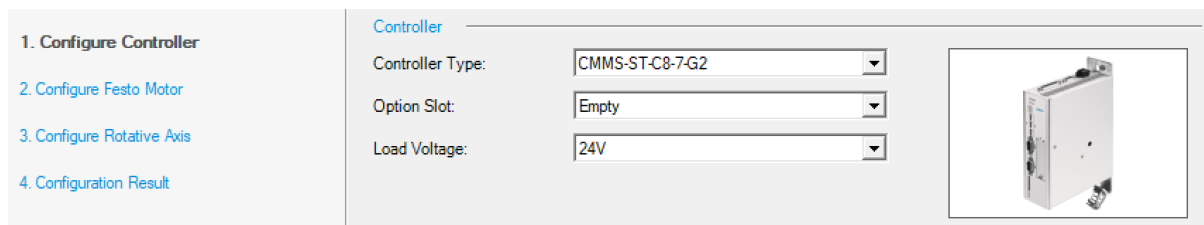
3.3.1 «Configuration»

Som vist i Figur 3.1, finnes det tre menyer for å velge MK.

«Controller Type» forteller hvilken MK man har valgt.

«Option Slot» sier noe om ekspansjonskortet man har i MKen, eksempler er ProfibusDP.

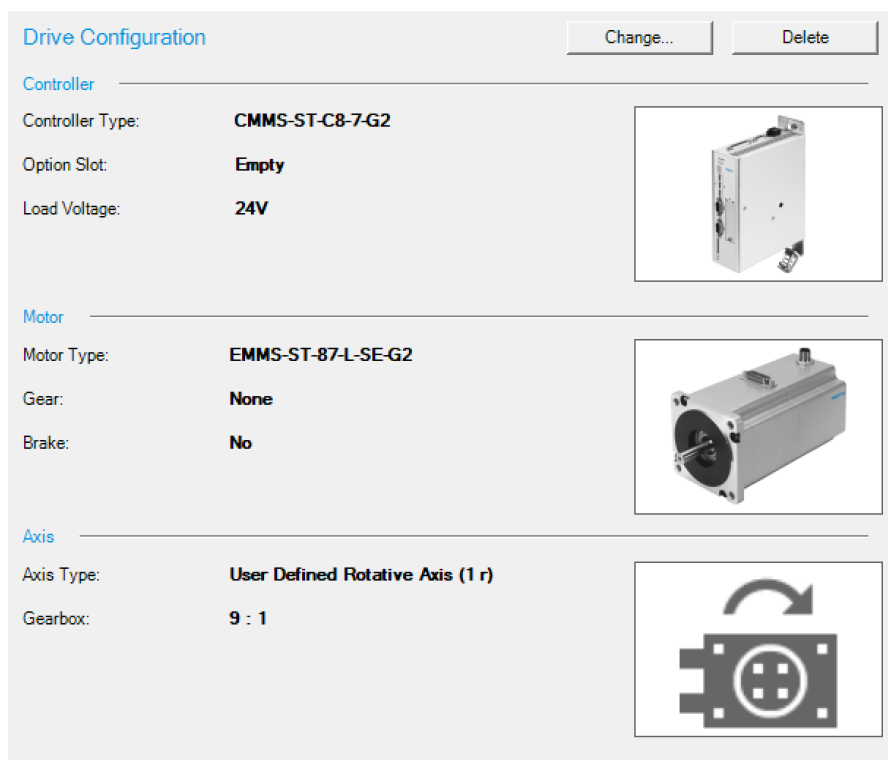
«Load Voltage» er hvilket spenningsnivå MKen får tilført, alt ifra 24 V til 72 V.



Figur 3.1 Valg av MK. (Skjerm bilde fra FCT)

Stepper-motoren som kobles til MKen kan komme i flere størrelser og varianter, med eller uten enkoder, og med eller uten gir. Resultatet av valgene i Figur 3.1 vises igjen i Figur 3.2. Her kan man også se valget av motor og valget av egen gir-modul.

Det er viktig at gir ratioen stemmer overens med den faktiske mekaniske utvekslingen for at posisjonering skal fungere riktig.



Figur 3.2 Valg av motor og gir. (Skjerm bilde fra FCT)

3.3.2 «Application data»

For å kommunisere over RS-485 må «Control Interface» være satt til RS-485. Deretter kan konfigurasjonen lastes ned til MKen og lagres der. Det er også mulig å simulere CANOpen over RS-232 som beskrevet på side 38-40 i «Assembly and installation» dokumentet [19].

Andre alternativer under «Control Interface» i Figur 3.3 er blant annet CANOpen, Analogue interface og Digital interface.

The screenshot shows the 'Operating Mode Settings' dialog box with the following configuration:

- Selected Axis:** User Defined Rotative Axis (1 r)
- Control Interface:** RS-485
- Used Operating Modes:**
 - Profile Position Mode
 - Homing Mode
 - Interpolated Position Mode
 - Profile Velocity Mode
 - Profile Torque Mode
- Used Functions:**
 - X10 Function
 - Encoder Emulation (Master)
 - Synchronisation (Slave)
 - Flying Measure
 - Falling Edge
 - Rising Edge

Figur 3.3 Valg av kommunikasjon. (Skjerm bilde fra FCT)

Om man monterer et medium på den roterende aksen til stepper-motoren så vil dette endre treghetsmomentet og dermed oppførselen til motoren.

For å stille inn MKen riktig kan treghetsmomentet derfor legges inn som en verdi under «Mass moment of inertia» i Figur 3.4:

Operating Mode Settings		Environment	Messages
Selected Axis:	User Defined Rotative Axis (1 r)		
Parameters			
Orientation:	<input checked="" type="radio"/> Horizontal	<input type="radio"/> Vertical	
Inverse Rotation Polarity	<input type="checkbox"/>		
Application Data			
Mass moment of inertia:	<input type="text" value="2,500"/> kgcm ²		
Hint Closed loop settings will be recalculated after change!			

Figur 3.4 Parametre for akse. (Skjerm bilde fra FCT)

Deretter kan «Calculate» knappen som er vist i Figur 3.8 brukes til å regne ut nye reguleringsparametre.

3.3.3 «Axis»

Her kan grensebryterens normal-tilstand parametriseres til enten «Normally Open» (NO) eller «Normally Closed» (NC) som vist i Figur 3.5.

Dette vil avhenge av hvilken grensebryter man har valgt å bruke, og må stemme overens med den for å unngå alarmer.

Selected Axis:	User Defined Rotative Axis (1 r)	
Switch Types		
Limit Switch Type:	<input type="radio"/> NC - Normally Closed <input checked="" type="radio"/> NO - Normally Open	
General Limitations		
Velocity:	19,039	0,111 rpm .. 23,799 rpm
Acceleration	30416,278	20,890 rpm/s .. 30416,278 rpm/s
Stop Decelerations		
Quick Stop:	30416,278	rpm/s
Monitoring time Quick Stop:	500	ms
Stop Input Signal:	30416,278	rpm/s
Limit switch:	30416,278	rpm/s
Enable Editing		

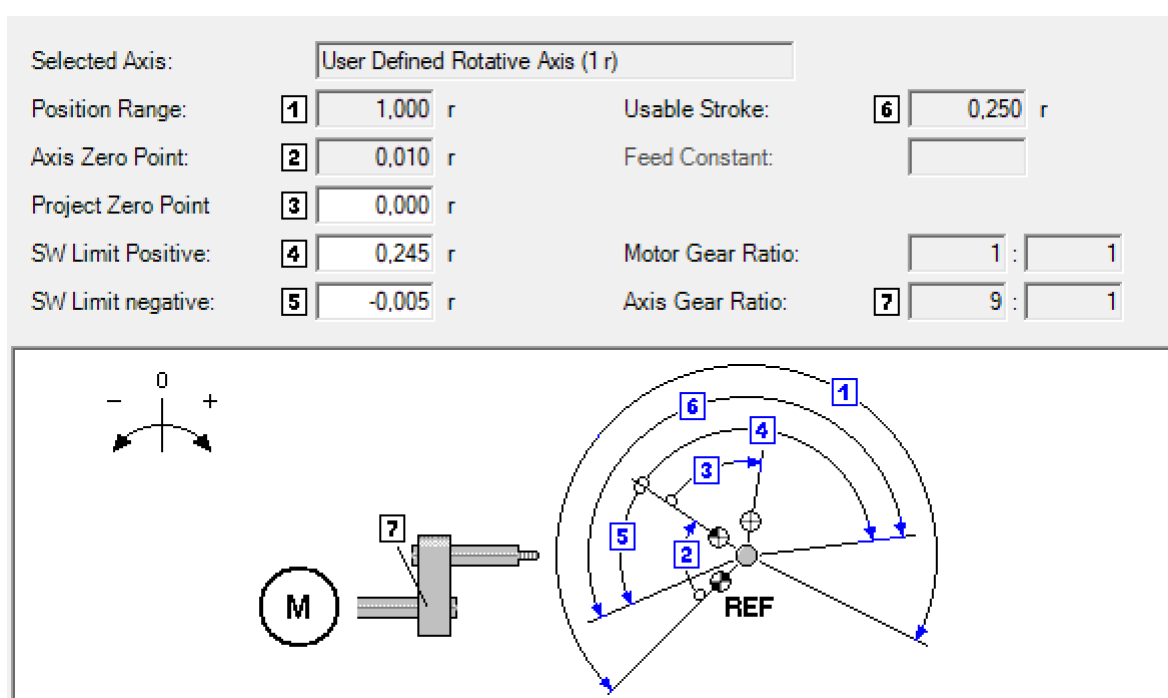
Figur 3.5 Parametrisering av grensebrytere. (Skjerm bilde fra FCT)

Som vist i Figur 3.8 justeres posisjonen for grensebrytere, arbeids område og virtuelle grenser under «axis» menyen i FCT. Figur 3.6 viser FCT sitt skjermbilde under innstilling av parametre.

Tabell 3-1 er hentet fra Festo sitt «Function and Commissioning» dokument, som beskriver formel for å regne ut de forskjellige posisjonene, med utgangspunkt i «REF»:

Tabell 3-1 Formler for posisjoner på motorens roterende akse [3]

Point of reference	Calculation rule		
Axis zero point	AZ	= REF + a	
Project zero point	PZ	= AZ + b	= REF + a + b
Negative software end position	SLN	= AZ + d	= REF + a + d
Positive software end position	SLP	= AZ + e	= REF + a + e
Target position/actual position	TP/AP	= PZ + c	= AZ + b + c = REF + a + b + c



Figur 3.6 Parametrisering av motorens roterende akse. (Skjermbilde fra FCT)

3.3.4 «Homing»

Under «homing» menyen finnes innstilling for hastighet på homing sekvensen, samt valg av retning for å finne grensebryteren som vist i Figur 3.7.

Ved å krysse av for «Go to axis zero point after homing» kan man enten velge at motoren skal stå i ro etter den har funnet grensebryteren, eller bevege seg til det valgte nullpunktet.

Selected Axis: User Defined Rotative Axis (1 r)

Homing Method: Limit switch

Destination: Limit switch

Direction: Negative Positive

Method Description: 17: Limit switch negative

Parameters

	Velocity [rpm]	Acceleration [rpm/s]
Search:	0,500	629,000
Crawl:	0,300	629,000
Running:	0,900	629,000
Torque Threshold:		%
Axis Zero Point:	0,010 r	

Options

Go to the axis zero point after homing

Homing at controller enable

Figur 3.7 Parametrisering av "Homing" metoden. (Skjerm bilde fra FCT)

3.3.5 «Closed loop»

Som vist i Figur 3.8 har «closed loop» menyen parameterne for den interne regulatoren i MKen. Av aktuelle parametere for bachelor oppgaven nevnes «Position Control» ruten. Her kan det stilles inn «Gain», altså regulatorforsterkning (KP), maks korreksjons hastighet og dødbånd for settpunktet.

Om man ønsker kan det kalkuleres nye parametere automatisk ved bruk av «slideren» på bunnen, der «soft» vil resultere i en jevn, men treg regulering, og «hard» vil resultere i en rask, men aggressiv regulering. Kalkulasjonen tar utgangspunkt i hva «slideren» er satt på, samt hva treghetsmomentet er stilt inn til som vist i Figur 3.8.

Closed Loop Settings for unsupported configuration

Control Type	Parameter	Value	Unit
Current Control	Gain	7,50	
	Time Constant	5,09	ms
Velocity Control	Gain	0,59	
	Time Constant	10,21	ms
	Actual Velocity Filter	2,00	ms
Position Control	Gain	0,28	
	Max. Correction Velocity	19,039	rpm
	Dead Range	0,000	r
Application Data	Mass moment of inertia	2,500	kgcm ²
	Inertia Ratio	1,0	

Soft Middle Hard

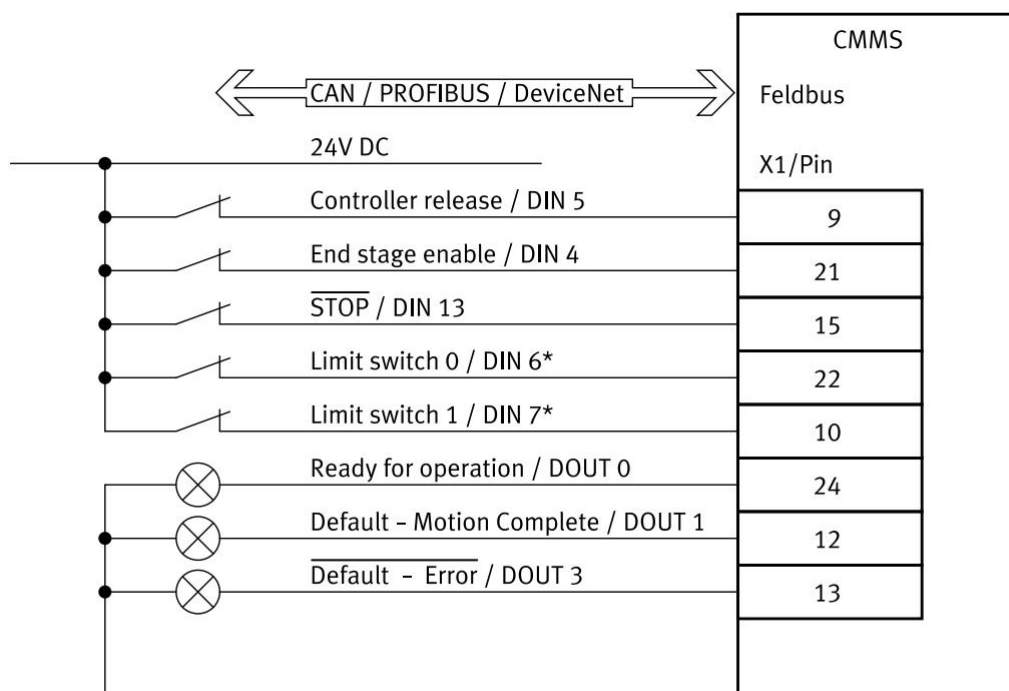
Calculate Enable Editing

Figur 3.8 Parametere for regulatorer i FCT(Skjerm bilde fra FCT)

3.3.6 «Control interface»

Det digitale I/O grensesnittet består i hovedsak av innganger og utganger med en bestemt funksjon. Noen kan programmeres til å indikere fullførte bevegelser, feil koder eller settes konstant høy eller lav. Som vist i Figur 3.10 finnes det tre programmerbare, digitale utganger.

Det er krav om at diverse innganger må settes høy eller lav for at MKen skal bli operasjonell. Dette er beskrevet i Figur 3.9 Krav for digital I/O for feltbuss kommunikasjon:



The connection plan shows the switch position in the active operating state.

*) The limit switches are set by default to opener (configuration over FCT)

Figur 3.9 Krav for digital I/O for feltbuss kommunikasjon [19]

Mode Selection over DIN9 and DIN12

active Active Mode:

Digital Inputs		Digital Outputs	
DIN0: Record Select 0	DIN8: Start Positioning	DOUT0: <input type="text" value="Controller ready for operation"/>	
DIN1: Record Select 1	DIN9: Mode Select Bit 1	DOUT1: <input type="text" value="Homing mode complete"/>	
DIN2: Record Select 2	DIN10: Record Select 4	DOUT2: <input type="text" value="Target position reached"/>	
DIN3: Record Select 3	DIN11: Record Select 5	DOUT3: <input type="text" value="Error"/>	
DIN4: Enable Power	DIN12: Mode Select Bit 0		
DIN5: Enable Control	DIN13: Stop		
DIN6: Limit Switch 0			
DIN7: Limit Switch 1			

Offline View of the mode dependent I/O Configuration

Offline Mode:

Figur 3.10 Digital I/O oversikt. (Skjerm bilde fra FCT)

Det analoge grensesnittet kan brukes til å sende settpunkt for hastighet eller moment, og har en programmerbar og skalerbar tilbakekobling som vist i Figur 3.11. Grensesnittet ble brukt i en tidligere versjon av Lone Wolf for å få tilbakekobling til regulator.

Analogue Input

A Input Voltage of + 10 Volt correspond...

Scaling (Velocity): rpm

Scaling (Torque): %

Offset: V

Safe Zero: V

Analogue Output

Analogue Monitor:

Scaling: r

Offset: V

Numeric Overflow Limitation

Figur 3.11 Analog utgang for tilbakekobling. (Skjerm bilde fra FCT)

3.3.7 «Fieldbus»

Som beskrevet i kapittel 2.2.2 brukes IEC standarden CiA 402 for kommunikasjon over RS-232 og RS-485. «Data Profile» må derfor settes til «CiA 402» som vist i Figur 3.12:

The screenshot displays two main configuration panels. The top panel, titled 'Interface Parameters', contains the following fields: 'Interface Type' (text box with 'CANopen'), 'CAN Address' (text box), 'Bit Rate' (text box), and 'Data Profile' (dropdown menu with 'CiA 402' selected). The bottom panel, titled 'Factor Group', includes a 'Used' checkbox, a 'Unit' dropdown, and several input fields: 'Exponent Position', 'Exponent Velocity', and 'Exponent Accel.' (each a dropdown menu); 'Gear' (two text boxes separated by a colon); 'Feed Constant' (text box); and 'Factor Position', 'Factor Velocity', and 'Factor Accel.' (each consisting of two text boxes separated by a colon, with '1' entered in both boxes of each pair).

Figur 3.12 Parametrisering av feltbuss. (Skjerm bilde fra FCT)

Om man ønsker at MKen skal regne om verdiene den får tilsendt over feltbussen så kan «Factor Group» brukes. Omregning kan også gjøres før dataen blir sendt til MKen, av for eksempel Arduinoen.

3.3.8 «Error Management»

Det finnes en foriglingsmatrise som er parametriserbar. Tabell 3-2 viser kan man for eksempel dempe feilmeldinger som skulle oppstå eller angi respons for feilmeldingen. Ved simulering av CANOpen så kan for eksempel feilmelding nr. 122 dempes.

Tabell 3-2 Foriglingsmatrise for feilmeldinger i FCT(Skjerm bilde fra FCT)

No.	Error Text	PS off	QStop	Warn	Ignore
20	DC Bus undervoltage	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
31	Overheating error (Motor)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
40	Overtemperature power stage	<input checked="" type="radio"/>	<input type="radio"/>		
122	CAN communication	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
170	Following error	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
180	Motor temperature 5°C below maximum	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
181	Output stage temperature 5°C below maximum	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
190	I _{Pt} at 80%	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
220	PROFIBUS init error	<input checked="" type="radio"/>	<input type="radio"/>		
290	No SD available	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
291	SD initialisation	<input checked="" type="radio"/>	<input type="radio"/>		
292	SD parameter set	<input checked="" type="radio"/>	<input type="radio"/>		
310	I _{Pt} -error motor (I _{Pt} at 100%)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
311	I _{Pt} -error power stage (I _{Pt} at 100%)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
400	Softwarelimit negative	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	
401	Softwarelimit positive	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	
402	Target position behind softwarelimit negative	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	
403	Target position behind softwarelimit positive	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	
418	Record sequence: Unknown command	<input checked="" type="radio"/>	<input type="radio"/>		
419	Record sequence: Invalid branch destination	<input checked="" type="radio"/>	<input type="radio"/>		
421	Position precomputation	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
424	Please enforce homing run!	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
430	Limit switch negative	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	
431	Limit switch positive	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	
439	Both limit switches on	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	
650	DeviceNet assembly	<input checked="" type="radio"/>	<input type="radio"/>		
651	DeviceNet initialisation	<input checked="" type="radio"/>	<input type="radio"/>		
703	Operating mode	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
790	RS232 communication error	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

3.4 Programmering

Arduino IDE baserer koden sin på en enkelt kjørende oppstarts-metode og en gjentakende løkke. For at programmet skal kjøres problemfritt hentes biblioteker og globale variabler settes i starten av programmet før oppstarts-metoden og løkken kjører.

For å hindre unødvendig bruk av ressurser og prosessor tid bør man begrense antall operasjoner ved hver gjennomkjøring av koden [24]. En god måte å gjøre dette på er bruk av metoder som kun blir kalt opp av hendelser. Konstant oppdatering av måleverdier fra aktuatorer og enkodere samt RCer og det autonome systemet bør gjennomføres i programmets hoved løkke «loop()».

For å finne riktige verdier i binær, heksadesimal og desimal kan det benyttes «RapidTables» som oversetter på deres nettside [35]. På Arduino sin hjemmeside finnes oppslag for å bestemme riktig oppsett av heksadesimale tall [23], der sifrene som kommer etter «0x» er heksadesimaler. Ved å benytte seg av «*String(0xVerdi, HEX)*», beskriver «HEX» at koden skal leses som et heksadesimalt tall.

3.4.1 Resurser for koding

For å skrive et velfungerende program er hjelpemidler som hjemmesiden til Arduino og litteratur om omhandler C++ programspråk til stor hjelp [23] [36] [24]. Siden Arduino er åpent for alle, er det mange tråder på Arduino sitt eget forum, samt andre forum for C++ språket som bidrar til alternative problemløsninger og skrivemetoder.

3.5 Kretskort

Ved bygging av kretskort trengs en grunnleggende kunnskap om lodding og design av kretser. Denne kunnskapen kan tilegnes på internett samt med prøve og feile metoden.

3.5.1 RS-485

Ettersom en Arduino Mega 2560 ikke har noen utgang som støtter de elektriske egenskapene til RS-232/485, er man nødt til å bruke en IC for å etablere kommunikasjon via TTL seriell på Arduinoen.

For å koble opp en MAX485 følges databladet for ICen som finner her [32].

På den ene enden trengs en fullstendig koblet nullmodem kabel igjennom en D-sub 9 kontakt. På den andre enden trengs det tilkobling fra Arduinoen sine TTL serielle pinner og de digitale utgangene for å oppnå signalgang.

Den enkleste metoden er å anskaffe ett tomt test-kretskort der det lages baner på kortet ved hjelp av tinn og broer. Da loddes en MAX485 IC på kortet som deretter tilføres de nødvendige inngangene og utgangene samt strøm og jord.

3.5.2 RS-232

En alternativ styringsmetode benytter RS-232, da kan et kretskort designes med bruk av databladet til MAX232 ICen som kan finnes her [31].

Dette utnytter en MAX232CPA+ IC som opererer i «full-duplex» mode, og er derfor en DIP-16 IC i motsetning til DIP-8 ICen til MAX485. «DIP» står for «Dual in-line package» og er en standard for den fysiske formen til en IC. Tallet etter DIP beskriver antall bein på ICen. Interfacen på hver side av ICen vil bli det samme, minus de digitale utgangene for signalgang. Derimot trengs det flere kondensatorer for å oppnå riktige spenningsverdier.

3.5.3 Digital I/O

Som beskrevet i kapittel 3.3.6 er det noen bestemte digitale innganger som må settes høy for å gjøre MKen operasjonell. Disse inngangene opererer på 24 V spenning som utelukker muligheten til å styre de direkte med Arduinoen sine digitale I/Oer.

Flere reléer, transistorer eller optokoblere kan brukes for å oppnå aktivering av MKen sine digitale innganger.

3.5.4 Fullstendig kretskort

For å slippe å ha flere forskjellige kretskort med tilhørende kabler mellom seg, anses det som en fordel å designe en fullstendig løsning som inneholder alle kretskortene beskrevet i kapittel 3.5.

Det eksisterer løsninger for prototype bygging som kan kjøpes inn, for eksempel tomme, perforerte kretskort med baner laget for ICer. Deretter kan designet for kretskortet testes på for eksempel et tomt Arduino «breadboard» for å være sikker på at det fungerer slik det er tiltenkt.

Når alt er testet kan designet implementeres permanent på ett prototype kretskort der alle komponentene loddes fast. Om det skulle mangle baner for å lage koblinger kan «jumpere» lages med en bit kobber ledning. På samme måte kan man også lage koblingspunkter mellom forskjellige baner ved å lodde en kobber ledning på tvers av dem.

3.6 Testrigg

En testrigg er en plattform for å teste og videreutvikle et prosjekt. Den hjelper til å skille ut deler av et system og gjør det lettere å jobbe, ettersom den vil være mindre omfattende.

Samtidig er det nødvendig å ta med nok deler slik at man får gjennomført de omfattende testene, uten å trenge å integrere delene tilbake i systemet. Derfor må en grense bli satt for hvor man fjerner deler fra de forskjellige systemene, både elektrisk, mekanisk og signal.

Det er meget behjelpelig om testriggen er enkel å flytte på samtidig som den er romslig med plass for nye deler og kabler. Strøm tilførsel til systemet på testriggen bør være så likt systemet på ATVen som mulig. Da er man sikker på at systemet vil ha samme virkemåte alltid. En viktig del av strøm tilførsel er også signal jord. Om systemene er isolert fra hverandre kan man få potensial forskjell. Da vil spennings verdier bli feil, og signaler kan slutte å fungere som de skal. Enhver strømkilde som har en last på seg bør også ha en sikring. Ved kortslutning eller feilkobling forhindrer man skade på utstyr ved at sikringen smelter.

For å simulere plasseringene på ATVen bør grensebryter plasseres tilsvarende på testriggen. Om man lager nye kabler til testriggen så vil ikke disse nødvendigvis være lange nok når utstyret plasseres tilbake på ATV. Dette bør tas med i betraktning ved designing av kablen slik at den er enkel å forlenge eller slik at den blir den korrekte lengden.

3.7 Tester

Før styresystemet ble demontert fra ATVen ble det gjennomført noen tester i form av å teste RCen med ny mottaker for å verifisere at ATVen fungerte som normalt. Dette var foretatt før tiltakene til koronautbruddet.

3.7.1 Fjernstyring av ATV

ATVen hadde to forskjellige virkemåter ved prosjekt start. Én der den var autonom og én der den ble styrt med en RC. For å få ett best mulig utgangspunkt for arbeidet videre var det viktig at ATVen fungerte som den hadde gjort ved sist sommer prosjekt [2]. Da må det testes at oppstarten og styringen har samme virkemåte som beskrevet i den rapporten [2].

Da ATVen ble styrt med RC ble det brukt en kombinasjon av en 4-kanals mottaker og en RC sender. Mottakeren var koblet direkte på de digitale inngangene på Arduinoen.

I Arduino koden ble det så brukt «PulseIn» metoden som regner ut lengden på hver puls, tiden mellom hver stigende flanke, i millisekunder og gir ut verdien som en integer.

Systemet skrues på og løsningen ble testet ved å kjøre ATVen med RCen og se på virkemåten til oppstarts prosedyren og responsen til styringen.

3.7.2 Kommunikasjon mot MK med RS-232

«Serial interface» på MKen brukes til parametrisering ved hjelp av FCT. MKen støtter også parametrisering over RS-232.

Ved hjelp av en ATEN USB til RS-232 adapter og en «female to female» nullmodem-kabel kunne vi koble en PC til MKen. Terra term ble så brukt for kommunikasjon med innstillingene vist i Tabell 3-3, hentet fra side 33-34, tabell 3.5, 3.6 og 3.7 i dokument [19]:

Tabell 3-3 Innstillinger for Terra term

Parameter	Innhold
New Line, Receive	Auto
New Line, Transmit	CR+LF
Terminal ID	VT100
Baud Rate	9600 baud
Data	8-bit
Parity	None
Stop bits	1-bit
Flow Control	None

Når MKen starter opp forventes det å motta en «bootup»-melding som inneholder programvare versjonen til MKen og noe annen informasjon. Det vil også være mulig å sende over generelle kommandoer for å parametrisere kommunikasjonen som vist i dokument [19].

For å dempe feilmeldingen som kommer når MKen er parametrisert til feltbuss kommunikasjon, men ikke har noe koblet til «X4» pluggen må «CAN communication» settes til «ignore» som vist i Tabell 3-2 Forigligningsmatrise for feilmeldinger.

3.7.3 Test av «Homing» over RS-232

For å bevege stepper-motoren kreves det at tre digitale innganger er satt høy, [3] side 95 figur 5.9. Man må derfor ha en relé-modul koblet til en Arduino som setter disse inngangene høye etter at MKen har blitt skrudd på, per figur 5.11, [3] side 101.

Etter at MKen er koblet til PC-en åpner man Terra Term og bruker test 3.7.2 for å sette opp kommunikasjonen riktig. Deretter følger man «Example Homing Mode via RS-232» på side 40 i «Assembly and Installation» dokumentet [19]. Eventuelle feilmeldinger vil kunne leses av på 7-segment displayet til MKen. For å dekode disse brukes kapittel 8.2 [19].

For å parametrisere verdiene som brukes i «homing» prosedyren brukes FCT som beskrevet i kapittel 3.3.3.

3.7.4 Test av «Profile positioning» over RS-232

Stegene i test 3.7.3 må gjennomføres før man kan gjennomføre denne testen.

Ved å følge «Example Profile Position Mode via RS-232» på side 38-40 [19] kan man teste posisjonering med MKen over RS-232. [37]

Det gjøres noen endringer av prosedyren beskrevet for å tilpasse den til vår situasjon:

- Punkt 1 trengs ikke å gjennomføres ettersom dette er gjort av «homing»
- Punkt 4 må endres til «=604000:009F»

Under Punkt 4 må endres for å sette bit 7 «høy», den sletter feil koder som ligger i systemet [38] [40]. Dette står forklart under «Device Profile Segment» - «6040h - Control word». I manualene fra Festo står det ingenting om denne løsningen. Under forrige test dukket det opp «warning» om «software limit negativ» ved «homing» der grensebryteren aktiveres. Vi vurderte at dokumentasjonen ikke tar hensyn til disse alarmene ved gjennomført «homing». Vi undersøkte Festo sine manualer uten funn etter hvilken bit i «data stringen» som styrer kvittering av alarmer. Det ble funnet en side under «Manufacturer segment» - «2011h – Warning bits» [37] som beskrev kontroll ordet som sendes til MKen. Den omtaler «bit 7: fault reset» som vil kvittere eventuelle feil før posisjonering utføres om den er satt høy. Posisjonen som skal brukes i testen må være innenfor grensene satt i parametriseringen av MKen.

3.7.5 Kretskort for TTL til RS-485

Den letteste måten å teste kretskortet på uten å koble det direkte til MKen blir å gå til anskaffelse av en RS-485 til USB adapter.

Deretter kobler man kretskortet til adapteren og Arduinoen og skriver ett test program.

Programmet kan simulere styring av MKen, som deretter kan leses av i klartekst på PC-en for å verifisere at alt fungerer slik det skal.

3.7.6 Kretskort for TTL til RS-232

For å teste kretskortet sender man signalet fra Arduinoen til PC-en igjennom en USB til RS-232 adapter. Det er viktig å bruke samme baud rate som er valgt i Arduino koden.

Dermed kan man lese av det som sendes fra Arduinoen i klar tekst i Terra Term. Om det trengs å gjøres noen endringer til programmet vil disse kunne leses av umiddelbart.

4 Resultater

Resultat kapittelet gjenspeiler metodene som er brukt i rapporten opp mot oppgavebeskrivelsen. Dette kapittelet gjennomgår hvordan hele testtriggen er bygd opp sammen med programbeskrivelse og resultatet av testene.

4.1 Programbeskrivelse

Programmet for dette prosjektet er satt opp slik som det pleier å gjøres i C type programspråk. Dette vil si at biblioteker og globale verdier er på toppen, før selve programmet med «setup()» og «loop()». Metodene er skrevet i bunn av programmet. Med dette oppsettet har programmet tilgang til all informasjon som skal til for å kjøre. Det er valgt å ikke bruke konstanter i koden da koden er ment for å kunne være dynamisk ved senere implementering av det autonome systemet. Figurer som inneholder kode, er hentet fra programkoden i Vedlegg C.

4.1.1 Variabler

Det er brukt globale og lokale verdier i programkoden. Det er valgt å ikke benytte seg av konstanter i koden da det ikke er behov for å låse verdiene. Variablene blir brukt som parameterverdier og for å sette innganger og utganger til RCen og reléer. I Tabell 4-1 er alle variablene listet opp med beskrivelse:

Tabell 4-1 Variabler some er brukt i Arduino programkode

Type	Format	Benevning	Beskrivelse
Global Variabel	Int	RCPin2	DI fra RC mottakeren satt til D8
Global Variabel	Int	controlEnable	DO for kontroll av «Control Enable» relé fra D7
Global Variabel	Int	outputStageEnable	DO for kontroll av «Output Stage Enable» relé fra D6

Global Variabel	Int	mkStop	DO for kontroll av «MK Stop» relé fra D5
Global Variabel	Int	RCPWM	Variabel for inngangssignalet fra RC
Global Variabel	Long	setPoint	Settpunkt for styresignalet
Global Variabel	Long	lastAngle	Lagrer siste styreverdi for bruk i filter
Global Variabel	String	command	Første del av ordren.
Global Variabel	String	data	Andre del av ordren
Global Variabel	String	leadZeroes	Antall nuller foran «data»
Global Variabel	String	order	Hele ordren samlet
Lokal Variabel	Int	differenceAngle	Forskjellen fra nåværende styresignal fra forrige
Lokal Variabel	Int	deviation	Filter parameter. Forskjellen som kreves for at nytt settpunkt skal endre seg
Lokal Variabel	Int	i	Lokal variabel i «for-løkken»
Lokal Variabel	Int	j	Lokal variabel for formatering av «leading zeros»
Lokal Variabel	Long	mult	Utregningsfaktor for maks styreutslag for MKen i desimaler dividert på styrevinkelen 90 °
Lokal Variabel	Long	hexSteering	«mult» multiplisert med «angle»

De mest sentrale variablene som er verdt å merke seg er «command» og «data» som danner «order». «order» blir formatert til lesbar verdi for MKen ved å legge inn «=» og «leadZeroes». Hvordan variabelen er bygget opp vises i Figur 4.1:

```
order = String("=") + command + String(":") + leadZeroes + data;
```

Figur 4.1 «order» variabel

For å danne «command» og «data» bruker programmet enten ferdige prosedyrer som står i Festo sine manualer, eller styresignal fra RC systemet. Styresignalet fra RC systemet blir satt av funksjonen «pulseIn» som setter PWM signalet fra RCen til verdien «RCPWM». Denne variabelen strekker seg fra litt under 1 192 til litt over 1 761. Når verdien overføres til «setPoint» brukes en «constrain» med «map» funksjon inni seg for å omgjøre «RCPWM» til et 0 ° til 90 ° signal. Dette sammen med «Serial.print» og metoden «SetPosition» er alt som gjøres i «loop()». Innhentning av signalet vises i Figur 4.2:

```
RCPWM = pulseIn(RCPin2, HIGH);  
setPoint = constrain(map(RCPWM, 1761, 1192, 0, 90), 0, 90)
```

Figur 4.2 «RCPWM» Styresignal

4.1.2 Metoder

Det er fem metoder som er komponert for å lage et ryddig og brukervennlig program. Disse blir kalt opp fra ulike steder i koden. Tabell 4-2 viser metodene som utfører de sentrale funksjonene i programmet.

Tabell 4-2 Metoder som er brukt i Arduino programmet

Metodenavn	Kalles opp fra	Beskrivelse
Void Transmit	Alle metoder	Samler «order» og sender den ut på «Serial1»
Void Startup	Setup()	Gjennomfører oppstarts prosedyre
Void Homing	Setup()	Gjennomfører homing
Void SelectPositioning	Setup()	Setter MKen i Absolut posisjoneringsmodus
Void SetPosition	Loop()	Gjennomfører posisjonering basert på inngangsvariabelen «angle»

«Transmit» gjennomfører formatering av «order» og sender ut på TTL seriell utgangen «Serial1». Denne metoden kalles opp hver gang Arduinoen skal fortelle MKen om å gjøre noe og er vist i Figur 4.3:

```
void Transmit() {
    order = String(«=») + command + String(":") + leadZeroes + data;
        Serial1.println(order);
    Serial.println(order);
}
```

Figur 4.3 «Transmit» metode

Under «setup()» som kjøres en gang når programmet starter opp, kalles «Startup», «Homing» og «SelectPosition» opp. Disse tar seg av de nødvendige prosedyrene som må gjennomføres for å sette opp MKen. I Figur 4.4 vises hele sekvensen som «setup» kjører.

```
void setup() {  
  Serial.begin(9600);  
  Serial1.begin(9600);  
  pinMode(RCPin2, INPUT);  
  pinMode(controlEnable, OUTPUT);  
  pinMode(outputStageEnable, OUTPUT);  
  pinMode(mkStop, OUTPUT);  
  delay(65);  
  digitalWrite(controlEnable, HIGH);  
  digitalWrite(outputStageEnable, HIGH);  
  digitalWrite(mkStop, HIGH);  
  Startup();  
  Homing();  
  SelectPositioning();  
}
```

Figur 4.4 «setup()» setter opp grunninnstillinger

«Startup» aktiverer mottak for RS-232 i MKen ved å sende «Enable Logic» ordren. Deretter sender den «Shutdown» ordren som stopper opp bevegelsen som motoren eventuelt skulle ha uten å aktivere bremsen. Til slutt gjennomfører den «Switch on/Disable operation» som klar- gjør for bevegelse før den setter MKen i «Ready state». Metoden er vist i Figur 4.5:

```
void Startup() {  
    command = String(0x651010, HEX);  
    data = String(0x0002, HEX);  
    leadZeroes = "000";  
    Transmit();  
    command = String(0x604000, HEX);  
    data = String(0x0006, HEX);  
    leadZeroes = "000";  
    Transmit();  
    data = String(0x0007, HEX);  
    leadZeroes = "000";  
    Transmit();  
    data = String(0x000F, HEX);  
    leadZeroes = "000";  
    Transmit();  
}
```

Figur 4.5 «Startup» metode

«Homing» metoden gjennomfører «homing» med å sette MKen i «Homing mode», for deretter å starte Festo sin «homing» sekvens ved å sende «Start homing». Det er lagt inn en forsinkelse på 20 sekunder, 5 sekunder mer enn den lengste tiden den bruker for å nå grensebryteren. Til slutt settes MKen i «Ready state». Metoden er vist i Figur 4.6:

```
void Homing() {
    command = String(0x606000, HEX);
    data = String(0x06, HEX);
    leadZeroes = "0";
    Transmit();
    command = String(0x604000, HEX);
    data = String(0x001F, HEX);
    leadZeroes = "00";
    Transmit();
    delay(20000);
    data = String(0x000F, HEX);
    leadZeroes = "000";
    Transmit();
}
```

Figur 4.6 «Homing» metode

«SelectPositioning» gir MKen beskjed om at alle tilførte settpunkter skal leses som absolutte og ikke relative verdier. Dette referer Festo til som «Profile positioning mode» [3]. Metoden er vist i Figur 4.7:

```
void SelectPositioning() {
    command = String(0x606000, HEX);
    data = String(0x01, HEX);
    leadZeroes = "0";
    Transmit();
}
```

Figur 4.7 «SelectPositioning» metode

«SetPosition» er metoden som bestemmer hvilken posisjon motoren skal gå til. Den har et heltall inngangsparameter som er ønsket styreretning. Til å starte med gir den MKen beskjed om å gå i «Ready state». MKen krever å stå i «Ready state» for å gjenkjenne en ordre om posisjonsendring. Den gjennomfører deretter filtrering som står beskrevet over Figur 4.9. Den filtrerte verdien «angle» som er styrevinkel i grader multipliseres med 1 638. Dette er nærmeste desimale heltall verdien av 0,25 rotasjoner dividert på 90 grader. Den multipliserte verdien er da «hexSteering» som legges inn i «data» og gir MKen ny settpunktverdi. Til slutt sender den «Go to steering setpoint» som gjør at MKen starter posisjoneringen mot det gitte settpunktet. Metoden er vist i Figur 4.8:

```
void SetPosition(long angle){
    command = String(0x604000, HEX);
    data = String(0x000F, HEX);
    leadZeroes = "000";
    Transmit();
    int differenceAngle = abs(lastAngle - angle);
    int deviation = 4;
    if(differenceAngle < deviation ){angle = lastAngle;}
    long mult = 1638; //147456/90=1638
    long hexSteering = mult*angle;
    lastAngle = angle;
    command = String(0x607A00, HEX);
    data = String(hexSteering, HEX);
    leadZeroes = "";
    if(data.length() < 8){
        int j = 8- data.length();
        for(int i = 0; i < j; i++){data = «0» + data;}}
    Transmit();
    command = String(0x604000, HEX);
    data = String(0x009F, HEX);
    leadZeroes = "00";
    Transmit();
}
```

Figur 4.8 «SetPosition» metode

4.1.3 Spørringer, løkker og formatering

Programkoden innehar spørringer og løkker som tar seg av filtrering og formateringsproblematikk rundt «leading zeros». Alle spørringene og løkkene befinner seg i «setPosition» metoden som vist i Figur 4.8.

Under filtreringen lages «differenceAngle» som er absoluttverdien av differansen fra forrige styrevinkel og ny styrevinkel. Det benyttes deretter en «if» setning som spør om «differenceAngle» er større enn innstilt «deviation» verdi. Dersom kriteriene oppfylles settes «angle» lik forrige styringsvinkel og verdien blir den samme som forrige gjennomføring. Dersom forskjellen er større enn innstilt verdi, beholder «angle» sin opprinnelige verdi og blir brukt i multiplisering. På slutten av filteret skrives «lastAngle» til «angle» for å brukes neste gang metoden kjøres. Spørringen vises i Figur 4.9:

```
if(differenceAngle < deviation ){angle = lastAngle;}
long mult = 1638; //147456/90=1638
long hexSteering = mult*angle;
lastAngle = angle;
```

Figur 4.9 Filtrering av styresignal

Formatering av «data» variabelen for å få riktig antall «leading zeros» blir gjennomført med en «if»-spørring og fulgt opp med en «for-løkke». Siden dette er det eneste stedet der ikke «data» inneholder et fast antall «leading zeros» er det satt inn i «setPosition» metoden i stedet for i «Transmit».

«data.length()» metoden blir brukt for å lese hvor mange karakterer «data» variabelen inneholder. Siden MKen krever at «data» variabelen består av to heksadesimale tall, der hver av de har reservert fire karakterer vil spørringen sjekke om kriteriet er oppfylt.

For løkken oppretter «int» variabelen «i» og setter kriteriet at løkken skal kjøres så lenge i er mindre enn den lokale variabelen «j». «j» regnes ut til manglende enheter for at lengden på data skal bli åtte.

Løkken legger dermed til en «0» karakter på første posisjon i «data» variabelen «j» antall ganger. Så dersom data før spørringen inneholder fire karakterer vil «j» regnes ut til fire og løkken kjøres fire ganger. Dette legger til fire «leading zeros» som oppfyller kravet for «data»-variabelen. Spørringen er vist i Figur 4.10:

«data» vil i denne oppgaven alltid bestå av fem enheter eller mindre. Dette grunnet at «24000» heksadesimal er 90° sving, og «0» heksadesimal er 0° sving.

```
data = String(hexSteering, HEX);
leadZeroes = "";
if(data.length() < 8){
    int j = 8- data.length();
    for(int i = 0; i < j; i++){data = «0» + data;}}
```

Figur 4.10 «Leading zeros» prosedyre

4.1.4 Gjennomkjøring av koden

Ved oppstart starter programmet med å lese inn alle de globale variablene i toppen av koden. Siden det ikke ligger noen biblioteker i denne koden, blir ikke dette kalt opp. Flytskjemaet i Figur 4.13 kan også følges for gjennomkjøring av koden.

Etter dette kjøres «setup()» som starter opp «Serial» og «Serial1» med en «baud-rate» på 9600 baud, før «pinMode» for RCen blir kalt opp som inngang og reléene blir kalt opp som utganger. Deretter aktiveres reléene og metodene «Startup», «Homing» og «SelectPositioning» blir kjørt fortløpende som vist i Figur 4.11:

```
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  pinMode(RCPin2, INPUT);
  pinMode(controlEnable, OUTPUT);
  pinMode(outputStageEnable, OUTPUT);
  pinMode(mkStop, OUTPUT);
  delay(65);
  digitalWrite(controlEnable, HIGH);
  digitalWrite(outputStageEnable, HIGH);
  digitalWrite(mkStop, HIGH);
  Startup();
  Homing();
  SelectPositioning();
}
```

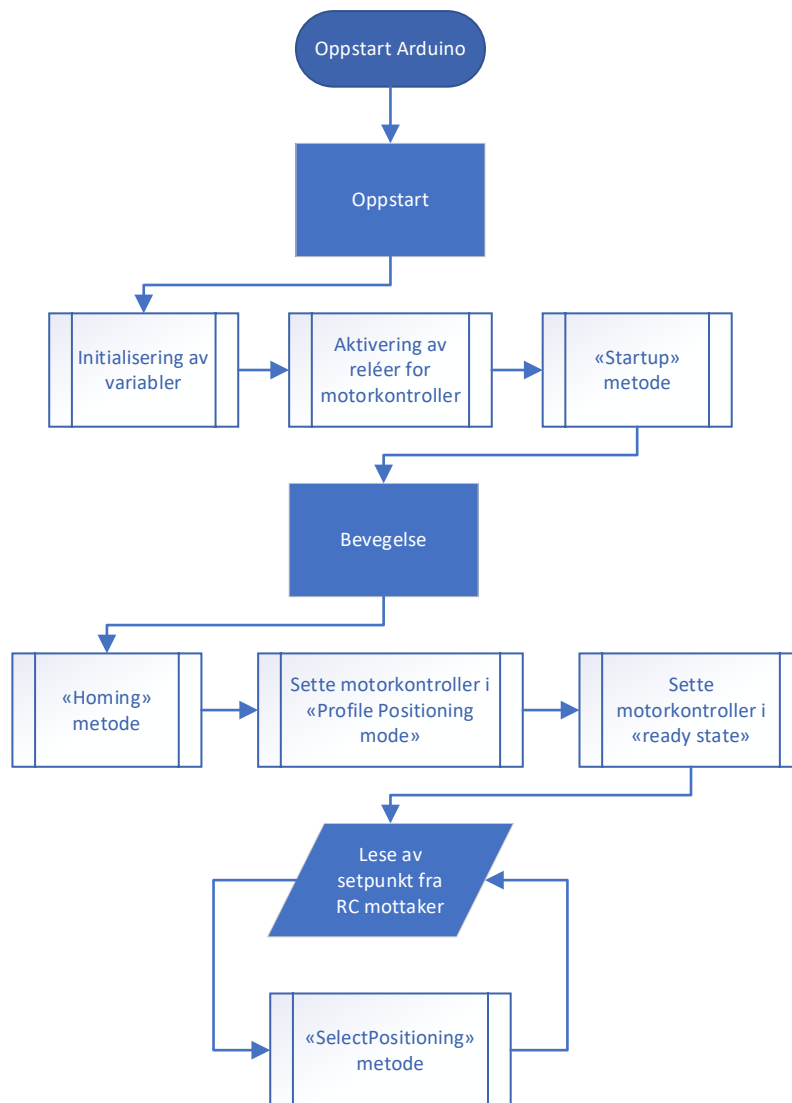
Figur 4.11 «setup()» kodedel

Etter gjennomført «setup()» kjøres «loop()» helt til Arduinoen er skrudd av. Når «loop()» starter vil alt av oppstarts prosedyrer og kalibrering være utført, det er derfor bare lesing av inngangen fra RC og aktivering av «SetPosition» metoden som blir utført gjentagende ganger. Dette er vist i Figur 4.12:

```
void loop() {  
  RCPWM = pulseIn(RCPin2, HIGH);  
  setPoint = constrain(map(RCPWM, 1761, 1192, 0, 90), 0, 90);  
  Serial.print(setPoint + String(" --- "));  
  SetPosition(setPoint);  
}
```

Figur 4.12 «loop()» programmets hovedløkke

«loop()» setter frekvensen fra «RCPin2» til variabelen RCPWM. Når dette er lest inn blir «map» funksjonen brukt sammen med «constrain» for å regne om «RCPWM» til et $0^\circ - 90^\circ$ styresignal kalt «setPoint». «setPoint» blir til slutt skrevet som inngangsparameter til metoden «SetPosition» før «loop()» kjøres igjen. «Serial.print()» metoden er brukt her i henhold til feilsøking-metoden beskrevet i kapittel 0.



Figur 4.13 Flytskjema for Arduino kode

4.2 Resultat av tester

Resultatene av testene stammer fra testene som blir beskrevet i kapittel 3.7. Under samtlige tester ble sikkerhetsaspekter rundt brann- og klemfare vurdert. Informasjon om dette kan leses i kapittel 2.4 som omhandler sikkerhet.

4.2.1 Fjernstyring av ATV

Før ATVen ble demontert og endret ble det kjørt test 3.7.1 som omhandler oppførselen til ATVen med sommerprosjektets styremåte [2]. ATVen var rask og nøyaktig med styringen igjennom RCen uten noen merkbar treghet eller unøyaktighet.

Kalibreringen utføres ved å manuelt vri styret på ATVen slik at hjulene står i maksimalt utslag mot venstre før oppstart. Metoden er tungvint og krever at systemet startes helt på nytt for hver kalibrering. Dette resultatet viser at det trengs en ny kalibreringsprosedyre, mens for manuell kontroll er dagens styremåte tilstrekkelig.

4.2.2 Kommunikasjon mot MK over RS-232

Da alle koblingene var gjort slik som beskrevet i kapittel 3.7.2 og Terra Term var åpen, ble dette mottatt data som vist i Figur 4.14.

Vi visste da at vi kunne motta data fra MKen. For å teste om sending av ordre fungerte ble det sendt over noen av de generelle ordrene på side 34 i [19].

«Reset!» restartet MKen og man fikk «bootup»-meldingen på nytt.

«Version?» returnerte «E300:VERSION:01.4» som beskriver «firmware versjonen» til MKen.

```
*** DSP 56F8357 Boot-  
loader (c) Metronix ***  
  
Bootcode : 0001.0008  
  
Waiting...  
  
Checksum P-Flash: OK!
```

Figur 4.14 «Bootup»-melding sendt
fra MK over RS-232

Resultatene var som forventet og kommunikasjonen ble ansett som fungerende.

4.2.3 Test av «Homing» over RS-232

Etter å ha parametrisert «homing» i FCT som beskrevet i kapittel 3.3, og gjennomført test 3.7.2 ble så test 3.7.3 gjennomført.

I gjennomføringen av testen startet motoren å rotere frem til den traff grensebryteren før den stoppet og indikerte at homing var fullført ved hjelp av syv segment displayet. Deretter var MKen klar for «Profile Positioning».

4.2.4 Test av «Profile Positioning» over RS-232

Det viste seg etter et forsøk på å følge eksempelet på side 38-40 [19], at vi fikk en feilmelding når vi skulle starte posisjoneringen. Denne viste seg å være på grunn av feil verdi av «SW limit negative» og «Axis zero point».

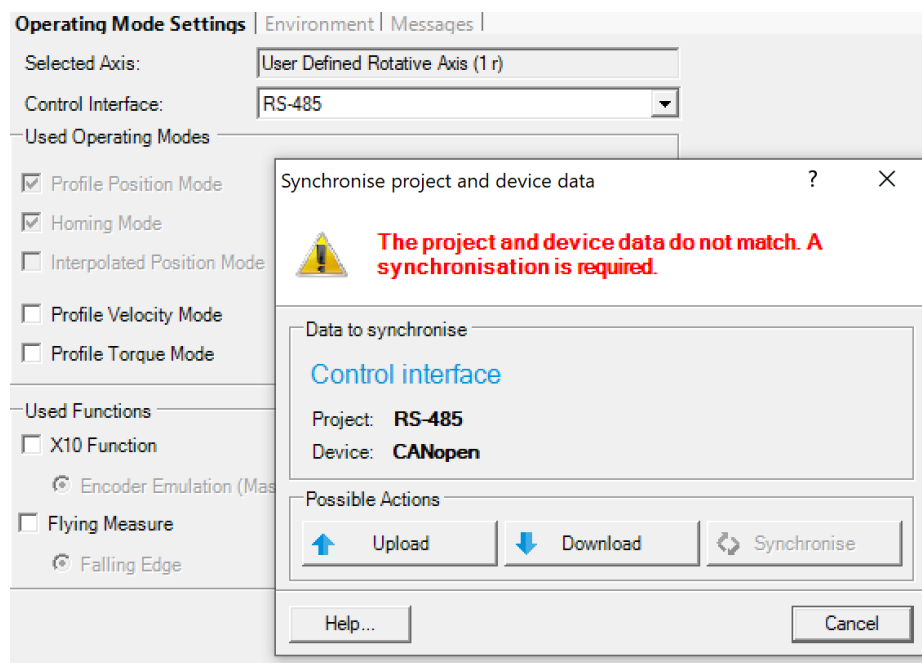
Rekkefølgen av disse ble endret slik at det er tre posisjons grenser. Den første er «Axis zero point» på 0,010 rotasjoner, nummer to er «SW limit Negative» som er på 0,005 rotasjoner og den siste er grensebryteren som per definisjon er 0,000 rotasjoner.

Ved å bruke endre bit 7 i «data» ble feilen nullstilt, og posisjoneringen fungerte som den skulle. Metoden som ble brukt til å finne fram til dette står forklart i kapittel 3.7.4. Den totale ordren som ble sendt var «=604000:009F» for å starte posisjoneringen.

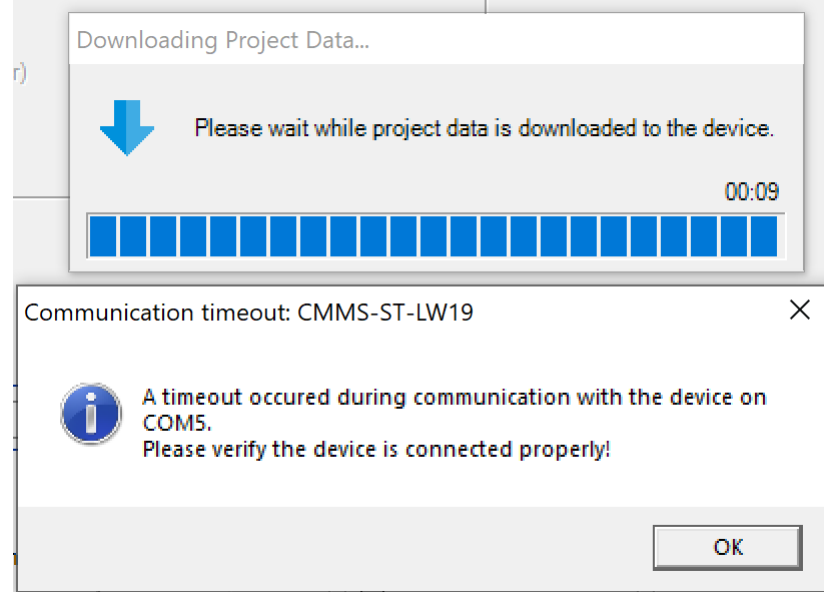
Posisjoneringen ble startet og motoren beveget seg til posisjonen som var spesifisert.

4.2.5 Parametrisering til RS-485

Da vi fulgte metoden i kapittel 3.7.5 oppstod det en feil. Figur 4.15 og Figur 4.16 viser de aktuelle skjermbildene. Kontakten til MKen ble brutt så snart vi prøvde å laste ned konfigurasjonen til MKen. Det ble bekreftet med support fra Festo at metoden som ble brukt var riktig. Et arkiv av prosjektet ble sendt på e-post til support for videre feilsøking.



Figur 4.15 Nedlastning av program til MK over RS-232. (Skjermbilde fra FCT)



Figur 4.16 Feilmelding ved nedlastning av program. (Skjerm bilde fra FCT)

Som resultat av dette ser gruppen seg nødt til å bytte kommunikasjons protokoll over til RS-232, med simulering av CANOpen protokollen.

4.2.6 Test av Arduino TTL til RS-232

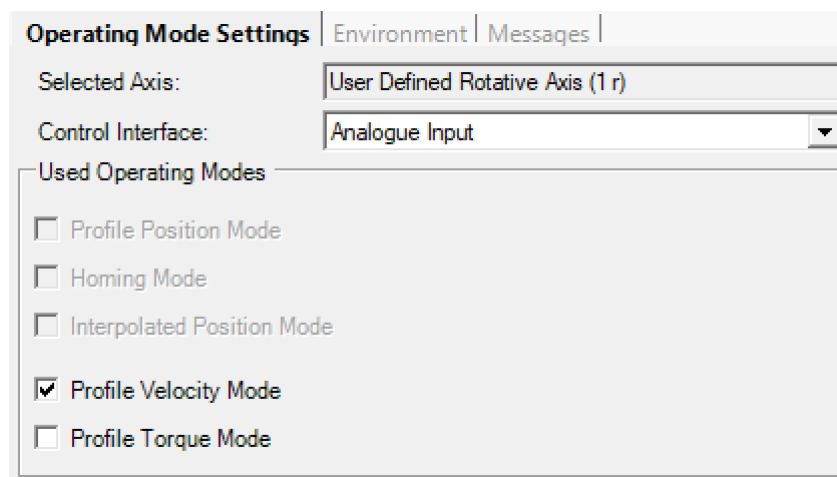
Som beskrevet i kapittel 3.7.6 ble Arduinoen koblet til RS-232 kretskortet igjennom «TX» og «RX» pinnene. D-sub 9 kontakten ble så koblet til USB til RS-232 adapteren som deretter ble koblet i USB porten på PC-en.

Terra Term ble parametrisert til 9600 baud og Arduinoen ble så skrudd på.

Vi kunne dermed lese det samme som ble sendt fra Arduino programmet på skjermen i klar tekst.

4.3 Valg av kommunikasjons-protokoll

Etter gjennomgang av kommunikasjonsmetoder som står beskrevet i kapittel 2.1.1, ble det utelukket flere mulige protokoller på bakgrunn av at det ikke er noen støtte for de med Arduino. RS-485 og simulering av CANOpen ved bruk av RS-232 stod frem som mulige valg siden de er basert på seriell kommunikasjon, noe som støttes av Arduinoen ved bruk av en enkel IC. Resultat 4.2.1 viste at den analoge protokollen var tilstrekkelig for nøyaktighet og respons. Men som vist i Figur 4.17 støtter ikke denne protokollen «homing» metoden for automatisk kalibrering siden den bare kan brukes med «velocity» og «torque» bevegelses profiler.



Figur 4.17 Analog styremåte, og profilene den støtter. (Skjerm bilde fra FCT)

Resultatene av test 4.2.2, 4.2.3 og 4.2.4 viste til at RS-232 kunne være ett fullstendig alternativ til RS-485. Det viste seg med resultatet 4.2.5 at det ikke ville være mulig å bruke RS-485 videre, derfor falt valget av kommunikasjonsprotokoll til simulering av CANOpen over RS-232.

4.4 Grensebryter

For å sikre rask implementering av homing funksjonen ble det søkt etter brytere på nett grossisten «Elfa Distrelec» [39]. Ved å bruke et filter på søket som bestod av 24 V spenning, stod valget mellom Siemens og Eaton. Gruppen har mest erfaring med Siemens samtidig som de har et godt utvalg av ekstrautstyr til grensebryterne sine.

Etter vurdering av kapittel 2.1.9 og 2.1.10 kom vi frem til at tilfeller der bevegelsen fra en aktuator direkte påvirker en bryter for å oppnå en presis posisjonering vil en «slow action» bryter være å foretrekke over den umiddelbare, men mindre presise, «snap action» bryteren.

Gruppen kjøpte inn en «Slow action», «plunger» type grensebryter med identifisering «0BC05» og en «Spool lever» med identifisering «0AE10» som sikrer at grensebryteren ikke blir skadet ved for store påkjenninger.

4.5 «Homing»

Under gjennomføring av «homing» kreves det forståelse om aspekter rundt virkemåte, giring og programvare. Under kapittel 3.3.4 blir «homing» parametrisert i FCT. Dette vedvarer når programmet ber MKen om å gjennomføre «homing».

4.5.1 Hardware

I sammenheng med giret som gir 9 rotasjoner av drevet for hver rotasjon av stepper-motoren, som beskrevet i kapittel 2.1.6, kan man regne det som sannsynlig at det vil oppstå flere «zero pulser» i hver retning før en mekanisk grense treffes. Posisjonen til zero pulsen må bli funnet og kalibrert for hver gang motoren flyttes fra ATVen, noe det ikke finnes noe verktøy for å gjøre.

Ratt stammen er konstruert for kreftene som påføres av et menneske, den vil ha for stor hysteres mellom posisjonen der hjulene stopper å svinge og, posisjonen der motoren gir 90 % av sin maksimale kraft, til å gi en nøyaktig nok referanse punkt.

Ved å bruke en grensebryter som er «slow action» og er montert på rammen, isolert fra bevegelsene til aktuatoren, og alltid i samme posisjon, vil man ha et fast punkt å kalibrere drevet mot ved demontering, samt et stabilt referansepunkt med meget liten hysteres.

En god metode for å redusere slitasje på grensebryteren samt å øke buffer sonen mellom den mekaniske grensen og aktuatoren sin programmerte grense er å legge inn en buffer sone. Denne buffer sonen vil legges imellom referanse posisjonen funnet med grensebryteren og programmert nullpunkt. Dette vil ikke bli overskredet ved normal operasjon av aktuatoren.

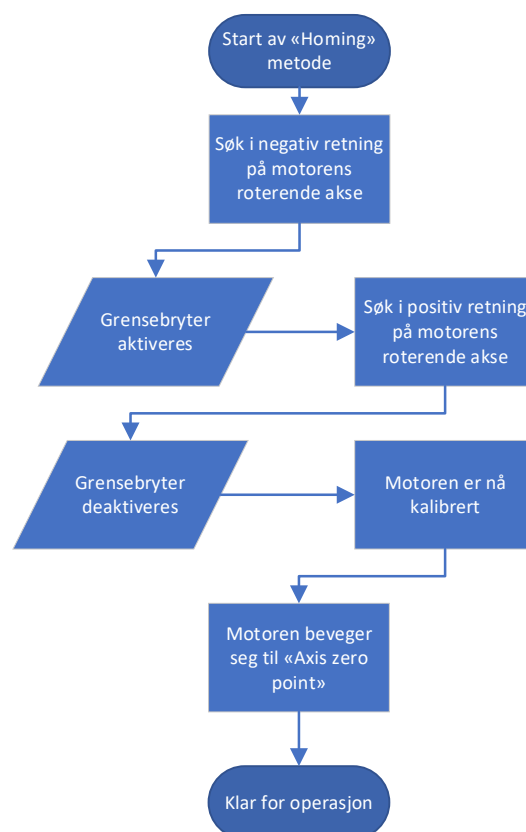
4.5.2 Parametere til motorens roterende akse

For å forstå bedre hvilke valg som ble gjort refereres det til Tabell 3-2 fra side 76 i «function and commissioning» dokumentet [3]. For en enkel introduksjon til metoden og posisjonene funnet med testen se Figur 4.18 og Tabell 4-3.

Resultatet for «Homing» parameterne er funnet med test 3.7.3 og er som følger:

Søk etter grensebryter utføres i negativ retning. Når denne aktiveres mekanisk blir posisjonen lagret som «Axis zero point». Stepper-motoren forflytter seg så 0,010 rotasjon i positiv retning og setter «Project zero point» her. Dette blir nullpunktet for alle videre posisjoner.

Verdiene for negativ og positiv software (SW) grensebryter blir satt mellom «Axis zero point» på -0,010 rotasjoner og «Project zero point» på 0 rotasjoner. Negativ SW bryter får altså posisjonen -0,005 rotasjoner og vil være en buffer før man treffer HW grensebryteren. Positiv SW bryter blir plassert på 0,255 rotasjoner slik at den er 0,005 rotasjoner høyere enn styrevinkelen.



Figur 4.18 Flytskjema av "Homing" prosedyre

Tabell 4-3 Posisjoner på motorens roterende akse

Posisjon:	Grader	Rotasjoner	Heksadesimal
Grensebryter = «Axis zero point»	-3,6 °	-0,010 r	<i>Utenfor limit</i>
«Negative software limit»	-1,8 °	-0,005 r	<i>Utenfor limit</i>
«Project zero point»	0 °	0 r	0000 0000
«Rett frem»	45 °	0,125 r	0001 1FEE
Styrevinkel range	90 °	0,250 r	0002 3FDC
«Positive software limit»	91,8 °	0,255 r	<i>Utenfor limit</i>
SW limit range	93,6 °	0,260 r	<i>Utenfor limit</i>
HW limit range	97,2 °	0,270 r	<i>Utenfor limit</i>

4.6 Kretskort for kommunikasjon mot motorkontrolleren

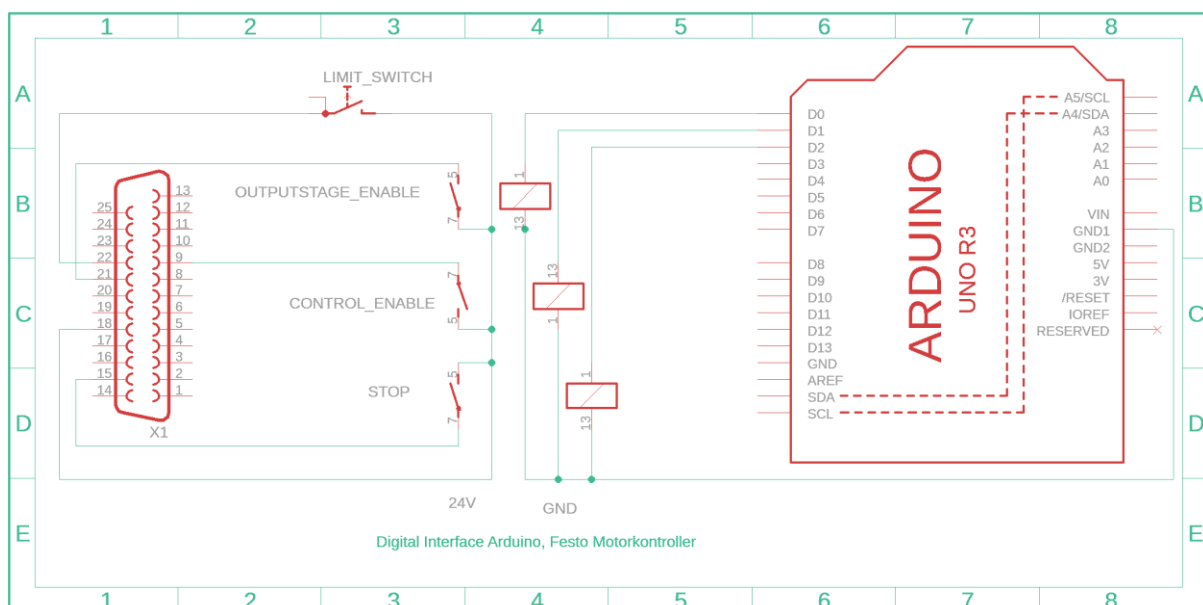
Kretskort er en vital komponent for at en mikrokontroller kan kommunisere med en industriell MK. Design og oppkobling av kretskort har bidratt til et fungerende system som er brukervennlig for videre arbeid.

4.6.1 Digital I/O på motorkontrolleren

Som beskrevet i kapittel 3.5.3 var det behov for styring av de digitale inngangene til MKen. Den enkleste måten å utføre dette på var med 5 V reléer satt på et kretskort, der spolen blir aktivert med de digitale utgangene på Arduinoen.

24 V utgangen av MKen brukes til å tilføre spenning til relé-bryterne samt grensebryteren. Dette gir en form for isolering mellom 24 V spennings-nivået på MKen og 5 V spennings-nivået på Arduinoen.

Vedlagt er et utsnitt av det endelige kretskortet, designet i «Eagle PCB» som vist i Figur 4.19 Skjematisk tegning av Digital I/O:

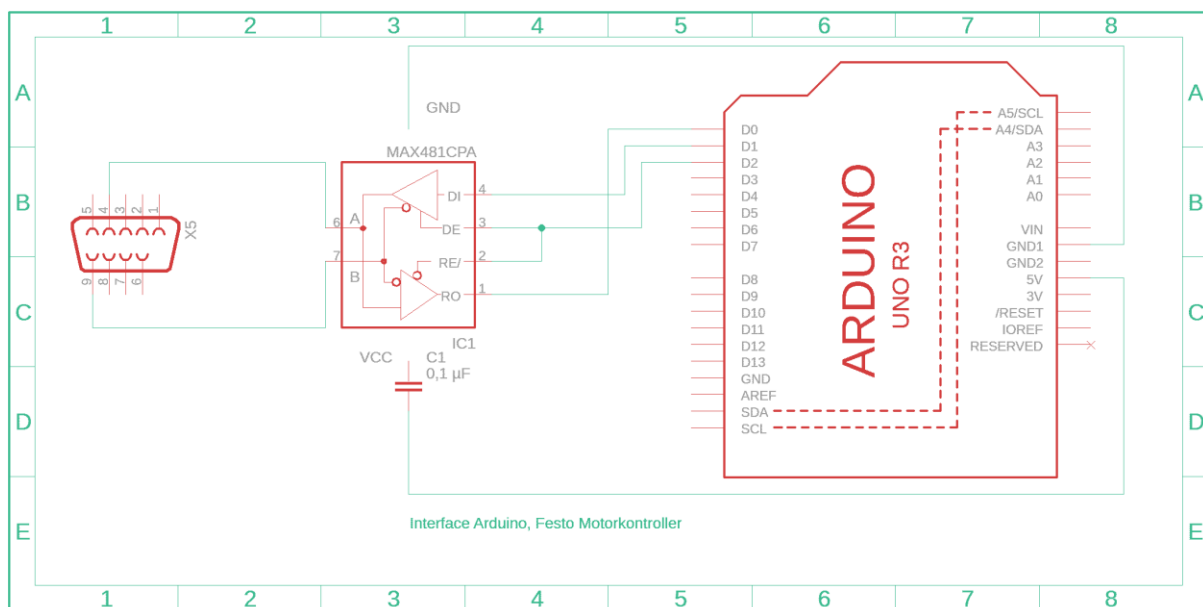


Figur 4.19 Skjematisk tegning av Digital I/O kretskort

4.6.2 RS-485

Det ble designet og laget et kretskort for kommunikasjon mellom Arduino og RS-485. Dette ble gjort før problemstillingen rundt RS-485 som forklart i kapittel 4.2.5. Metoden for å lage dette kretskortet, som skal være fullstendig operasjonelt er beskrevet i kapittel 3.5.1.

Vedlagt er et utsnitt av det endelige kretskortet, designet i «Eagle PCB» som vist i Figur 4.20 Skjematisk tegning av RS-485 interface:



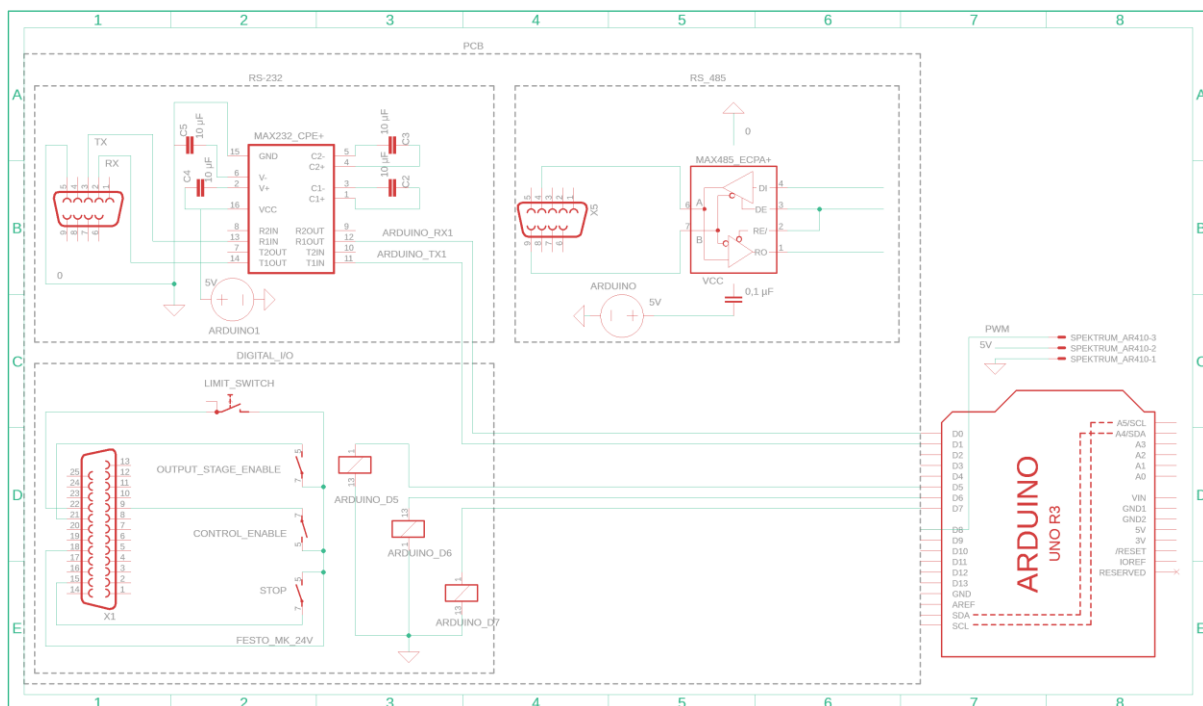
Figur 4.20 Skjematisk tegning av RS-485 interface kretskortet

4.6.4 Kombinert kretskort for RS-232, RS-485 og digital I/O

Da alle kretskortene var designet og testet hver for seg ble metoden i kapittel 3.5.4 fulgt for å kombinere de til et kretskort.

Det endelige kretskortet har både RS-232 og RS-485 ICer påmontert, slik at fremtidige grupper enkelt kan bytte til en mer bus-orientert løsning om det skulle trenge, men kun RS-232 er koblet til Arduinoen.

Vedlagt er koblingsskjema av det endelige kretskortet, designet i «Eagle PCB» som vist i Figur 4.22:



Figur 4.22 Skjematisk tegning av det kombinerte kretskortet

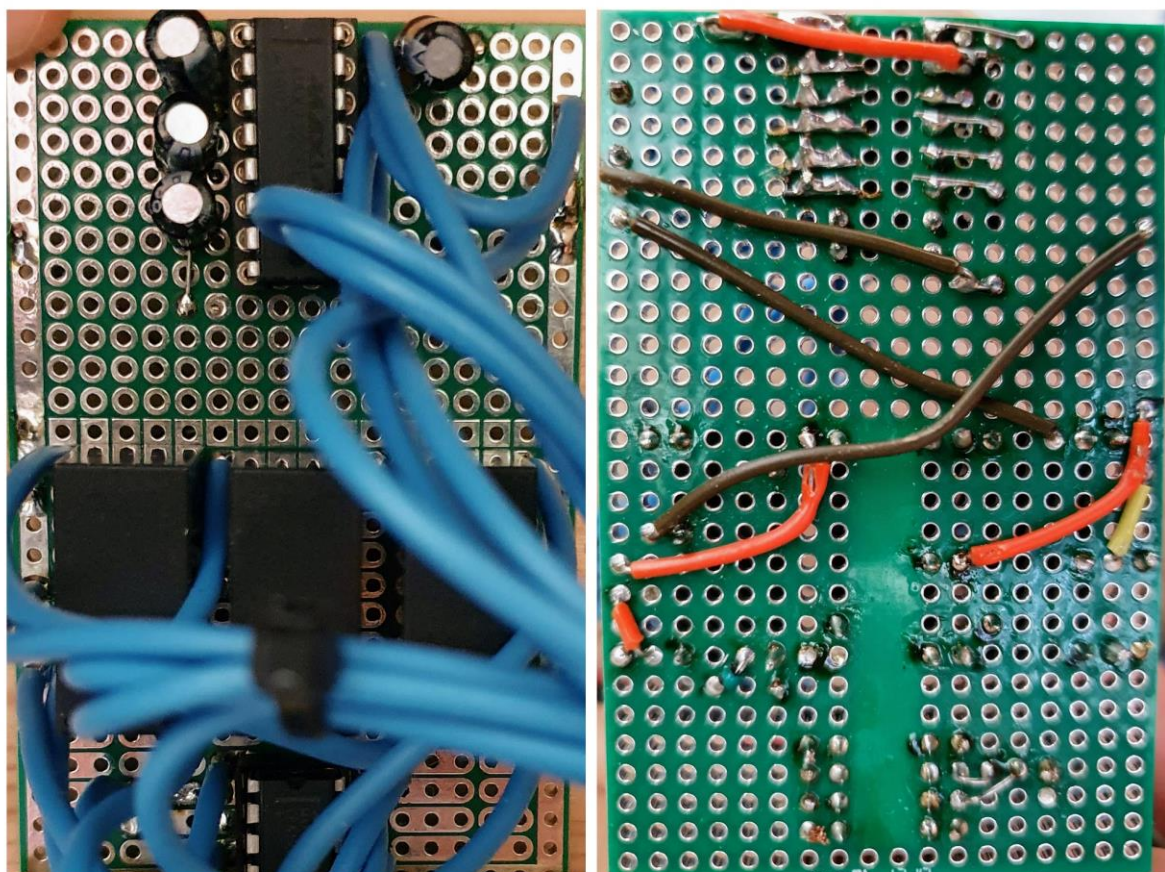
Det ble brukt et kretskort av typen «FR4 Prototyping board». Denne har rundt-gående jord tilførsel og design for enkel lodding av ICer.

Siden MAX485 ICen ble loddet på før gruppen fant ut at de ble nødt til å bytte til MAX232, er denne ICen fortsatt på kretskortet. Alle ledningene for tilførsel av spenning, jord og kommunikasjon er fortsatt på plass, men ikke tilkoblet.

ICen som er i bruk i dag er en MAX232 som kan sees øverst på Figur 4.23 med tilhørende kondensatorer. Det ble laget egne baner for å følge metoden beskrevet i kapittel 3.5.2. Dette ble nødvendig på grunn av mangel på plass på de egnede stedene på kretskortet.

Resultatet av loddingen er noe som kan kategoriseres som en første utkast prototype for prosjektet, med mange muligheter for forbedringer. Virkemåten til kretskortet er utmerket og uten noen feil.

Vedlagt er bilde av over- og underside av det endelige kretskortet som vist i Figur 4.23:



Figur 4.23 Bilde av kretskort (privat foto)

4.7 Bygging av testtrigg

Testtriggen er bygget som en tilnærming av å være montert på ATVen. Tanken er at kabling ikke skal behøves endres for å montere styresystemet tilbake, og at den skal være en sikker plattform å arbeide på og lære systemet.

4.7.1 Design

Testtriggen er konstruert som en kasse uten to av sidene. Det er montert inn en plate midt inne i kassen for å styrke konstruksjonen, samt for å gi plass til å montere komponenter. Steppermotoren er plassert vertikalt og er skrudd til kassens overside, mens batteriet står festet til gulvet i kassa. Tilførsel og komponenter er skilt fra hverandre slik at batteri og 24 V DC omformereren er plassert på den ene siden av midtplaten, mens MKen, Arduino og kretskort er montert på den andre siden. Siden batteriet tas ofte ut og er den tyngste delen i testtriggen er det laget en dedikert plass til den med lastestropp over for å holde den på plass. Omformereren har en svært lav vekt og er skrudd fast til midtplaten ovenfor batteriet.

4.7.2 Montasje

For å bygge kassen til testtriggen er det brukt bord-sag, kapp og gjærsag, batteridrill, stikksag trebor og tre skruer. Selve testtriggen er laget av furu som gjør den lett å jobbe med og rimelig. Ved å montere platen midt i kassen ved siden av MKen fikk man utnyttet mest mulig av materialene, samtidig som at alle komponentene inni er mulig å jobbe på uten å måtte fjerne andre komponenter i mellomtiden.

For å montere komponentene og ved oppkobling ble det brukt mye eksisterende komponenter fra ATVen og noe innkjøpt fra flere produsenter kjøpt gjennom to leverandører. Leverandørene var nettbutikken Elfa Distrelec [39] og lokal Biltema butikk [40]. Vi valgte Elfa Distrelec på grunn av sitt brede utvalg, rask levering og tidligere positive erfaringer. Biltema ble brukt for deres brede utvalg og tilgjengelighet her i distriktet.

4.7.3 Komponenter og verktøy brukt i bygging av testtrigg

Oppkobling av komponenter ble gjort med tanke på god kontakt og unngå mulige feilkilder. Under prøvemonteringen ble alle signal ledninger ut fra MKen klipt i samme lengde. Db9 og Db25 kontakten ble ferdig loddet med de relevante utgangene. Under test 3.7.3 og 3.7.4 ble «Control Enable», «Stop» og «Output Stage Enable» koblet sammen med «+24 VDC» via en Klikk-Wago (Wago koblingsklemme). Dette utgjorde samme funksjon som utgangene på et relée, og ga oss mulighet til å se om vi hadde alle utgangene rett satt opp for å kunne kjøre MKen i «Terra Term». Etter fullført test ble koblingene loddet til reléer på kretskort styrt av Arduino. Videre på kretskortet befinner det seg en IC holder («IC Socket») og en MAX232 IC. ICen gjør det mulig å kommunisere med RS-232 via seriell utgang på Arduino. Grensebryteren er montert på oversiden og blir trigget med en berøringsstav festet på drevet.

I Tabell 4-4 er det listet opp komponentene som er benyttet for å konstruere testtriggen som ikke fulgte med fra ATVen.

Tabell 4-4 Testtrigg komponenter

Komponent	Bruksområde	Leverandør
Db25 kontakt	Festo MK	Elfa Distrelec
Db9 kontakt	Festo MK	Elfa Distrelec
0,5mm Cu ledning blå	Alle områder	Elfa Distrelec
Kabelsko m/krympestrømpe	Batteri, omformer og bryter	Biltema
Klikk-Wago	Midlertidig oppkobling	Privat
Batteri 44 A timer	Batteri	Biltema
Lastestropp	Feste av batteri	Biltema
Tre skruer	Konstruksjon	Biltema
Hyllplater Furu	Konstruksjon	Biltema

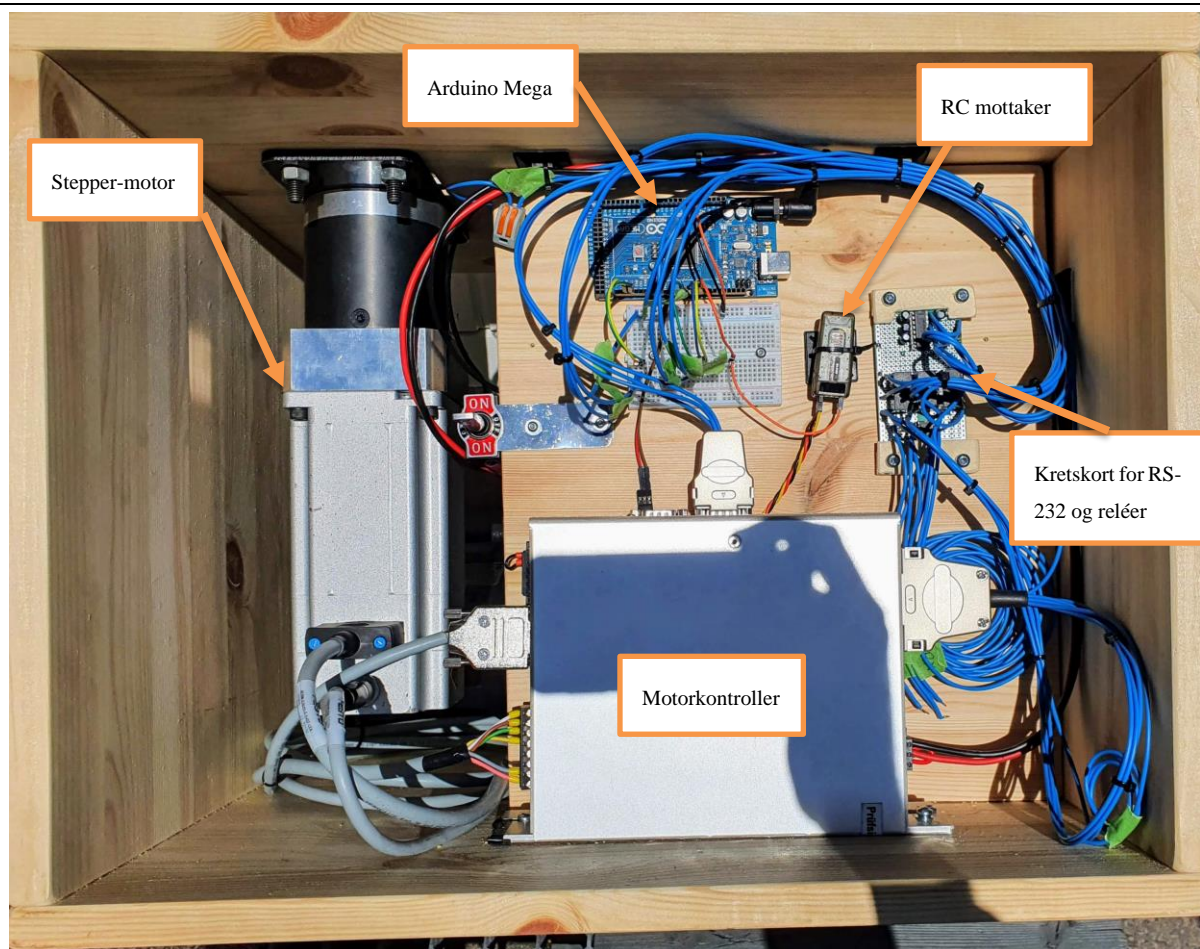
MAX232 IC	Kretskort	Elfa Distrelec
MAX485 IC	Kretskort	Elfa Distrelec
IC holder 8 bein	Kretskort	Elfa Distrelec
IC holder 16 bein	Kretskort	Elfa Distrelec
Sikringsholder	Batteri	Biltema
15 A flatstiftssikring	Batteri	Biltema
Bryter 12 V	Batteri	Biltema
5 V Rel�er	Kretskort	Elfa Distrelec
Arduino MEGA 2560	Arduino	Elfa Distrelec
DSMX DXe RC Mottaker	Arduino	Fra ATV
FR4 Prototyping Board	Kretskort	Elfa Distrelec
Grensebryter Siemens	Grensebryter	Elfa Distrelec
L�pehjul for grensebryter	Grensebryter	Elfa Distrelec

Verktøy som ble brukt under arbeidet som ikke omhandler byggingen av kassen til testtriggen er:

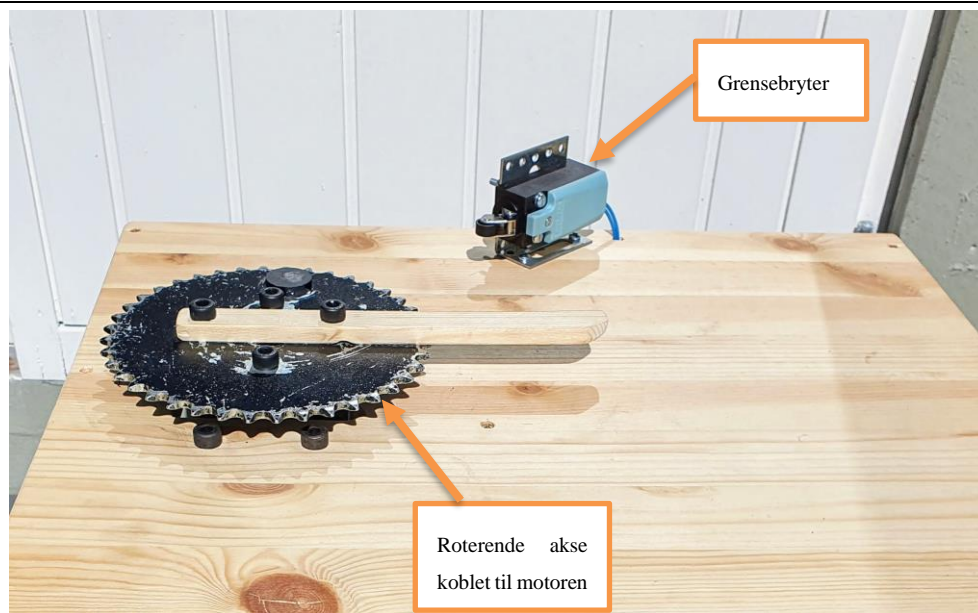
- Sekskantnøkkel-sett i mm (Umbrako)
- Fastnøkler i mm
- Bitssett
- Skrujern ment for elektronikk
- Avbitertang
- Kabelsko tang
- Avmantlingstang
- Varmepistol
- Loddebolt m/justerbar temperatur
- Flussmiddel
- Multimeter

Det er lurt å ha en loddebolt med flat liten tupp og regulerbar temperatur. Dette, riktig bruk av flussmiddel og lodde tinn i riktig tykkelse gjør det svært mye lettere å få gode loddinger. Med gode loddinger menes det å ha god forbindelse med riktig mengde tinn mellom elementet og punktet som vist på Figur 4.23. Multimeter med Ohm-funksjon bidrar til å verifisere at kortet ikke har noen feil koblinger. Smeltede lasker eller feil på koblinger kan føre til kortslutning og ødelagte komponenter.

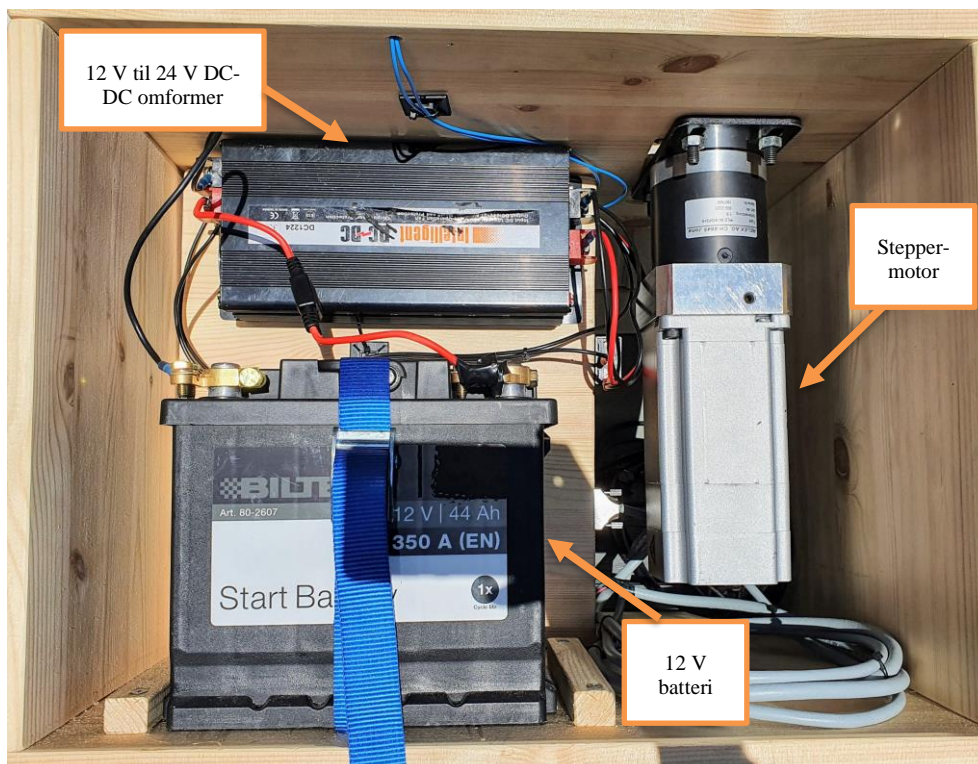
Figur 4.24, Figur 4.25 og Figur 4.26 viser endelige resultat med beskrevne deler.



Figur 4.24 Testrigg sett forfra (privat foto)



Figur 4.25 Testrigger ovenfra (privat foto)



Figur 4.26 Testrigger bakfra (privat foto)

5 Diskusjon

Dette prosjektet ble preget av spesielle omstendigheter og tekniske utfordringer som diskuteres i dette kapitlet.

5.1 Korona situasjonen

Ved start av prosjektet ble det bestemt at alt arbeid skulle foregå på USN sin signallabb og prosesshall. Dette ble besluttet på grunn av at ATVen står tryggere der, samt tilgang til utstyr.

Det ble fort klart at situasjonen med korona viruset hindret gruppen i å få arbeide fysisk på universitetets områder, noe som igjen hindret fremgangen til prosjektet. Etter to uker med skrive- og planlegging fikk gruppen innvilget en søknad om å hente deler av styresystemet fra prosesshallen. Dette var deler som gjorde det mulig å lage en testtrigg for simulert styring.

Det ble bygget en testtrigg som beskrevet i kapittel 3.6 og 4.7 som deretter ble basis for videre tester og utvikling. Denne løsningen viste seg å forenkle arbeidet betraktelig ettersom alt utstyret var samlet på et sted og var lett tilgjengelig og modifiserbart.

Utstyret skal monteres tilbake på ATV for videre utvikling etter endt bacheloroppgave.

5.2 Problemer med parametrisering

Da gruppen skulle begynne å teste RS-485 kommunikasjonen oppstod det et problem:

Når MKen ble parametrisert til RS-485 og det ble forsøkt å lagre disse innstillingene i MKen kom det opp en feilmelding, og PC-en mistet konsekvent kommunikasjon med MKen. Det viste seg at de nye parameterne ikke ble lastet over grunnet tapet av kommunikasjon, og MKen var fremdeles stilt inn på CANOpen kommunikasjon per testen i kapittel 3.7.2.

Gruppen tok kontakt med Festo sin support telefon der det raskt kom frem til at problemet måtte ligge i selve MKen og ikke i parameterne og PC-en. Et alternativt ble fremmet for å få tilsendt en ny MK som var ferdig parametrisert, men dette ble stemt ned på grunn av tidsmangel. En løsning var å bytte kommunikasjon til RS-232 med CANOpen simulering ettersom det allerede var utført suksessfulle tester med denne metoden som vist i kapittel 3.7.4. For å bytte til denne metoden trengtes det et nytt kretskort design med en annen IC, en MAX232.

5.3 Videre forbedringer

Dette kapittelet presenterer noen av mulighetene for videre utvikling og vil gi inspirasjon for arbeidet videre.

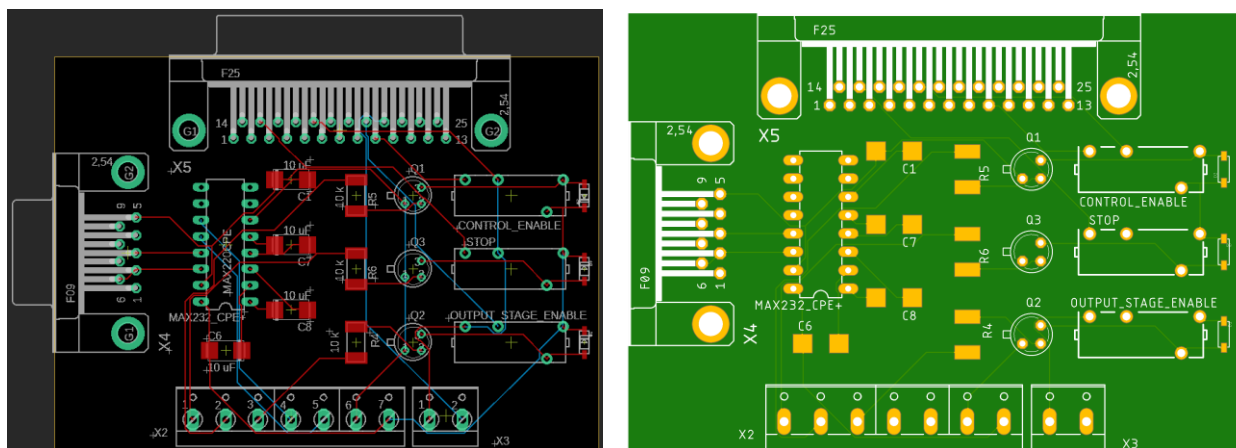
5.3.1 Transistor styring av reléer

For å lette strøm-trekken fra de digitale utgangene på Arduinoen vil det være et bedre design å implementere en transistor for å styre reléet. Denne vil være plassert i serie mellom jord og 5 V på spolen til reléet og vil aktiveres av de digitale utgangene på Arduinoen. Det vil også være en diode i parallell over spolen på reléet for å hindre overspenning.

5.3.2 Ny revisjon av kretskort

I en revisjon to ville vi tatt med forbedringer som de beskrevet i kapittel 5.3.1 og implementert det eksisterende designet i Figur 4.22. Deretter kan det bestilles to prototype kort fra en leverandør med ferdig etsede baner, hull for komponenter og markering for plasseringer. Dette ville resultert i mye mindre ledninger samt et mer oversiktlig og robust design.

Et eksempel laget i «Eagle PCB» kan sees i Figur 5.1:



Figur 5.1 Render fra «Eagle PCB» av et forbedret kretskort

5.3.3 Avlesning av aktuell informasjon

«CiA 402» protokollen støtter direkte avlesning av ulik informasjon på MKen. Å implementere en lesning av feilkoder ville tillatt en fremtidig HMI å vise frem alarmer og feilmeldinger som kunne oppstått. For det autonome systemet kan motorens posisjon være aktuell. I «Assembly and installation» [19] dokumentet til Festo står det at man kan sende inn ordre med kommando '=606400' for avlesning av motorens aktuelle verdi [30]. Andre ordre for uthenting av aktuell informasjon står forklart videre i dette dokumentet.

5.3.4 Optimalisering av kommunikasjon og bevegelse

For å optimalisere bevegelser og oppstart av systemet, samt redusere belastningen på Arduinoen og RS-232 protokollen kan man bruke de digitale utgangene på MKen.

Det finnes tre programmerbare utganger som beskrevet i kapittel 3.3.6. Ved å bruke FCT kan man endre på disse slik at de blir høye ved spesifikke hendelser, alle eksemplene er beskrevet på side 58 – 60 i «Installation and commissioning» dokumentet [3].

6 Oppsummering

Vi har til tross den spesielle situasjonen klart å oppnå kravene i oppgavebeskrivelsen for prosjektet. Delmålene om forbedring av strømtilførsel, fiksing av kontroll-boks til det autonome systemet og henting speedsensor signal faller bort grunnet stengt universitet. Det samme gjør hovedmålene om å implementere de nye endringene i det autonome systemet og å kartlegge hele Lone Wolf sitt system.

Det ble under semesteret arbeidet jevnt med oppgaven. Gode sikkerhetstiltak for smittevern ble fulgt i form av arbeid ut av privat bolig og begrenset omgangskrets.

Rapporten kartlegger hvilke deler av styresystemet som er nødvendig for prosjektet. Dette inkluderer 12 V strøm tilførsel, 24 V omformer, Arduino MEGA, MK og stepper-motor. Disse er derfor isolert fra det autonome systemet og implementert i en egen testtrigg. Denne testtriggen er brukt videre i utviklingen av prosjektet.

Alle metoder og resultater er detaljert beskrevet, med fokus på sikkerhet, læring og reproduserbarhet. Resultatet fra testen i kapittel 4.2.5 utelukket kommunikasjons protokollen RS-485 grunnet problemer med prosjektets MK. Som dens erstatning støtter også RS-232 CiA 402, med begrensningen at det ikke er en buss-protokoll. Med RS-232 standarden oppfyller vi målene for prosjektet om å implementere «homing mode» og «profile positioning mode». RS-232 blir brukt som styremetode sammen med et dødbånd-filter programmert inn i Arduinoen.

Ett kretskort ble designet med «Eagle PCB» og det ble kjøpt inn komponenter for å bygge en prototype. Kretskortet ble bygget for hånd, testet og implementert i testtriggen. Ved videre arbeid kan det tas utgangspunkt i de allerede designede kretskortene og bestille et ferdig etset og freset prototype kretskort. Dette vil gjøre styresystemet mere robust og oversiktlig.

Ved oppstart gjennomfører styresystemet kalibrering av stepper-motoren. Dette gjør at operatøren kun trenger å slå på en bryter for å gjøre styringen operasjonell. Styresystemet er i dag montert i en selvkomponert testtrigg som simulerer miljøet på ATVen. Testtriggen styres med en fjernkontroll og driftes med et 12 V bil-batteri slik at den har lang operasjonstid. Dette gir også fordeler som at den er mobil og ikke krever en ekstern strømtilkobling.

Vi ser på testtriggen som en fordel for neste gruppe som skal arbeide på ATVen. Den gir muligheten for at fremtidige grupper som involveres i Lone Wolf enklere skal lære systemet å kjenne. Samtidig kan testtriggen jobbes på parallelt med andre systemer på ATVen.

Noen av forbedringene som kan gjøres i videre arbeid er mer fullstendig implementering av CiA402 protokollen. Dette vil innebære et mer fullstendig Arduino program som tar for seg flere aspekter av protokollen slik at man kan lese alarmer og posisjoner direkte fra MKen.

7 Referanser

- [1] M. M. Helle, T. T. Tokvam, K. V. Leirvik, S. A. Mørtvedt, L. T. Thorud og O. Loe, «Guided Wolf,» Universitetet i Sørøst Norge, Kongsberg, 2019.
- [2] C. S. V. Berg , J. Løken, S. G. Løkken, V. B. Abrahamsen, S. Bøe, C. Lia, A. Hauglie-Hanssen, M. S. Mullins , M. Jervan og N. S. Semb, «Lone Wolf 2019,» Kongsberg, 2019.
- [3] Festo SE & Co. KG, «CMMS-ST Commisioning,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/Downloads/344923/349395/CMMS_D-FW_2014-04_8034520g1.pdf. [Funnet 10 april 2020].
- [4] W. contributors, «Interpreter (computing),» Wikipedia, The Free Encyclopedia, [Internett]. Available: [https://en.wikipedia.org/w/index.php?title=Interpreter_\(computing\)&oldid=947588891](https://en.wikipedia.org/w/index.php?title=Interpreter_(computing)&oldid=947588891). [Funnet 9 april 2020].
- [5] J. Gisle, «Store Norske Leksikon,» [Internett]. Available: <https://snl.no/autonom>. [Funnet 9 april 2020].
- [6] Vocabulary, «Vocabulary,» [Internett]. Available: <https://www.vocabulary.com/dictionary/autonomous>. [Funnet May 2020].
- [7] Kambria, «Kambria,» 23 June 2019. [Internett]. Available: <https://kambria.io/blog/the-history-and-evolution-of-self-driving-cars/>. [Funnet may 2020].
- [8] Wikipedia contributors, «Self-driving car,» 13 May 2020. [Internett]. Available: https://en.wikipedia.org/w/index.php?title=Self-driving_car&oldid=956436062. [Funnet 14 May 2020].
- [9] Velodyne Lidar inc., «Velodyne puck,» [Internett]. Available: <https://velodynelidar.com/products/puck/>. [Funnet 11 april 2020].

- [10] NovAtel inc., «RTK GPS,» [Internett]. Available: <https://www.novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/real-time-kinematic-rtk/>. [Funnet 11 april 2020].
- [11] Festo SE & Co. KG, «FESTO CMMS-ST Controller overview,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/Downloads/84308/11756/cmms-st_en.pdf. [Funnet 11 april 2020].
- [12] Festo, «Festo Stepper Motor EMMS-ST,» [Internett]. Available: https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/EMMS-ST_EN.PDF.
- [13] Festo SE & Co. KG, «EMMS ST with CMMS ST,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/Downloads/9172/12885/PSI_141_3_en.pdf. [Funnet 19 05 2020].
- [14] Arduino AG, «Arduino Mega 2560 dokumentasjon,» [Internett]. Available: <https://store.arduino.cc/arduino-mega-2560-rev3>. [Funnet 06 05 2020].
- [15] Arduino CC, «Arduino FAQ,» [Internett]. Available: <https://www.arduino.cc/en/Main/FAQ#toc13>. [Funnet 11 april 2020].
- [16] Festo SE & Co. KG, «Festo Mounting and Installation CMMS-ST-G2-HW-EN,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/Downloads/363759/349412/CMMS-ST-G2-HW_2014-04_8034456g1.pdf. [Funnet 9 april 2020].
- [17] Shneider Electric, «Shneider Electric, snap action contacts,» [Internett]. Available: <https://www.se.com/uk/en/faqs/FA22871/>. [Funnet 10 april 2020].
- [18] National Instruments, «National Instruments encoder,» [Internett]. Available: <http://www.ni.com/tutorial/7109/en/>. [Funnet 06 05 2020].
- [19] Festo SE & Co. KG, «CMMS-CO CANOpen,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/Downloads/385513/403859/CMMS-CO_2010-12a_554352g1.pdf. [Funnet 9 april 2020].

- [20] Festo SE & Co. KG, «FESTO Assembly and installation,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/Downloads/363758/349405/CMMS-ST-G2_2010-08_573125g1.pdf. [Funnet 01 05 2020].
- [21] CiA association, «CiA association,» 2020. [Internett]. Available: <https://www.can-cia.org/about-us/>.
- [22] Festo SE & Co. KG, «Support portal CMMS-ST,» [Internett]. Available: https://www.festo.com/net/no_no/SupportPortal/default.aspx?cat=1530&tab=3&s=t#result. [Funnet 9 april 2020].
- [23] M. Khamlichi, «Unixmen,» April 2016. [Internett]. Available: <https://www.unixmen.com/dennis-m-ritchie-father-c-programming-language/>. [Funnet 01 May 2020].
- [24] Arduino, «Arduino Programmeringshjelp,» [Internett]. Available: <https://www.arduino.cc/reference/en>. [Funnet 24 04 2020].
- [25] P. D. o. H. Deitel, Visual C# How to program 6th edition, London: Deitel & Associates, Inc, 2017.
- [26] Arduino CC, «map funksjon Arduino,» [Internett]. Available: <https://www.arduino.cc/reference/en/language/functions/math/map/>. [Funnet 11 05 2020].
- [27] Bombardier Recreational Products Inc, «Outlander Max 650 operator guide,» [Internett]. Available: http://www.operatorsguides.brp.com/OperatorsGuidesAttachments/OwnersManuals_SA/attach/03/2018/518910EN.pdf. [Funnet 9 april 2020].
- [28] K. Kanjanawanishkul, R. Phoohuengkaeo og A. Kumson, «Development of an automated guided vehicle with omnidirectional mobility for transportation of lightweight loads,» *2015 2nd International Conference on Mechatronics and Mechanical Engineering (ICMME 2015)*, nr. 2, p. 4, 2015.
- [29] J. Liu, P. Jayakumar, J. L. Stein og T. Ersal, «Combined Speed and Steering Control in High-Speed Autonomous Ground Vehicles for Obstacle Avoidance Using Model

Predictive Control,» *IEEE Transactions on Vehicular Technology*, vol. 66, nr. 10, pp. 8746-8763, 2017.

- [30] Festo SE & Co. KG, «Festo FHPP Description,» [Internett]. Available: https://www.festo.com/net/SupportPortal/Files/403824/CMM_-FHPP_2010-06a_555696g1.pdf?fbclid=IwAR3r9MhwDThCnYjQ3o32aEoV15l4tg1Wif-uyG7n6wkhneXMWO9X4zrNLiM. [Funnet 18 05 2020].
- [31] Festo, «Festo CANopen CiA402,» 2020. [Internett]. Available: https://www.festo.com/net/SupportPortal/Files/403851/CMMS_D-C-CO_2014-04_8034536g1.pdf.
- [32] Maxim Integrated, «MAX232 datasheet,» [Internett]. Available: <https://www.maximintegrated.com/en/products/interface/transceivers/MAX232.html>. [Funnet 24 04 2020].
- [33] Maxim Integrated, «Datasheet MAX485,» [Internett]. Available: <https://www.maximintegrated.com/en/products/interface/transceivers/MAX485.html>. [Funnet 24 04 2020].
- [34] Arduino, «Arduino Donate,» [Internett]. Available: <https://www.arduino.cc/en/Main/Donate>. [Funnet May 2020].
- [35] Festo SE & Co. KG, «Festo Configuration Tool,» [Internett]. Available: [https://www.festo.com/net/no_no/SupportPortal/Downloads/9840/9650/2020.02.13.1%20\[CMMS-ST%202.7.0.24\].Prereq.exe](https://www.festo.com/net/no_no/SupportPortal/Downloads/9840/9650/2020.02.13.1%20[CMMS-ST%202.7.0.24].Prereq.exe). [Funnet 01 05 2020].
- [36] RapidTables, «RapidTables converts Hex, binary and decimal,» [Internett]. Available: <https://www.rapidtables.com/convert/number/decimal-to-hex.html>. [Funnet 24 04 2020].
- [37] J. P. M. o. J. Cogswell, C++ all-in-one for dummies 3rd Edition, New Jersey: Jon Wiley & Sons, Inc., 2014.
- [38] ServoTronix, «ServoTronix CiA 402 beskrivelse,» [Internett]. Available: http://www.servotronix.com/html/stepIM_Help/HTML/home.htm?fbclid=IwAR111QF

DzwBnoTUR3ELH5kbkcRtqOhnLNkfyHXCir9gojcaj5_bqPjfkICI#!Documents/2011hwarningbits.htm. [Funnet 24 04 2020].

- [39] Servotronix Motion Control Ltd., «CANopen Devices profile segment, Controlword,» [Internett]. Available: http://www.servotronix.com/html/stepIM_Help/HTML/home.htm?fbclid=IwAR1_sXcOkwWi3J7LSr4xjEYVXEikV6D4TATgqRWQJrruPaEJ3qh4AscbYcs#!Documents/6040hcontrolword.htm. [Funnet 20 04 2020].
- [40] Servotronix, «Servotronix CiA402 6040h Control Word,» [Internett]. Available: http://www.servotronix.com/html/stepIM_Help/HTML/home.htm#!Documents/6040hcontrolword.htm. [Funnet 13 05 2020].
- [41] Elfa Distrelec, «Elfa Distrelec,» Elfa Distrelec, [Internett]. Available: <https://www.elfadistrelec.no/>.
- [42] Biltema, «Biltema,» Biltema, [Internett]. Available: <https://www.biltema.no/>.

Vedlegg

Vedlegg A – Oppgavebeskrivelse

Vedlegg B – Fremdriftsplan

Vedlegg C – Programkode Arduino

Vedlegg D – Brukerveiledning



PRH612 Bacheloroppgave

Studieretning: Informatikk og Automasjon

Prosjektgruppe: IA5-6-20

Tittel: Kommunikasjon mellom mikrokontroller og industriell motorkontroller på autonom 'All Terrain Vehicle': Lone Wolf

Veiledere: Saba Mylvaganam, Håkon Viumdal, Hans-Petter Halvorsen, USN;
Øivind Grønli, Kongsberg

Samarbeidspartner: Kongsberg Gruppen, Land systems

Prosjektets bakgrunn:

Oppgaven er basert på arbeidet gjort for Kongsberg Land Systems i en tidligere bachelor oppgave, «Guided Wolf», samt ett sommerprosjekt ved navn «Lone Wolf». Prosjektet består av en 'All Terrain Vehicle' (ATV) med ett autonomt styringsystem som erstatter føreren.

Systemet består av en Arduino Mega, med tilhørende signalbehandling og diverse aktuatorer, motorkontrollere (MK) og enkodere.

En påmontert stepper-motor med tilhørende enkoder, samt en FESTO MK tar seg av styringen av ATVen.

MKen er designet til å brukes med en PLS, men er i dette tilfellet styrt av ett kretskort tilknyttet en Arduino igjennom Inter-Integrated Circuit bussen (I²C).

Kretskortet består av en 24V til ±12V DC-DC omformer, kombinert med en 10-bit DA-omformer og en inverterende forsterker for å generere ett ± 10V analogt styre-signal. Tilbakekobling kommer fra enkoderen til stepper-motoren og har en utgang igjennom MKen. Utgangen er 0-10 V DC, men ettersom dette er for høy spenning for de analoge inngangene til Arduinoen, må signalet først igjennom en spenningsdeler. MKen er av typen FESTO CMMS-ST-C8-7-G2 med mulighet for parametrisering over RS-232, samt styring over RS-485, CANopen eller ProfibusDP.

Når man parametriserer MKen har man flere valg basert på oppsett av enkoder og stepper-motor, samt styringsmetode som blir brukt.

Velger man rett kombinasjon åpner muligheten seg for å bruke en 'Homing mode', samt 'Profile positioning mode', altså å sette settpunkt direkte i MKen slik at den innebygde regulatoren tar seg av resten.

Per dags dato må MKen kalibreres manuelt for hver oppstart, noe som gir en komplisert oppstarts-prosedyre og mindre nøyaktig styring.

Kalibreringen utføres ved å manuelt vri styret på ATVen slik at hjulene står i maksimalt utslag mot venstre, ihht. fører, for så å vri om tenningsnøkkelen som starter ATVen.

**Målbeskrivelse for prosjektet:**

Prosjektet skal igjennom bachelor oppgaven utvikle kode og hardware til Lone Wolf slik at den kan startes utelukkende med nøkkelen, deretter selv gjøre seg operativ og klar til autonom kjøring.

Arbeidet skal omfatte

- 1) Klargjøre Lone Wolf for bacheloroppgave.
 - o Isolere styringssystemet fra det autonome systemet.
- 2) Kartlegge de ulike signal alternativene mellom Arduino og FESTO MK.
- 3) Velge ny styremåte av MK.
- 4) Implementere 'Homing mode' og 'Profile positioning mode' fra Arduinoen.
- 5) Teste styresystemets virkemåte etter de foretatte endringene.
- 6) Implementere det autonome systemet med de nye endringene.
- 7) Kartlegge systemet rundt styring og kontroll av ATVen.
- 8) Leverer en Bachelor oppgave i henhold til retningslinjer satt av USN for faget PRH612-1 20V.

Avgrensning

Bacheloroppgaven omhandler primært styringssystemet innenfor hardware og software. Andre systemer som bremse- og pådragssystem er ikke beskrevet i oppgaven, men informasjon står i «Lone Wolf 2019» dokumentasjonen. Prosjektet skal ikke ta for seg fullstendig implementering av 'Festo Handling Program Protocol' (FHPP).

Delmålsbeskrivelse for prosjektet:


- 1) Påse at dokumentasjon og ATV er klar for årets sommer prosjekt.
- 2) Forbedre strømtilførsel for styringssystem.
- 3) Fikse kontrollboks for autonomt styringssystem.
- 4) Hente 'speedsensor' signal.


Praktiske tilrettelegging:


ATV samt diverse software- og hardwaremoduler leveres av Kongsberg Land Systems.

Bachelor Oppgave i sin helhet gjennomføres på USN, campus Porsgrunn. Minst ett møte med oppdragsgiveren og USN-veiledere skal gjennomføres. Microsoft Teams vil bli brukt for å forenkle daglig samarbeid om dokumenter, informasjonsdeling og diskusjoner.

Signatur



Saba Mylvaganam, veileder


Håkon Viumdal, veileder

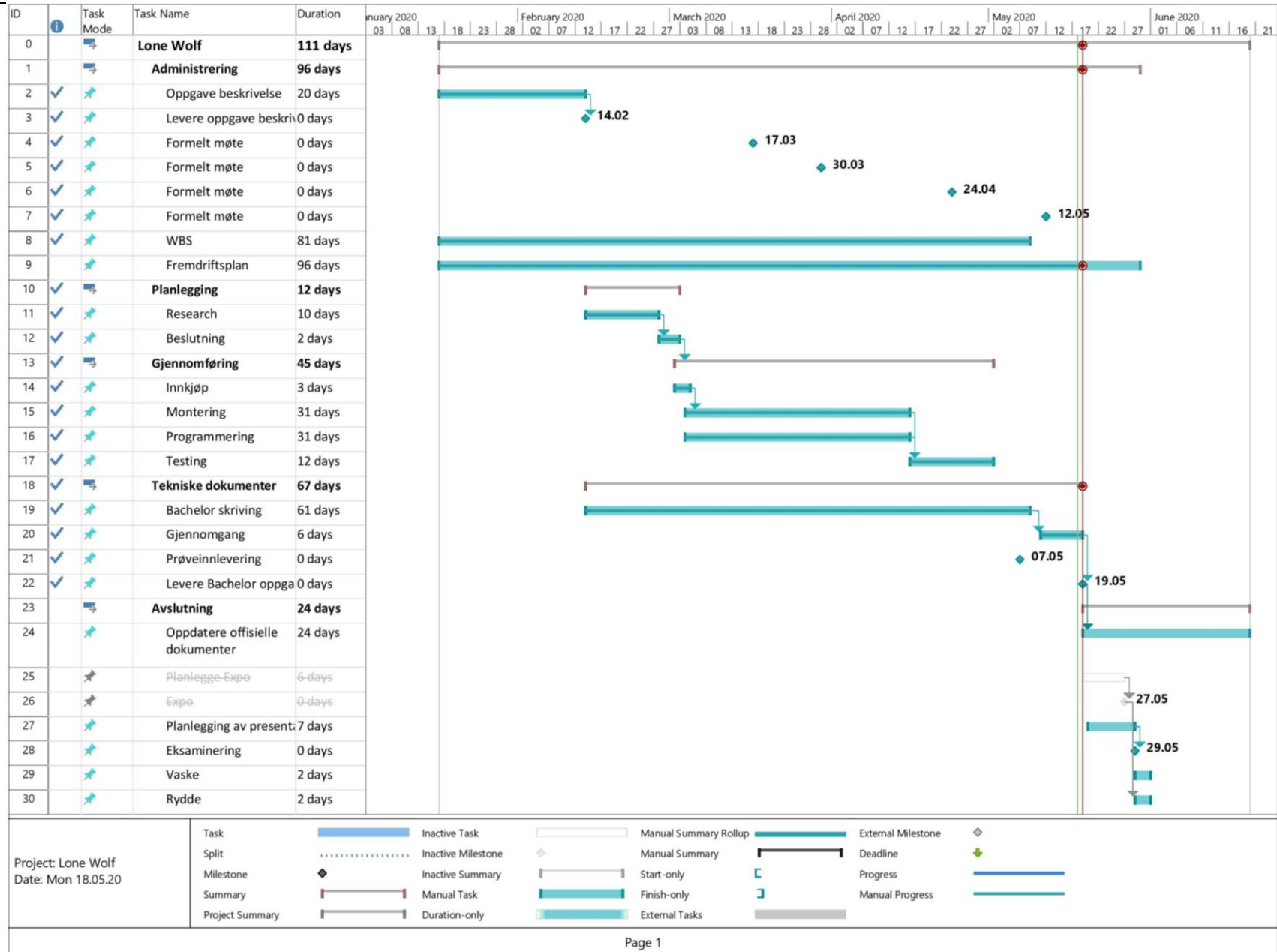

Hans-Petter Halvorsen, veileder

Dato: 14.02.2020


Johan Nicolai Barnholt, student


Sindre Bøe, student

Vedlegg B - Fremdriftsplan



Programmkode Arduino

```

int RCPin2 = 8; //Velg riktig pin for inngang
int controlEnable = 7; //Pin for Control Enable relè
int outputStageEnable = 6; //Pin for Outputstage Enable relè
int mkStop = 5; //Pin for Motorkontroller stop
                    relè

int RCPWM; //RC PWM verdi før map()
long setPoint; //RC verdi etter map() med
                    constrain

long lastAngle = 0; //Verdi for sist kjørte settpunkt
String command = String(0x000000); //command blir til "order". Verdi
                    i hexadesimal

String data = String(0x0000); //data blir til "order". verdi
i
                    hexadesimal

String leadZeroes = ""; //Lagring i antall leading zeroes
String order = "";

void setup() {
  Serial.begin(9600); //Setter baudrate for Serial0 til
                    9600

  Serial1.begin(9600); //Setter baudrate for Serial1 til
                    9600

  pinMode(RCPin2, INPUT); //Setter pin for "RCPin2" som en
                    input

  pinMode(controlEnable, OUTPUT); //Setter pin for "controlEnable"
                    som en output

  pinMode(outputStageEnable, OUTPUT); //Setter pin for
                    "outputStageEnable" som en
                    output

  pinMode(mkStop, OUTPUT); //Setter pin for "mkStop" som en
                    output

  delay(65); //delay for relèene blir aktiverte
  digitalWrite(controlEnable, HIGH); //Aktiviserer "controlEnable" relè
  digitalWrite(outputStageEnable, HIGH); //Aktiviserer "outputStageEnable"
                    relè

  digitalWrite(mkStop, HIGH); //Aktiviserer "mkStop" relè
  Startup(); //Gjennomfører Startup() metoden

```

```

Homing(); //Gjennomfører Homing() metoden
SelectPositioning(); //Gjennomfører SelectPositioning()
                    metoden
}

void loop() {
  RCPWM = pulseIn(RCPin2, HIGH); //Sender verdi fra RC
  setPoint = constrain(map(RCPWM, 1761, 1192, 0, 90), 0, 90); //fordeler
                    verdi til et 0-90 styresig-
                    nal med constrain

  Serial.print(setPoint + String(" --- ")); //Skriver "setPoint" ut på Se-
  rial0
                    slik at den kan leses på
                    skjermen

  SetPosition(setPoint); //Sender over "setPoint" til
                    SetPosition() metoden
}

void Transmit(){ //Metode for sending av ordre
  til
                    Motorkontrolleren

  order = String('=') + command + String(":") + leadZeroes + data; //Setter
                    sammen "order" til riktig
                    verdi.

  /* For endring av protokoll til RS-485 legges XT00: forran "=" i
  "String('=')". Det er også nødvendig med endring i kabling */

  Serial1.println(order); //Sender "order" til
                    motorkontroller

  Serial.println(order); //Skriver ut "order" på skjermen
  i
                    heksadesimal verdi
}

/* For kommende metoder settes "command", "data" og "leadingZeroes" før de
settes sammen i Transmitt() metoden. Valg av riktig kode står i Festo sine
manualer*/

void Startup(){ //Gjennomfører oppstart av
                    motorkontrolleren

  command = String(0x651010, HEX);
  data = String(0x0002, HEX);
  leadZeroes = "000";
  Transmit(); //Sender "enable logic" ordre

```

```

command = String(0x604000, HEX);
data = String(0x0006, HEX);
leadZeroes = "000";
Transmit(); //Sender "shutdown" ordre
data = String(0x0007, HEX);
leadZeroes = "000";
Transmit(); //Sender "Switch on/Disable
              operation" ordre
data = String(0x000F, HEX);
leadZeroes = "000";
Transmit(); //Sender "Enable operation/ go to
              ready state" ordre
}

void Homing(){ //Gjennomfører "Homing" sekvensen
               //til motorkontrolleren

command = String(0x606000, HEX);
data = String(0x06, HEX);
leadZeroes = "0";
Transmit(); //Sender "go to Homing mode" ordre
command = String(0x604000, HEX);
data = String(0x001F, HEX);
leadZeroes = "00";
Transmit(); //Sender "start homing" ordre
delay(20000); //Venter på at "Homing"
               //gjennomføres. 20 sekunder
               //er tiden den maksimalt bruker på å nå grensebryter

data = String(0x000F, HEX);
leadZeroes = "000";
Transmit(); //Sender "go to ready" ordre
}

void SelectPositioning(){ //Metode for sette
                          //motorkontrolleren i "absolute mode"

command = String(0x606000, HEX);
data = String(0x01, HEX);
leadZeroes = "0";
Transmit(); //Sender "set position to
              absolute" ordre

```

```

}

void SetPosition(long angle){                                     //Metode for å sette ordren for
                                                                styringen. Den har inng-
                                                                angsparameter som er RCens
                                                                styresignal.

    command = String(0x604000, HEX);
    data = String(0x000F, HEX);
    leadZeroes = "000";
    Transmit();                                                //Sender "go to ready state" ordre
    int differenceAngle = abs(lastAngle - angle); //lager en absolutverdi for
                                                                differansen fra siste sty-
                                                                reverdi og nåværende

    int deviation = 4;                                         //Filterinstilling for minimum
                                                                differanse som skal tillat-
                                                                tes for endring. Høyere tall
                                                                gir dårligere oppløsning,
                                                                men mindre preget av støy.

    if(differenceAngle < deviation ){angle = lastAngle;} //Dersom differansen
    i
                                                                styrevinkel er lik eller
                                                                mindre enn "deviation"
                                                                verdi settes nåverdi til
                                                                samme som forrige verdi og
                                                                styringen vil forbli samme
                                                                posisjon. Hvis større blir
                                                                styrevinkelen til ny ordre

    long mult = 1638;                                         //147456/90=1638.
                                                                Multiplikasjonsfaktor for
                                                                riktig styresignal. Desi-
                                                                malverdi av 90 grader på mo-
                                                                tor etter gir.

    long hexSteering = mult*angle;                             //Multipliserer med ønsket
                                                                styrevinkel

    lastAngle = angle;                                        //Siste styreverdi settes til
                                                                nåværende styreverdi for
                                                                bruk i neste oppkall.

    command = String(0x607A00, HEX);
    data = String(hexSteering, HEX);
    leadZeroes = "";
    if(data.length() < 8){                                     //Her legges manglende antall
                                                                leading zeroes inn i forkant
                                                                av data, til data består av
                                                                8-bit.

        int j = 8- data.length();
        for(int i = 0; i < j; i++){data = '0' + data;}}
    Transmit();                                                //Sender Styrevinkel ordre til

```

```
motorkontrolleren

command = String(0x604000, HEX);
data = String(0x009F, HEX);

//*** OBS OBS. Denne verdien
//avviker fra dokumentasjon.
//endring av bit 7 til høy re-
//seter feilkoder i motorkon-
//trolleren som tillater be-
//vegelse ***

leadZeroes = "00";
Transmit(); //Sender "Go to Steering SP" ordre
```

Brukerveiledning

Påse at batteriet er riktig montert og sikret bak på testtriggen. Deretter kan man skru på 24 V omformeren som er montert rett over batteriet. Skru deretter på fjernkontrollen.

Beveg deg så til fremsiden av testtriggen og sett på bryteren i øvre posisjon. Testtriggen vil nå kjøre «Homing» prosedyren. Denne vil vare i 20 sekunder før tannhjulet så beveger seg til midtre posisjon.

Testtriggen er nå klar til bruk og kan styres med venstre stikke i x-planet.

Feilsøking

Med flere systemer som kommuniserer sammen samt flere spenningsnivåer kan det ofte være vanskelig å finne feil på testtriggen. Det finnes flere verktøyer som kan hjelpe med feilsøking, der de viktigste er multimeter, USB til RS-232 adapter og FCT.

Spennings problemer

Det er tre forskjellige spenningsnivåer i testtriggen: 5 V, 12 V og 24 V.

5 V kommer fra omformeren på Arduinoen som tilføres 12 V fra batteriet.

Dersom en av kommunikasjons ICene eller reléene ikke skulle fungere vil 5 V spenningen være en mulig grunn til dette.

12 V blir tilført fra bilbatteriet bak på testtriggen. Den første tingen som kan hindre spenningen å komme frem er 15 A sikringen på den positive polen. Fra sikringen går spenningen direkte til 24 V omformeren og til Arduinoen. Om spenningen på batteriet synker under 12 V vil 24 V omformeren gi en alarm.

24 V spenningen kommer fra DC-DC omformeren montert over batteriet. Denne må slås på med bryteren på høyre side(mot motor). Om det ikke skulle være 24 V vil dette hovedsakelig gå utover MKen og stepper motoren, men også reléene på kretskortet. Mulige årsaker for dette kan være sikringen på 12 V tilførselen og for lavt spennings nivå på batteriet.

Kommunikasjon

RS-232 er hoved protokollen som brukes mellom Arduinoen og MKen. Denne baserer seg på at TTL seriell signalene fra Arduinoen når frem til MAX232 ICen. Deretter endrer spenningsnivået til ± 10 V og sender det videre til MKen.

For å måle om ICen har riktig spenningsnivå kan man se på databladet til MAX232 ICen her: [31]. Der vil man kunne se at «pin 2» på ICen skal ha 10 V og «pin 6» skal ha -10 V. Om disse spenningene stemmer vil ICen fungere dersom resten er i orden. Neste steg er å sjekke at kommunikasjonen beveger seg i riktig retning igjennom ICen. Man kan igjen se på databladet til ICen [31] og se at den har to kanaler i hver retning. «Pin 10» eller «Pin 11» tar imot TTL seriell signalet fra Arduinoen og sender det videre ut på «Pin 7» eller «Pin 14». Det vil si at TX utgangen på Arduinoen må kobles til «Pin 10» eller «Pin 11». RX inngangen på D-sub 9 kontakten («Pin 2») må kobles til «Pin 7» eller «Pin 14».

På samme måte må man påse at kommunikasjonen kommer tilbake i riktig retning.

«Pin 8» eller «Pin 13» tar imot RS-232 signaler og sender de videre ut på «Pin 9» eller «Pin 12». Derfor må TX utgangen på D-sub 9 kontakten kobles til «pin 8» eller «Pin 13», og RX inngangen på Arduinoen må kobles til «Pin 9» eller «Pin 12».



Om dette skulle være koblet rett, men kommunikasjonen fremdeles ikke funker, kan man feilsøke med USB til RS-232 adapteren. Koble utgangen av ICen til adapteren, og adapteren til PCen. Du kan nå bruke Terra Term med riktig baud rate (samme som er satt opp i Arduino utgangen) for å lese av alt som sendes fra Arduinoen i klar tekst.

Avlesning av alarmer på motorkontrolleren

Det finnes flere alarmer fra MKen, en liste over alarmer finnes i kapittel 8.2 [19].

Tabell 7-1 Syv segment Alarmer, hentet fra Festo dokumentet, beskriver hvordan alarmer kan leses av fra MKen.

Tabell 7-1 Syv segment Alarmer [3]

Error/warning messages		
	E x x y	Error (E = error) Number: Two-position main index (x x), single-position sub-index (y) Example: E 0 1 0 → appendix A.
	- x x y -	Warning Number: Two-position main index (x x), single-position subindex (y). Example: - 1 7 0 - → appendix A.

1) Several characters are displayed one after the other.

En annen måte å lese av alarmer på er å koble PC-en til MKen igjennom USB til RS-232 adapteren. Om man så starter FCT og går «online» er det mulig å lese av alarmer i klartekst, samt modifisere forriglingsmatrisen til MKen som beskrevet i kapittel 3.3.8.

C++ kode

Arduino IDEen har ikke innebygde verktøyer for simulering av koden. En av de beste måtene å teste koden på er å legge inn «Serial.print()» metoden og skrive ut verdien man feil-søker til terminalen.

