

SYDDANSK UNIVERSITET

SEMESTERPROJEKT I KONTROL OG REGULERING AF ROBOTSYSTEMER, (F19) -
RB-PRO4

CIV. ROBOTTEKNOLOGI 4. SEMESTER - GRUPPE 1

DEADLINE FOR AFLEVERING: 29. MAJ 2019

Styring af pan-tilt system

Studerende:

Andreas Risskov Sørensen
asoer09@student.sdu.dk
15/03-1988
Eks.nr.: 456885

Studerende:

Dan Nykjær Jakobsen
dajak17@student.sdu.dk
30/04-1998
Eks.nr.: 455913

Studerende:

Joakim Lykke Stein
joste17@student.sdu.dk
14/08-1997
Eks.nr.: 456129

Studerende:

Lars Gylding Lindved
lalin17@student.sdu.dk
09/12-1987
Eks.nr.: 443299

Studerende:

Mirzet Traljesic
mtral17@student.sdu.dk
29/05-1996
Eks.nr.: 113419323

Studerende:

Troels Zink Kristensen
tkris17@student.sdu.dk
30/04-1997
Eks.nr.: 455427

Vejleder:

Christoffer Sloth
chsl@mmtmi.sdu.dk

Abstract

The purpose of this project is to control and regulate a pan-tilt system. It has to be designed and implemented with a microcontroller and an FPGA to control two DC-motors through a double H-bridge using pulse width modulation (PWM).

The project looks at controlling and regulating a modified Labyrinth-game, which can be controlled with two degrees of freedom equivalent to the original game.

The project is divided into four major topics: The application for the Labyrinth-game, the SPI-protocol for communication between the microcontroller and the FPGA, the communication protocol which defines certain messages to be sent between the devices, and the closed-loop system that regulates the pan-tilt system.

The application is used to compensate for the lack of user feedback in the original Labyrinth-game. The actual modified Labyrinth-map consists of nine photoelectric sensors, which are used to signal when the game is lost or won. A timer is started and shown on an LCD-display when the game starts, alongside the current high-score.

The application cannot operate without communication. The SPI-protocol uses a four-wire bus to communicate between a master and a slave; the microcontroller being the master and the FPGA being the slave. The master controls the beginning of the communication and a clock, and the slave obeys the master.

It is the communication protocol's responsibility to use the SPI-protocol to communicate using certain messages, which can set values from a joystick to the regulator. It can also get values from encoders and hall sensors on the pan-tilt system with a reply message, so the regulator can control the movement of the game.

A mathematical model of the pan-tilt system is determined. This mathematical model is used to create a closed-loop model of the system with a Proportional-Derivative (PD) controller. The PD-controller can control and regulate the position for the Labyrinth-game.

The project concludes with the Labyrinth-game controlled by a joystick and regulated by a PD-controller to achieve low sensitivity and fast response.

Indholdsfortegnelse

1 Forord	1
2 Indledning	2
3 Projektbeskrivelse	3
4 Applikation	4
5 Elektronik til spillet	7
6 SPI-protokol	10
6.1 Master	11
6.2 Slave	13
7 Kommunikationsprotokol	16
7.1 Protokol	16
7.2 Implementering på mikrocontroller	17
7.3 Implementering på FPGA	19
8 Motorstyring	21
8.1 PWM	21
8.2 Hall-sensorer og enkodere	22
9 Matematisk modellering af systemet	23
9.1 Det elektriske system	23
9.2 Det mekaniske system	24
9.3 Tilstandsmodel	25
9.4 Motorkonstanter	25
9.5 Systemets inertiomomenter	28
10 Regulering af systemet	31
10.1 Krav	31
10.2 Klargøring af den matematiske model	31
10.3 Design af regulator	33
10.4 Simulering af systemet	38
10.5 Implementering	41
10.6 Test af systemet	43
11 Konklusion	45
12 Perspektivering	46
13 Litteraturliste	47
14 Oversigt over elektronisk bilag	47

1 Forord

Rapporten er udarbejdet af Andreas Risskov Sørensen, Dan Nykjær Jakobsen, Joakim Lykke Stein, Lars Gylding Lindved, Mirzet Traljesic og Troels Zink Kristensen.

Projektet (RB-PRO4) tilhører uddannelsen, Civilingeniør i Robotteknologi, på 4. semester 2019 v. Syddansk Universitet i Odense. Projektet dækker 10 ECTS point, hvor de to kurser, RB-EMD4 og RB-ELR4, anvendes som supplement til at udføre projektet.

Projektet blev udleveret d. 5. februar 2019 og afsluttes med aflevering af rapport senest d. 29. maj 2019. Der er givet en arbejdsperiode på 16 uger og 1 dag.

Vi vil gerne takke vores vejleder, Christoffer Sloth, for at have vejledt os.

Læsevejledning

Der skal gøres opmærksom på, at kilder refereres i rapporten med tal-eksponenter placeret i firkantede parenteser, således: ^[2]. Disse referencer kan ses under sektionen: *Litteraturliste*.

I det elektroniske bilag er jurnaler, diagrammer, kode mm. tilgængelige. Oversigten over det elektroniske bilag ses under sektion 14: *Oversigt over elektronisk bilag*.

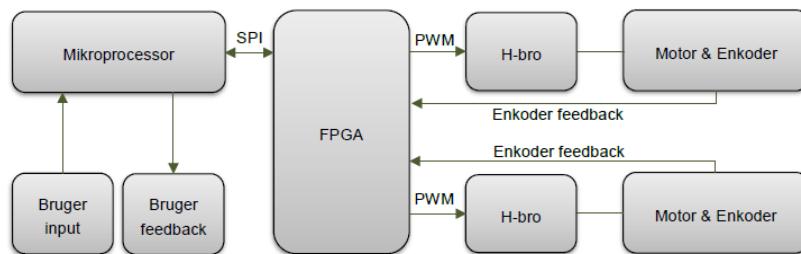
2 Indledning

Projektet omhandler regulering og kontrol af robotsystemer, som har til formål at styre et dynamisk system. Et pan-tilt system skal designes og regulering skal implementeres, således at styring af systemet kan foregå vha. én eller flere brugerinputs.

De essentielle krav til systemet er som følgende:

- Regulatorerne skal implementeres på en mikrocontroller.
- Der skal benyttes SPI-kommunikation mellem en mikrocontroller og en FPGA.
- FPGA'en skal styre Pulse-Width Modulation (PWM) signaler til motorerne.
- FPGA'en skal benyttes til at bestemme motorernes position via enkoderne.

Figur 1 viser blokdiagrammet over pan-tilt systemet:



Figur 1: Blokdiagram af pan-tilt systemet.^[1]

Projektet omhandler i stor grad kommunikation via SPI, PWM-signaler til at styre motorerne, og regulatorer til at regulere systemet.

Disse projektdele anvendes i en modificering af det klassiske labyrintspil, hvor de to frihedsgrader opnås ved ombygning af pan-tilt systemet. Figur 2 viser det konstruerede spil.



1. Gaffel af pan-tilt systemet.
2. Ramme af pan-tilt systemet.
3. DC-motor til gaflen.
4. DC-motor til rammen.
5. Hjemmelavet joystick til styring.
6. Banen til labyrintspillet.
Indeholder otte fejhuller, som kuglen skal undgå, og ét målhul, som indikerer sejr.

Figur 2: Det modificerede labyrintspil på pan-tilt systemet.

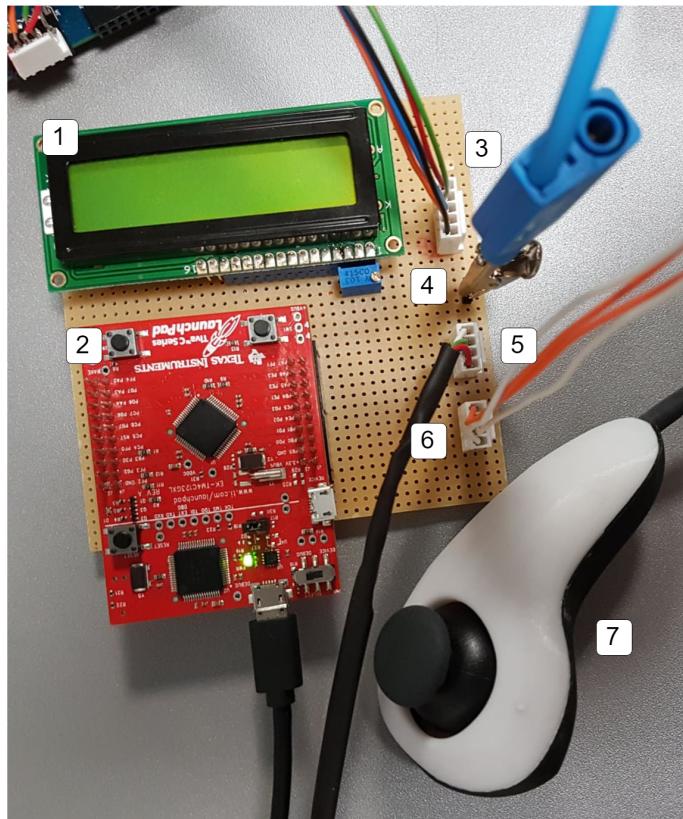
3 Projektbeskrivelse

Projektet omhandler konstruktion af det klassiske labyrintspil med egne modifikationer. Det udleverede pan-tilt system modificeres således, at systemet opnår samme to frihedsgrader som det klassiske spil. Systemet reguleres af en mikrocontroller, som styrer en FPGA, der er forbundet til de to motorer. Der skal konstrueres et joystick, som kontrollerer spillets bevægelse. Herpå realiseres brugerfeedback i form af informationer på et display.

Projektets fokusområder udformes herunder:

- *Det klassiske labyrintspil skal realiseres vha. pan-tilt systemet*
 - Banens orientering skal styres vha. brugerinput. Det skal være muligt at registrere om spillet er gennemført, eller om kuglen er faldet i et hul, hvilket meddeles til brugeren.
- *Kommunikation mellem systemets enheder*
 - Kommunikationen mellem mikrocontroller og FPGA skal foregå gennem SPI-protokollen. Herudover skal der oprettes en grænseflade til mikrocontrollerernas håndtering af joysticket.
- *Motorstyring*
 - FPGA'en skal danne PWM-signaler, der regulerer motorernes hastigheder og omløbsretninger gennem H-broen. Motorernes position bestemmes på FPGA'en vha. enkoderne.
- *Pan-tilt systemet skal reguleres*
 - Systemet skal modelleres, og denne model skal anvendes til at designe og simulere forskellige reguleringssløjfer til styring af systemet. Den mest optimale implementeres på mikrocontrolleren.

4 Applikation



Figur 3: PCB med forbindelser.

- | | | |
|----------------|-----------------------------|-----------------------------|
| 1: LCD-skærm | 2: Mikrocontroller | 3: Forbindelse til FPGA |
| 4: Fælles stel | 5: Forbindelse til joystick | 6: Forbindelse til sensorer |
| 7: Joystick | | |

Implementeringen af selve spillet består af et joystick til at styre systemet, en timer til at tage tid på en spillerunde, ni sensorer til at registrere om kuglen falder i et hul, samt en applikation til at styre logikken. Desuden er der konstrueret et Printed Circuit Board (PCB), hvorpå mikrocontrolleren forbindes til en LCD-skærm, sensorer, joystick og FPGA'en (se figur 3).

RTCS

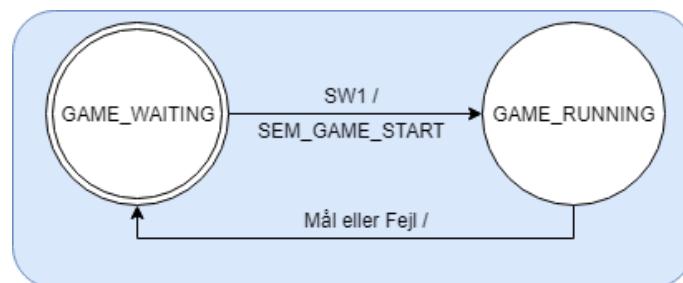
Programmet til mikrocontrolleren er programmeret i C. Programmet er designet og implementeret vha. RTCS (Run To Complete Scheduler) som er givet af Morten Hansen fra kurset RB-EMD4. RTCS anvender en scheduleringsmodel, hvor hver proces (her task) udføres indtil den er færdig, medmindre en task selv giver kontrollen tilbage til scheduleren.^[2]

I kurset RB-EMD4 er der, foruden RTCS, også blevet stiftet bekendtskab med FreeRTOS, som er en scheduleringsmetode med preemptive scheduling. Denne scheduleringsmetode er dog mere indviklet at forstå og implementere, hvilket var grunden til valg af RTCS. Ydermere er programmet til mikrocontrolleren ikke så stort eller kompliceret til, at RTCS ikke er tilstrækkelig.

RTCS kører med non-preemptive scheduling. Preemptive scheduling betyder, at tasks får tildelt CPU-tid i et specifikt tidsrum, og derved bliver afbrudt, hvis de ikke er færdiggjorte inden for det tidsrum. Herudover kan en proces med højere prioritet end den nuværende komme foran i køen. I dette tilfælde med non-preemptive scheduling får en proces tildelt CPU-tid, indtil processen er udført. Det vil altså sige, at andre processer ikke kan afbryde en igangværende proces.^[3]

Modul game.c

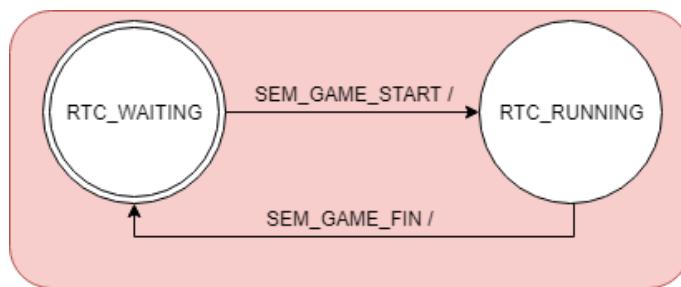
Spilapplikationen består af en `game_task` indeholdende en tilstandsmaskine med to tilstande: `GAME_WAITING` og `GAME_RUNNING`, hvor startes i førstnævnte. Hvis der registreres et tryk på knappen SW1 på mikrocontrolleren, nulstilles det elektriske kredsløb for kugledektering, og der signaleres en semafor til `rtc`-modulet om, at spillet er igangsat, og timeren skal startes. Derefter fortsættes til `GAME_RUNNING`. Her tjekkes det om kuglen falder i et af fejlhullerne eller målhullet. Er dette tilfældet signaleres en semafor til `rtc`-modulet om at spillet er slut, og der returneres herefter til `GAME_WAITING`. Var det målhullet, kuglen faldt i, bliver tiden desuden sammenlignet med den aktuelle rekordtid, og er den lavere, bliver dette den nye rekordtid.



Figur 4: Statediagram for game-task'en.

Modul rtc.c

rtc_task står for timeren, der tager tid på en spillerunde. Den indeholder også en tilstandsmaskine med to tilstande, RTC_WAITING og RTC_RUNNING. I første tilstand ventes på semaforen fra game_task om, at spillet er påbegyndt. Signaleres denne, nulstilles timeren, og den fortsættes til RTC_RUNNING. Her tælles timeren 10 ms op, og der indsættes en wait(2), som sørger for, at styresystemet venter fem ms to gange før task'en køres igen. Signaleres semaforen for, at spillet er afsluttet, returneres til tilstanden RTC_WAITING. Dette modul indeholder desuden en task, der viser den nuværende tid og rekordtid på LCD-skaermen, ved brug af lcd-, file- og string-modulerne anskaffet fra RB-EMD4-kurset.



Figur 5: Statediagram for rtc-task'en.

Modul joystick.c

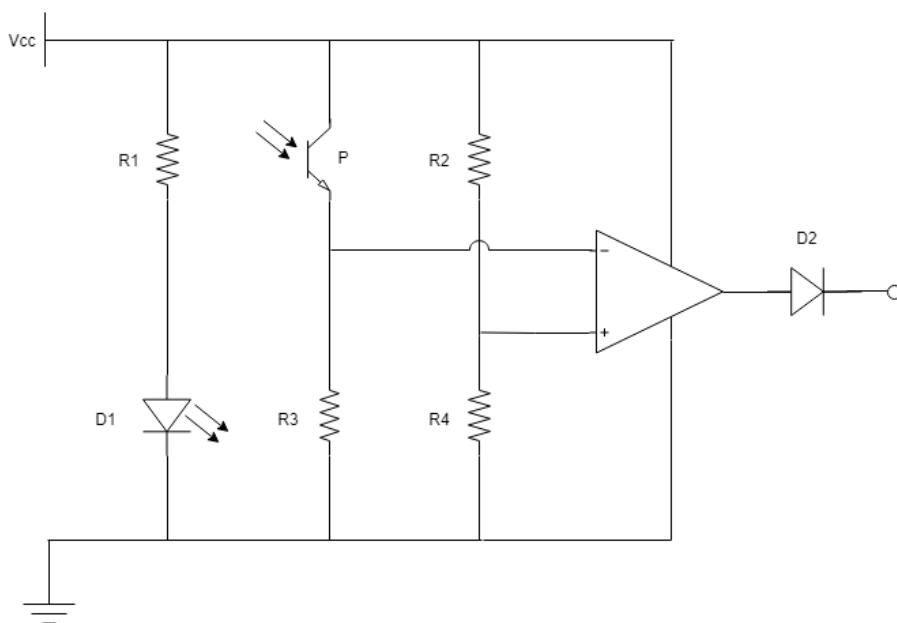
Joysticket har to akser, som hver består af et potentiometer. Værdierne fra disse aflæses af mikrocontrolleren ved hjælp af en Analog-Digital-konvertering (ADC) og omregnes til en vinkel. Den indbyggede ADC er 12-bit, hvilket giver en værdi mellem 0 og 4095. Da joystickets udgangspunkt er i midten, svarer den første halvdel af værdierne til en vinkel i den ene retning, mens den anden halvdel svarer til en vinkel i den anden retning.

5 Elektronik til spillet

Til detektering af kuglen blev to mulige løsninger overvejet.

Den ene mulighed var et kredsløb med en mekanisk kontakt. Når kuglen falder i et hul aktiveres denne, og man kan derved registrere hændelsen. En sådan løsning kræver en relativt tung kugle for at aktivere kontakten, og der skal desuden tages højde for, at en mekanisk kontakt kan resultere i prel.

Grundet disse problematikker fravælges ovenstående løsning, og i stedet konstrueres et kredsløb bestående af en infrarød-diode (IR), en fototransistor, og en rail-to-rail operationsforstærker anvendt som komparator (se figur 6).



Figur 6: Sensor-kredsløb.

For at IR-dioden lyser tilstrækkeligt, kræves en spænding over den på 1,8 V. Ved denne spænding løber der 75,5 mA gennem IR-dioden, hvilket giver følgende indre resistans, R_{D1} :

$$R_{D1} = \frac{V_{D1}}{I_{D1}} = \frac{1,8 \text{ V}}{75,5 \text{ mA}} = 23,8 \Omega$$

Grundet V_{cc} på 6 V, kræves et forhold mellem $R1$ og R_{D1} i nærheden af 2:1 for at opfylde spændingskravet for IR-dioden. Ud fra forholdet bestemmes $R1$ til at have en resistans på 50 Ω .

Ved den ikke-inverterende indgang på komparatoren er spændingsandelen over $R4$ 3 V, grundet spændingsdelingen mellem $R4$ og $R2$, der er ens.

Ved IR-belysning af fototransistoren, vil dennes indre resistans, R_P , være ca. 500 Ω , og uden belysning vil den være over 40 k Ω . $R3$ vælges da til at have en størrelse på 1100 Ω , så

spændingen over $R3$, V_{R3} , samt over den inverterende indgang på operationsforstærkeren, vil være:

$$V_{R3} = V_{CC} \cdot \frac{R3}{R3 + R_P} = 6 \text{ V} \cdot \frac{1100 \Omega}{1100 \Omega + 500 \Omega} = 4 \text{ V}$$

under belysning.

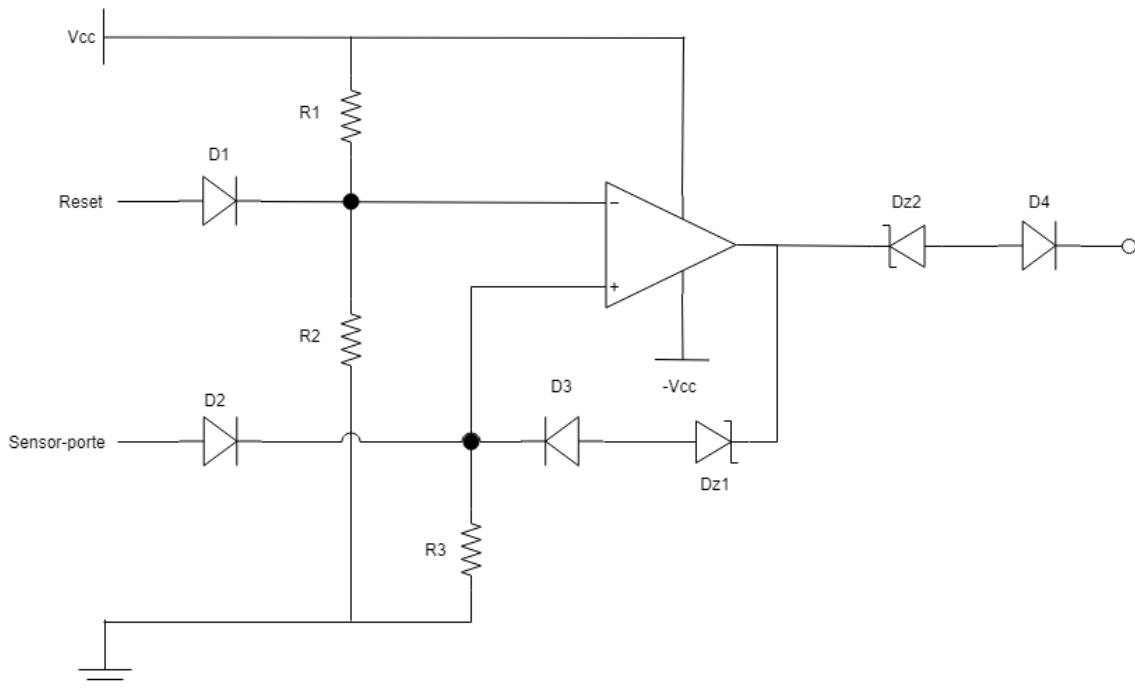
Brud af lys resulterer i 0,16 V over $R3$ på den inverterende indgang, da resistansen R_P nu er over 40 k Ω . Ud fra disse forudsætninger, vil spændingen over komparatorens udgang være nominelt V_{CC} når belysningen af fototransistoren brydes, og nominelt 0 V hvis ikke.

Tabel 1 viser de forskellige komponenter samt deres værdier.

Sensor-kredsløb				
Vcc [V]	R1 [Ω]	R2 [Ω]	R3 [Ω]	R4 [Ω]
6	50	1000	1100	1000
D1 OP298 IR-diode Ved 1,8 V: $R_{D1} = 23,8 \Omega$	D2 1N4007	P OP593 NPN-fototransistor Ved belysning: $R_P = 500 \Omega$ Uden belysning: $R_P > 40 \text{ k}\Omega$		Op-Amp 8031A

Tabel 1: Komponenter i sensor-kredsløbet.

Falder kuglen i målhullet eller en af fejlhullerne, ønskes det at signalet vedligeholdes, for at sikre, at dette registreres. Til det formål konstrueres en latch. Latchen består af en operationsforstærker, dioder, samt zenerdioder. Denne kan ses på figur 7:



Figur 7: Latch-kredsløb.

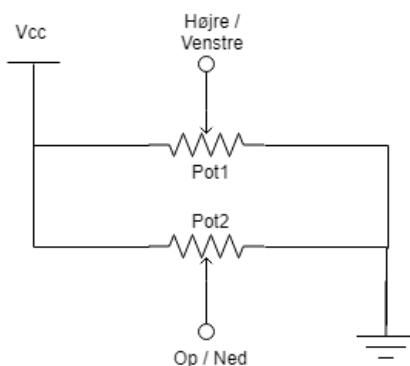
Den inverterende indgang på operatinsforstærkeren er forbundet til en spændingsdeling mellem $R1$ og $R2$, hvor spændingen over $R2$ er 2 V. Indgangen er desuden forbundet til *Reset* gennem en diode. *Reset* er en forbindelse til mikrocontrolleren, som anvendes til at nulstille latchen. Mikrocontrolleren vil påtrykke en spænding på 3,3 V ved *Reset*, hvilket ændrer spændingen på den inverterende indgang til 2,6 V. Hvis kuglen falder i et af hullerne, vil der påtrykkes en spænding ved den ikke-inverterende indgang på 5,3 V fra *Sensor-porten*. Dette vil resultere i en udgangsspænding for operationsforstærkeren på 6 V. Grundet den positive feedback, vil der efter zenerdioden $Dz1$ og dioden $D3$ ligge 2,3 V, ved den ikke-inverterende indgang. Dette resulterer i en positiv differens mellem operationsforstærkerens indgange, og signalet vedligeholdes indtil *Reset*-signalet sendes, og differensen mellem indgangene bliver negativ.

Latchen skal forbindes til mikrocontrolleren. Det giver begrænsninger for maksimum- og minimumspændinger der kan påtrykkes portene. Derfor tilføjes $Dz2$, så maksimum- og minimumspændingerne er henholdsvis 4,2 V og -4,2 V. Portene på mikrocontrolleren kan ikke håndtere at påtrykkes mindre end -0,5 V; derfor ensrettes strømmen med dioden $D4$.

Latch-kredsløb				
Vcc [V]	R1 [Ω]	R2 [Ω]	R3 [Ω]	D1-D4
6	2000	1000	10000	1N4007
Dz1 Zenerdiode 3 V	Dz2 Zenerdiode 1,8 V		Op-Amp μ A741	

Tabel 2: Komponenter i latch-kredsløbet.

Joysticket til at styre pan-tilt systemet, består af to potentiometere, hvilke forbindes til Vcc og stel. Spændingsdelingen for potentiometrene kan måles på deres respektive udgange. Diagrammet for joystick-kredsløbet kan ses på figur 8. Højre / Venstre og Op / Ned angiver, hvilken retning joysticket giver motorerne iif. pan-tilt systemet.



Joystick-kredsløb		
Vcc [V]	Pot1 [Ω]	Pot2 [Ω]
3,3	10000	10000

Tabel 3: Komponenter i joystick-kredsløbet.

Figur 8: Joystick-kredsløb.

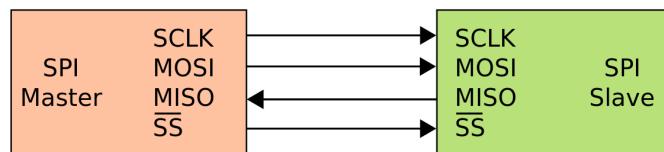
6 SPI-protokol

Applikationen kræver kontrol af motorerne på systemet for at spille Labyrint-spillet. Mikrocontrolleren styrer motorerne gennem en FPGA, hvilket kræver kommunikation mellem de to.

Kommunikationen mellem mikrocontroller og FPGA skal foregå vha. SPI-protokollen, jævnfør krav om dette i den udleverede projektbeskrivelse. Kommunikationen skal fungere tovejs, da der ifølge figur 1 både skal være bruger-input og bruger-feedback.

SPI er en forkortelse for *Serial Peripheral Interface*, hvilket indikerer, at kommunikationen fungerer serielt. Seriel kommunikation er bitvis dataoverførsel, i modsætning til parallel kommunikation, som er bytevis dataoverførsel. SPI benytter sig af fuld dupleks, hvilket indikerer tovejs-kommunikation og samtidig dataoverførsel for både afsender og modtager.

I denne protokol findes begrebet *master* og *slave*, hvor der altid kun er én master, men én til flere slaver. Masteren afgiver ordrer, som slaven/slaverne adlyder. Dette projekt indeholder en master i form af mikrocontrolleren og kun én slave i form af FPGA'en. Kommunikationen foregår således (se figur 9):



Figur 9: SPI-bus for masteren og den ene slave.^[4]

SCLK (kaldet SCK i koden) er *Serial CLCK*, MOSI er *Master Output, Slave Input*, MISO er *Master Input, Slave Output* og SS er *Slave Select*. Dvs. tre outputs og et input fra master til slave; det er altså en ”four-wire” seriellbus. Slaven kan ikke sende noget til masteren før slaven modtager noget fra masteren.

SCK angives af masteren, og sætter den clock-frekvens, hvormed transmissionen skal foregå. En fælles clock er nødvendig, da enhedernes egne clock-frekvenser kan være asynkrone og i forskellige hastigheder. SS angives ligeledes af masteren og betyder, at en transmission skal til at finde sted.^[5]

Master og slave skal opsættes på hver sin måde, både pga. deres roller, men også pga. forskellen på mikrocontroller og FPGA. Dette uddybes i de to følgende underafsnit.

6.1 Master

Mikrocontrolleren skal opsættes efter Texas Instruments' beskrivelse i deres datablad for Tiva TM4C123GH6PM Microcontroller. Kapitel 15 i databladet omhandler Synchronous Serial Interface (SSI), hvilket også kan opsættes som SPI.^[6]

Figur 10 viser opsætningen af SPI for master på mikrocontrolleren. Denne opsætning er opstillet ud fra en step-by-step guide fra databladet for mikrocontrolleren.

```

SYSCTL_RCGCSSI_R |= SYSCTL_RCGCSSI_R0;           //Enable SSI Module 0
SYSCTL_RCGGPIO_R |= SYSCTL_RCGGPIO_R0;         //Enable PORT A
GPIO_PORTA_AFSEL_R |= 0x3C;                      //Enable alternate function (SSI) for pins PA[5:2]
GPIO_PORTA_PCTL_R |= (GPIO_PCTL_PA5_SSI0TX |
                      GPIO_PCTL_PA4_SSI0RX |
                      GPIO_PCTL_PA3_SSI0FSS |
                      GPIO_PCTL_PA2_SSI0CLK); //Assign SSI-signals (PMCn) to pins PA[5:2] in GPIOPCTL
GPIO_PORTA_DEN_R |= 0x3C;                         //Enable SSI-pins PA[5:2]
GPIO_PORTA_DIR_R |= 0x2C;                          //Set PORTA data direction - PA5=transmit (out),
                                                //PA4=receive (in), PA3=enable (out), PA2=clk (out)
GPIO_PORTA_DR2R_R |= 0x3C;                        //Set PORTA pins to 2-mA drive
GPIO_PORTA_PDR_R |= 0x3C;                          //Set pull-down on PORTA pins PA[5:2] !!!PULL-UP VS PULL-DOWN
SSI0_CR1_R &= 0x0;                            //Set microcontroller as Master
SSI0_CC_R &= 0x0;                             //Base the clock-control on the system clock and prescaler:
SSI0_CPSR_R |= 0x02;                           //Configure clock prescaler: sysClk = 16 MHz - SClk = 2 MHz
SSI0_CRO_R |= 0x0300;                           //ClkPol = 1, ClkPhase = 1
SSI0_CRO_R |= SSI_CRO_SPH;                     //Set freescale SPI Frame format (Clear bits for safety)
SSI0_CRO_R &= 0xFFFFFFF;                      //Datasize = 16 bits
SSI0_CRO_R |= SSI_CRO_DSS_16;                  //Activate SSI
SSI0_CR1_R |= SSI_CR1_SSE;

```

Figur 10: Opsætning af SPI på mikrocontrolleren.

Det er blandt andet her, hvor SSI opsættes som SPI, men også hvilke fire porte, som skal anvendes til ”four-wire” kommunikation. Mikrocontrolleren sættes også til at være master. Herudover konfigureres serial-clock-frekvensen til at være 2 MHz vha. opsætningsregistre. Da mikrocontrolleren har en clock-frekvens på 16 MHz, vil den maksimale serial-clock-frekvens være 2,67 MHz. Det vurderes at 2 MHz er tilstrækkeligt, da der kan sendes over 100.000 beskeder i sekundet, med den maksimale datastørrelse på 16 bit.

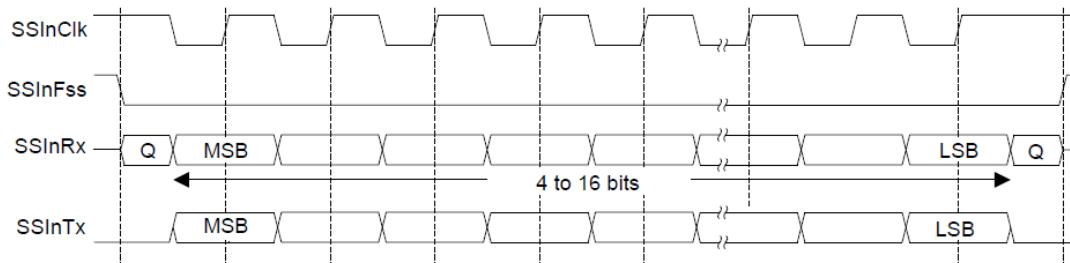
En vigtig opsætning er, hvordan dataoverførslen foregår ift. SS, hvilket bestemmes af to bit: Serial Clock Phase og -Pulse (hhv. SPH og SPO) på SSICR0-registret (SSI Control Register 0). Disse to bit kan opsættes således:

- SPH: SSI Serial Clock Phase
 - 0: Data læses på den første ændring i SCK.
 - 1: Data læses på den anden ændring i SCK.
- SPO: SSI Serial Clock Pulse
 - 0: SCK sættes til 0 når data ikke overføres.
 - 1: SCK sættes til 1 når data ikke overføres.

Dette giver fire forskellige muligheder for opsætning af dataoverførslen. Alle fire muligheder kan anvendes, men koden til FPGA'en, altså slaven, skal tilpasses den valgte opsætning. Derfor opsættes besked-formatet til at være:

- SPH = 1 og SPO = 1.

Denne opsætning ses herunder på figur 11:



Figur 11: Besked-format for SPI-protokollen med SPH = 1 og SPO = 1 set fra mikrocontrolleren.^{[6]s.960}

På den måde sikres det, at afsendelsen og modtagelsen først læses, når de to porte har fået en stabil værdi. SPI-protokollen er nu opsat, men for at kunne afsende og modtage til og fra FPGA'en, skal der opsættes to simple funktioner, som tager sig af dette. Disse to funktioner ses på figur 12:

```
void writeSPI(INT16U message)
{
    SSI0_DR_R        = message;
}

INT16U readSPI()
{
    INT16U input = 0;
    if (SSI0_SR_R & SSI_SR_RNE)
        input = SSI0_DR_R;

    return input;
}
```

Figur 12: De to funktioner som sørger for henholdsvis afsendelse og modtagelse af data.

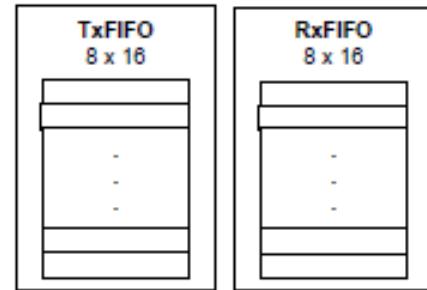
`writeSPI` har et 16 bit positivt heltal (INT16U) som parameter, som udsendes til `SSI0DR`-registret, hvilket er dataregistret. Denne `message` bliver sat ind på en buffer ud fra FIFO-princippet (First In, First Out).

`readSPI` er en INT16U-funktion, da den skal returnere en 16-bit størrelse til mikrocontrolleren. Den udføres en AND-operation på `SSI0DR`-registret og `SSISR`-registret (statusregister) på pladsen `RNE`. Dette tjekker om FIFO-bufferen indeholder elementer. Grunden til en datastørrelse på 16 bit forklares under sektion 7.

6.1.1 Buffere

Foruden `writeSPI` og `readSPI`, er der også skrevet tre funktioner, der tjekker statusregistret, til at holde styr på beskeder heri. I databladet for mikrocontrolleren kan størrelsen på de to buffer aflæses; buffer for afsendelse og buffer for modtagelse. Disse har størrelsen 8x16 bit, dvs. 8 beskeder med 16 bit i hver (se figur 13).

Herunder på figur 14 kan man se de tre funktioner: `tx_FIFO_isFull()`, `tx_FIFOIsEmpty()` og `rx_FIFOIsEmpty()`, som alle tre er boolske funktioner.



Figur 13: Henholdsvis buffer for afsendelse og buffer for modtagelse.^{[6]s.951}

```
BOOLEAN tx_FIFO_isFull()
{
    INT8U status = TRUE;
    if( SSI0_SR_R & SSI_SR_TNF )
    {
        status = FALSE;
    }
    return status;
}

BOOLEAN tx_FIFOIsEmpty()
{
    INT8U status = FALSE;
    if( SSI0_SR_R & SSI_SR_TFE )
    {
        status = TRUE;
    }
    return status;
}

BOOLEAN rx_FIFOIsEmpty()
{
    INT8U status = TRUE;
    if( SSI0_SR_R & SSI_SR_RNE )
    {
        status = FALSE;
    }
    return status;
}
```

Figur 14: De tre funktioner som tjekker statusregistret.

- `tx_FIFO_isFull()`: Som standard er `status = TRUE`. Hvis TNF-bitten er sat, vil `status = FALSE` returneres. Dette betyder, at Tx-bufferen *ikke* er fuld. Ellers returneres `status = TRUE`.
- `tx_FIFOIsEmpty()`: Som standard er `status = FALSE`. Hvis TFE-bitten er sat, vil `status = TRUE` returneres, hvilket betyder, at Tx-bufferen er tom. Hvis ikke, returneres `status = FALSE`.
- `rx_FIFOIsEmpty()`: Som standard er `status = TRUE`, hvilket er anderledes ift. `tx_FIFO_Empty()` pga. logikken i databladet. Hvis RNE-bitten er sat, vil `status = FALSE`, hvilket indikerer, at Rx-bufferen *ikke* er tom. Ellers returneres `status = TRUE`.

Funktionerne er navngivet således, at en boolsk funktion returnerer sandt eller falsk ift. logikken i navnet.

6.2 Slave

FPGA'en skal opsættes således, at den korresponderer med SPI-opsætningen af mikrocontrolleren. FPGA'en skal ikke på samme måde som mikrocontrolleren opsættes med nogle

bestemte bit for at få protokollen til at fungere, men i stedet skal protokollen kodes fra bunden for at få modtagelse og afsendelse af databeskeder til at fungere.

I bogen *Digital System Design with FPGA...*^[7] er der opgivet kodeeksempler for både modtagelse og afsendelse af data for FPGA'en i både slave- og master-tilstand. Disse eksempler blev anvendt for at teste funktionaliteten af protokollen, men blev ikke brugt i den endelige version.

```
-- TRANSMIT
-- Falling_edge(ss) - Prepare transmission
if (pre_ss = '1') and (cur_ss = '0') then
    miso <= '0';
    miso_data_temp <= miso_data;
    index <= data_length - 1;
end if;

-- Falling_edge(sck)      - Write data
if (pre_sck = '1') and (cur_sck = '0') then
    miso <= miso_data_temp(index);
end if;

-- RECEIVE
-- Rising_edge(sck)      - Read data
if (pre_sck = '0') and (cur_sck = '1') then
    mosi_data_temp(index) <= mosi;
    index <= index - 1;
end if;

-- Rising_edge(ss)        - End transmission
if (pre_ss = '0') and (cur_ss = '1') then
    miso <= '1';
    miso_data_temp <= (others => '0');
    mosi_data <= mosi_data_temp;
    clear_miso <= '1';
end if;

pre_ss <= cur_ss;
pre_sck <= cur_sck;
pre_mosi <= cur_mosi;
```

Figur 15: Afsendelse og modtagelse af data.

Figur 15 viser koden, der står for afsendelse og modtagelse af data fra slaven. Afsendelse af data fungerer ved, at der foretages et `falling_edge`-tjek på `ss`. Dette gøres ikke med den indbyggede kommando: `falling_edge`, da alt koden er synkroniseret efter et `rising_edge`-tjek på FPGA'ens egen system-clock, `clk`. Der kan altså ikke være en edge-detektering inde i en edge-detektering, og især ikke når mikrocontrollerens clock ikke er synkroniseret med FPGA'ens. Derfor foretages en `falling_edge`-detektering ved at `pre_ss = '1'` (`ss` for forrige `clk`) og `cur_ss = '0'` (`ss` for nuværende `clk`).

Hvis der er `falling_edge`-detektering på `SS` afsendes logisk 0 til master (gennem MISO) for at sikre en kendt værdi på porten inden transmissionen starter. Herudover gemmes det man ønsker at sende i et midlertidigt signal, og indekset får tildelt værdien 15 (`data_length = 16`).

Når der forekommer `falling_edge` på `sck` skrives bitten ift. indekset ud på `miso`. Dette gøres ved hvert `falling_edge` på `sck` indtil der ikke er mere at afsende.

Modtagelse af data fungerer ved, at der foretages et `rising_edge`-tjek på `sck`. Hvis dette er tilfældet gemmes dét som modtages fra `mosi` i et midlertidigt signal ift. indekset. Ydermere

dekrementeres indekset. Dette gøres ved hvert `rising_edge` på `sck` indtil der ikke er mere at modtage.

Transmission afslutter når der forekommer `rising_edge` på `ss`. Her afsendes logisk 1 til `miso` for at sikre en fast værdi; det midlertidige signal for `miso` tømmes, og det midlertidige signal for `mosi` gemmes på `mosi_data` og sendes videre til databehandling.

Til sidst opdateres `pre_ss`, `pre_sck` og `pre_mosi` til de nuværende værdier for henholdsvis `ss`, `sck` og `mosi`.



Figur 16: Testbench for FPGA'ens SPI-komponent.

På figur 16 ses en testbench over, hvordan FPGA'ens SPI-komponent modtager og afsender en besked til mikrocontrolleren. Her er det en besked på 16 bit, som er den datalængde der arbejdes med, hvilket vil blive uddybet senere. Den modtagne besked (MOSI) og afsendte værdi (MISO) er:

$$MOSI = 1101 \quad 0101 \quad 1110 \quad 0101 \quad MISO = 1001 \quad 0101 \quad 1011 \quad 0110$$

På testbenchen ses det, hvordan hvert bit i mikrocontrollerens besked modtages på `mosi`, og læses over i et 16-bit register `mosi_data_temp` på hver `rising_edge` af `sck`. Først når alt er modtaget og der er `rising_edge` på `ss`, lægges `mosi_data_temp` på `mosi_data`, som derefter sendes videre til behandling. Ligeledes findes en fast værdi på `miso_data`, der ved hver `falling_edge` af `sck` bit for bit, lægges ind på `miso`.

7 Kommunikationsprotokol

SPI-protokollen er nu opsat på både mikrocontroller og FPGA med master/slave-princippet. Derfor er det muligt at opsætte den ønskede kommunikationsprotokol. Denne protokol skal opfylde krav omkring brugerinput til at sende værdier til de to motorer. Herudover skal det være muligt at hente værdier fra motorerne, så regulatoren herefter kan justere motorerne i form af feedback.

7.1 Protokol

Tabel 4 viser de fire beskeder, der benyttes mellem mikrocontrolleren og FPGA'en, som skal til for at transmittere de nødvendige informationer:

SET				
Type 2 bit	Motorvalg 3 bit	Reserveret 3 bit	Retning 1 bit	PWM-værdi 7 bit
00	001 (M1) 010 (M2)	000	0 / 1	0 - 100 ₁₀
GET				
Type 2 bit	Motorvalg 3 bit	Reserveret 11 bit		
01	001 (M1) 010 (M2)	000 0000 0000		
REPLY				
Type 2 bit	Motorvalg 3 bit	Reserveret 1 bit	Retning 1 bit	enkoder-værdi 9 bit
10	001 (M1) 010 (M2)	0	0 / 1	0 - 360 ₁₀
POLL				
Type 2 bit	Reserveret 14 bit			
11	00 0000 0000 0000			

Tabel 4: Kommunikationsprotokollens beskeder og deres opbygning.

- **SET:** SET afsendes af mikrocontrolleren, og benyttes til at sætte en værdi og en retning for en af de to motorer. Beskedtypen indikeres af koden "00". Som i de andre beskeder bruges nogle bit på at fortælle, hvilken motor der er tale om, og dernæst bruges ét bit på retning, og 7 bit på PWM-værdien. Der benyttes kun 7 bit da værdien altid vil være mellem 0 og 100, hvilket kun kræver 7 bit.
- **GET:** GET afsendes ligeledes af mikrocontrolleren, og bruges når der skal hentes informationer omkring motorernes positioner. Beskeden identificeres med koden "01", og den eneste parameter er valget af motor. Resten er overflødige bit; mikrocontrolleren

får derefter de efterspurgte informationer i en REPLY-besked.

- **REPLY:** REPLY afsendes af FPGA'en, der, efter at have modtaget og afkodet en GET-besked, sender en REPLY-besked tilbage til mikrocontrolleren. Beskeden indeholder en type "10", den givne motor, en retning og 9 bit til at angive, hvor motoren er henne ifølge en enkoder, som konstant afkodes af FPGA'en.
- **POLL:** Da FPGA'en først ved, hvilke værdier den skal sende tilbage, efter den har modtaget en hel GET-besked fra mikrocontrolleren, er det nødvendigt, at mikrocontrolleren sender en tom besked for at tillade FPGA'en at sende mere data, da dette kun kan ske når masteren aktiverer SPI-protokollen.

Alle beskeder har mindst ét overflødigt bit, men det er alligevel besluttet at benytte beskeder på 16 bit, da mikrocontrolleren ikke anvender datatyper á størrelser mellem otte og 16 bit.

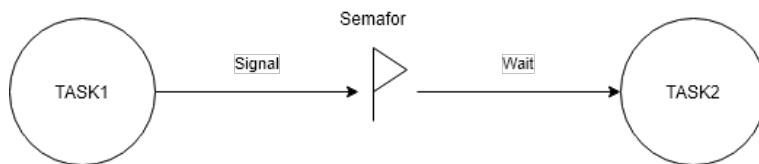
7.2 Implementering på mikrocontroller

Når der afsendes værdier til de to motorer, gemmes to beskeder (ét til hver motor) i en FIFO-kø før de skrives til FPGA'en vha. SPI-protokollen. Der kan maksimalt være 128 elementer i køen.

Når indsættelse af elementer på en kø fremtraede, udføres tjek på semaforer.

En semafor er en heltalsvariabel, som kan tilgås vha. to atomiske operationer: **signal** og **wait**. **wait**-operationen udføres når en task ønsker at få adgang til eksempelvis en kø, hvor **signal**-operationen udføres når en task er færdig med at tilgå køen. På denne måde undgås det, at to tasks prøver at tilgå samme element i køen samtidigt. Semaforen, S, tælles ned hvis S er over 0. Hvis dette ikke er tilfældet kan en task ikke tilgå køen før semaforen er talt op vha. **signal**-operationen.

Figur 17 viser et eksempel på et taskdiagram, hvor disse to operationer anvendes:

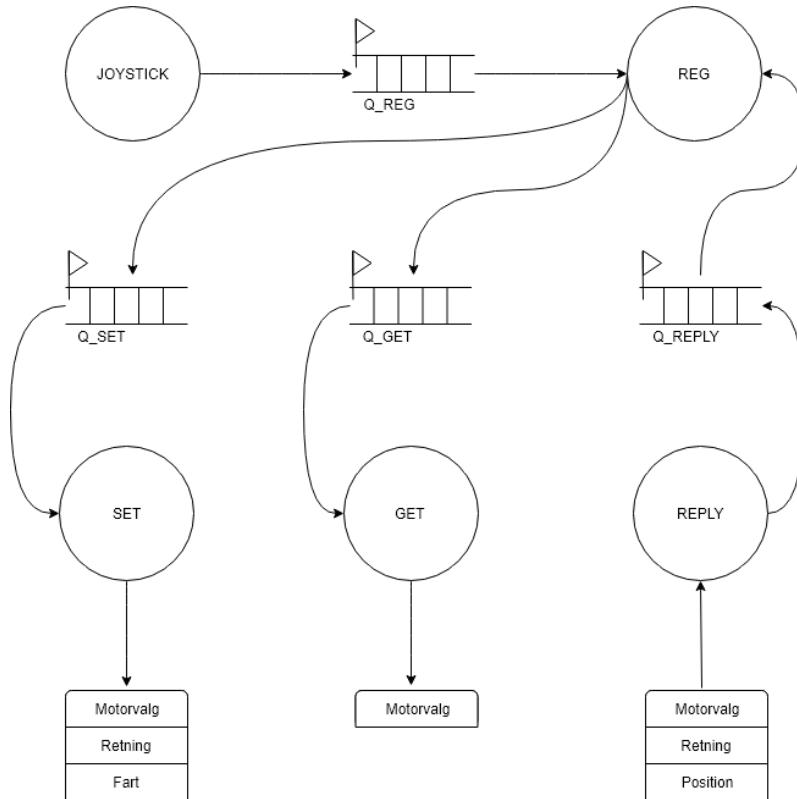


Figur 17: Eksempel på semafor med signal og wait.

7.2.1 Task-diagram

Task-diagrammet for kommunikationsprotokollen ses på figur 18.

Cirklerne repræsenterer tasks, imellem er event-buffere (her køer), og tabellerne er state-buffere der kan indeholde f.eks. variabler. Et task-diagram skal ikke forveksles med et flowchart, da et flowchart kører sekventielt, og et task-diagram kører parallelt.

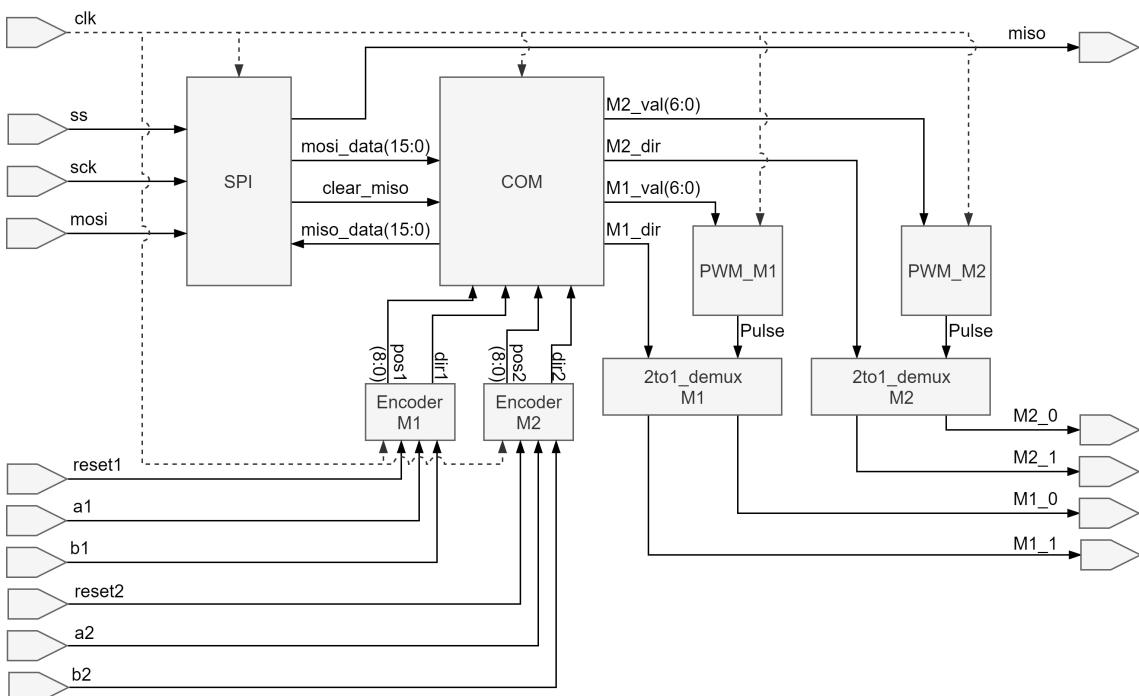


Figur 18: Task-diagram for kommunikationsprotokollen.

- JOYSTICK: Denne task gemmer ADC-værdier, konverteret til grader, ud fra den valgte motor, samt en specifik retning. Dette data samles i en besked på 16 bit og indsættes på en kø, Q_REG.
- REG: Denne task tager indholdet af beskeden fra Q_REG, og bearbejder disse data. Tasken kan indsætte værdier ind på både Q_SET og Q_GET. REG-tasken henter først værdier ved at sætte et valg af motor ind på Q_GET, hvor man så herefter kan hente værdier fra Q_REPLY, så systemet kan reguleres herefter. Det er så muligt at indsætte en besked på køen Q_SET efter at have modtaget fra Q_REPLY-køen.
- SET: Denne task tager indholdet af beskeden fra Q_SET (hvis der er flere beskeder, udtages fra FIFO-princippet), og skriver disse værdier ud til FPGA'en vha. SPI-protokollen. Her opdateres: Motorvalg, retning og fart.
- GET: Denne task tager indholdet af beskeden fra Q_GET ud fra FIFO-princippet. Beskeden sendes til FPGA'en. Her opdateres motorvalg.
- REPLY: Denne task får tilsendt en besked fra FPGA'en med motorvalg, retning og position. Dette indsættes på Q_REPLY.

7.3 Implementering på FPGA

FPGA'ens program er opbygget af komponenter, hvilke er individuelle blokke af kode, som kun kan kommunikere vha. interne signaler mellem dem. Centralt i designet ligger **COM**-komponenten, der håndterer beskederne fra mikrocontrolleren og handler derudfra. Omkring denne findes komponenterne **PWM** og **2to1_demux**, som styrer motorerne, og **Encoder**, der aflæser motorernes positioner. På figur 19 ses et diagram over FPGA'ens komponenter, input og output og signalerne imellem.



Figur 19: Diagram over FPGA'ens opbygning af komponenter.

- **PWM:** Til hver motor findes en PWM-komponent. Denne komponent får en værdi fra 0 - 100, som den laver om til en PWM-puls med en arbejdscyklus svarende til inputtet i procent, mere herom i afsnit 8.1.
- **2to1_demux:** Disse to komponenter får en puls fra den tilsvarende PWM-komponent, og afhængig af værdien af retnings-inputtet sendes pulsen ud på én af komponentens to forbundne porte, og videre til H-broen og motorerne.
- **Encoder:** I hver motor sidder en enkoder, der udsender to signaler forskudt af hinanden når akslen roterer, hvilket kan bruges til at fortælle motoren's position, mere herom i afsnit 8.2. I FPGA'en er der implementeret en **encoder**-komponent til hver motor, der ved hjælp af de to signaler fra de fysiske enkodere tæller, hvilke grader pan-tilt systemet står i. På systemet findes også to hall-sensorer, som udsender et signal når systemet passerer. Disse bruges til at nulstille enkoder-værdierne.

- SPI: SPI-komponenten udfører al kommunikation med mikrocontrolleren beskrevet i afsnit 6.2. Når en besked skal sendes bliver beskeden læst fra `miso_data`-signalet fra COM-komponenten, og når en besked er modtaget sendes den tilbage på `mosi_data`-signalet.
- COM: Hele FPGA'ens design styres af COM-komponenten. Når en besked modtages af SPI, afkodes beskeden ved først at tjekke typekoden.

Modtages en SET-besked tjekkes det, hvilken motor der er tale om, hvorefter PWM-værdi og retning sendes videre til de pågældende PWM og `2to1-demux`-komponenter, der har forbindelse til motorerne.

Modtages en GET-besked tjekkes der, hvilken motor der ønskes værdier for. Dernæst skrives en REPLY-besked med type-koden 10 og den pågældende motor, samt retningen og positionen fra den pågældende enkoder, i forhold til protokollen i tabel 4.

Normalt foregår SPI-protokollen ved, at slaven altid sender noget når masteren gør, men i dette tilfælde ved slaven ikke, hvad den skal sende, før den første GET-besked er modtaget. For at forhindre FPGA'en i at sende tilfældige beskeder til mikrocontrolleren, hver gang der modtages en vilkårlig besked, er der oprettet et signal `clear_miso` fra SPI, som får COM til at nulstille `miso_data`. Dette signal sendes til COM hver gang en transmission er slut. Hvis der lige er modtaget en GET-besked får `miso_data` den tilsvarende REPLY-værdi, men hvis der modtages en SET-besked vil `miso_data` forblive 0, og der sendes ingen besked retur ved næste transmission.

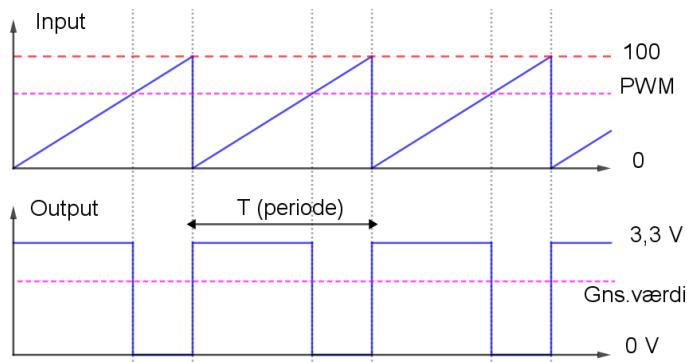
FPGA'ens svar vil altså altid være én transmission bagud. Derfor sendes der én tom POLL-besked fra mikrocontrolleren ved første GET-besked, så FPGA'en på den måde har indhentet den ekstra transmission.

8 Motorstyring

Fra FPGA'en udsendes to signaler til hver motor med en PWM-puls i én af de to retninger. Disse kommer ind i H-broen, hvor de skaleres op fra FPGA'ens 3,3 V til 12 V, og går videre til motorerne. Fra pan-tilt systemet går desuden seks signaler ind i FPGA'en fra motorernes enkodere og hall-sensorer, hvilket gennemgås i følgende afsnit.

8.1 PWM

Motorernes hastigheder styres ved brug af Pulse-Width Modulation (PWM). PWM virker ved at regulere tiden for, hvornår en spænding er henholdsvis høj og lav. Den gennemsnitlige spænding vil derved ændre sig afhængigt af forholdet mellem tiden, hvor den er høj og lav.



Figur 20: Illustration af PWM.

PWM-komponenten i FPGA'en er opbygget således, at ved hver `rising_edge` på `clk` inkrementeres en tæller op til 100, hvorefter den startes forfra. Så længe tællerens værdi er lavere end den valgte pulsbredde, er motoren tændt. Konceptet for PWM kan ses illustret på figur 20.

PWM-komponenten simuleres for de to motorer på figur 21:



Figur 21: Simulering af PWM-signal til de to motorer.

I ovenstående figur svarer [0] og [1] til de to retninger for motor 1, mens [2] og [3] svarer til det samme for motor 2. PWM-signalet starter med en pulsbredde på 25 % til begge motorer i én retning. Dette øges efterfølgende til henholdsvis 50 %, 75 % og 100 %. Derefter gentages forløbet på begge motorer, men i modsat retning.

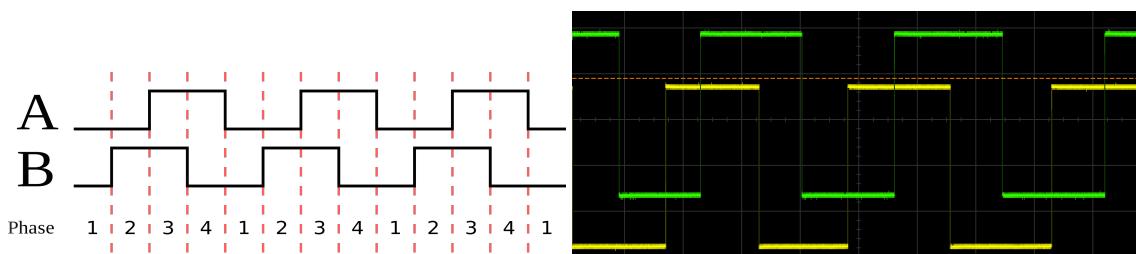
Til sidst sættes pulsbredden til 0 %, hvor der ses en kort aktivering af [0] og [2], hvilket skyldes at signalet, der bestemmer pulsbredde og retninger skifter øjeblikkeligt, mens PWM-signalet først skifter ved `rising_edge` på `clk`. Der kan altså opstå en form for fejlsignal, hvor en motor er tændt i højst én clock-cyklus, hvilket dog er for kort til at have en betydelig indflydelse.

8.2 Hall-sensorer og enkodere

En hall-sensor indeholder et stykke metal, hvorigennem en strøm udsendes. Når et magnetfelt kommer i nærheden af sensoren, skubbes elektronerne mod én af kanterne, hvilket skaber en spændingsforskel på tværs af strømretningen. Denne spænding kan så måles.

I dette projekt anvendes to hall-sensorer til at detektere når de to platforme (rammen og gaflen) i pan-tilt-systemet når deres respektive startpositioner. Desuden befinner der sig to hall-sensorer i hver motor, der anvendes til at bestemme deres kørselsretninger og positioner ved hjælp af en enkoder.

De to hall-sensorer i motoren er placeret forskudt fra hinanden, og retningen kan derfor bestemmes ved at aflæse faseforholdet mellem de to signaler. På figur 22 og 23 ses henholdsvis princippet og de faktiske målinger af udslagene fra to signaler, kaldet henholdsvis A og B. A leder i dette tilfælde B med en kvart periode, og er det omvendte tilfældet kører motoren den modsatte retning.



Figur 22: Signal A leder signal B. [8]

Figur 23: Målte enkoder-signaler.

I encoder-komponenterne i FPGA'en måles signalerne A og B løbende. Når et af signalerne skifter, bestemmes retningen af rotationen ved at XOR're A og det tidligere resultat for B. Afhængig af den fundne retning bliver en tæller for positionen enten inkrementeret eller dekrementeret. Derudover kan positionen nulstilles når den stationære hall-sensor passerer.

Regulatorerne (se afsnit 10) bruger enkoderne til at bestemme systemets positioner. Fra regulatorerne er der sat en grænse for yderpositionerne på $\pm 30^\circ$ fra hall-sensorerne.

9 Matematisk modellering af systemet

Applikationen for Labyrint-spillet kan nu anvendes med et joystick igennem en kommunikationsprotokol vha. SPI-protokollen. På grund af høj sensitivitet og mekanisk begrænsede områder, skal spillet også reguleres, således at pan-tilt systemet ikke kan gå ud i uønskede ydre områder. Herudover begrænses ufrivillige sensitive handlinger fra joysticket. Hidtil har joysticket styret motorerne direkte med en valgt hastighed; nu skal motorerne reguleres ift. en specifik position. Først bestemmes den matematiske modellering af systemet.

I dette afsnit anvendes information fra *Modeldannelse (Kapitel 2)*...^[9].

Pan-tilt systemet består af to DC-motorer. Rotation i en DC-motor skyldes påført spænding over sine terminaler. Denne type motor består af en elektrisk del og en mekanisk del, og derfor vil kraftoverførelsen ideelt have følgende sammenhæng:

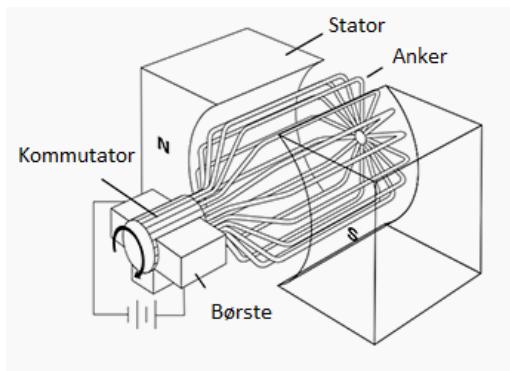
$$V \cdot I = \tau \cdot \omega \quad [\text{W}] \quad (1)$$

hvor V er spænding [V] ideelt set over udgangsterminalerne. I er strømmen [A], τ er drejningsmoment [Nm] og ω er vinkelhastigheden [$\frac{\text{rad}}{\text{s}}$].

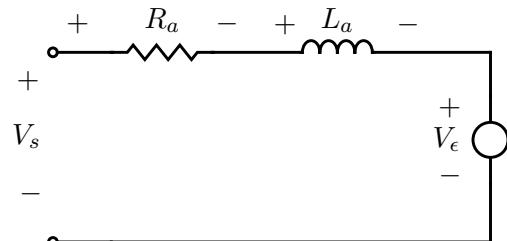
Først betragtes den elektriske del af kredsløbet.

9.1 Det elektriske system

En DC-motor består af én til flere spoler på en rotor, kaldet et anker (se figur 24), der roterer i en magnet, kaldet en stator. Rotorens elektriske ækvivalent kan ses som en modstand i serie med en spole og den mod-elektriskraft V_ϵ (se figur 25) .



Figur 24: Beskrivelse af en DC-motor.^[10]



Figur 25: Det ækvivalente elektroniske kredsløb for en DC-motor.

Gennem Kirchhoffs spændingslov fås følgende ligning:

$$V_s - V_R - V_L - V_\epsilon = 0 \quad [\text{V}] \quad (2)$$

hvor V_s er spænding over indgangene på motoren, V_L er spændingen over spolen og V_R er spændingen over modstanden.

$$V_L = L \cdot \frac{d}{dt} I \quad [\text{V}]$$

Eftersom V_L afhænger af den afledte af strømmen, omskrives de resterende led i ligning (2) til også at afhænge af strømmen. Spændingen over modstanden omskrives til:

$$V_R = R \cdot I \quad [\text{V}]$$

Når rotoren drejer i statoren indukseres en mod-elektrisk kraft V_ϵ , med enheden volt, som er modsatrettet forsyningsspændingen V_s , og kan beskrives med følgende ligning:

$$V_\epsilon = k_v \cdot \omega \quad [\text{V}]$$

hvor k_v er en hastighedskonstant, som angiver fluxdensiteten af statoren.

De overstående udtryk indsættes i ligning (2):

$$V_s - R_a \cdot I_a - L_a \cdot \frac{d}{dt} I_a - K_v \cdot \omega = 0 \quad [\text{V}] \quad (3)$$

Dette beskriver den elektriske del af motoren. Herefter betragtes motorens mekaniske del.

9.2 Det mekaniske system

Det undersøges hvilke kræfter, som påvirker DC-motoren, og herefter opstilles en ligning for energibalance jævnfør Newtons 1. lov, hvor summen af systemets drejningsmomenter skal give 0. Der er et elektromagnetisk drejningsmoment T_e , som kan beskrives som:

$$T_e = K_t \cdot I_a \quad [\text{Nm}]$$

hvor K_t er drejningsmoment-konstanten, som beskriver fluxdensiteten i statoren. Derudover er der en dæmpning, som afhænger af vinkelhastigheden:

$$T_b = B \cdot \omega + \tau_c \cdot \text{sign}(\omega) \quad [\text{Nm}]$$

hvor B er en dæmpningskonstant og τ_c er den statiske friktion. Der er også et drejningsmoment fra vinkelaccelerationen. Da man er interesseret i drejningsmomentet udtrykt ved vinkelhastigheden, bliver denne omskrevet på følgende måde:

$$T_{\omega'} = J \cdot \alpha = J \cdot \frac{d}{dt} \omega \quad [\text{Nm}]$$

hvor α er vinkelaccelerationen og J er inertimomentet.

Drejningsmomentet fra belastningen (pan-tilt systemet) kan udtrykkes som:

$$T_L = g \cdot J_L \cdot \frac{d}{dt} \omega \quad [\text{Nm}]$$

hvor J_L er inertimomentet fra belastningen og g angiver gearing mellem motoren og belastningen. For at motoren kan holde en konstant hastighed, kræver det, at alle kræfter udligner hinanden. Det er kun det elektromagnetiske drejningsmoment, som tilfører energi til systemet, og derfor bliver energiligningen følgende:

$$T_e - T_b - T_{\omega'} - T_L = 0 \quad [\text{Nm}]$$

Udtrykkene for de enkelte dele i ovenstående ligning indsættes:

$$K_t \cdot I_a - (B \cdot \omega + \tau_c \cdot \text{sign}(\omega)) - J \cdot \frac{d}{dt} \omega - g \cdot J_L \cdot \frac{d}{dt} \omega = 0 \quad [\text{Nm}] \quad (4)$$

9.3 Tilstandsmodel

Fra de to differentialligninger for henholdsvis det elektriske og det mekaniske system, opstilles en tilstandsmodel for motoren. Den højest afledte isoleres i ligning (3):

$$\frac{d}{dt} I_a = \frac{1}{L_a} V_s - \frac{R_a}{L_a} I_a - \frac{K_v}{L_a} \omega \quad (5)$$

Det samme gøres for ligning (4):

$$\frac{d}{dt} \omega = \frac{K_t}{J + g \cdot J_L} I_a - \frac{B}{J + g \cdot J_L} \omega - \frac{\tau_c}{J + g \cdot J_L} \cdot \text{sign}(\omega) \quad (6)$$

De to ligninger opsættes som en tilstandsmodel:

$$\begin{bmatrix} \frac{d}{dt} I_a \\ \frac{d}{dt} \omega \\ \frac{d}{dt} \theta \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_v}{L_a} & 0 \\ \frac{K_t}{J+J_L \cdot g} & -\frac{B+\tau_c \cdot \text{sign}}{J+J_L \cdot g} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} I_a \\ \omega \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_A} \\ 0 \\ 0 \end{bmatrix} V_s \quad (7)$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_a \\ \omega \\ \theta \end{bmatrix} \quad (8)$$

9.4 Motorkonstanter

Design af regulatoren og simulering af systemet kræver udregning af motorkonstanterne i tilstandsmodellen: R_a, L_a, J, K, τ_c og B . Hidtil er K_v og K_t benævnt separat, men fremover benævnes de blot som motorkonstanten K .

Disse bestemmes gennem tre forsøg beskrevet herunder, og som uddybes i journalen *Bestemmelse af motorkonstanter*, der kan findes i det elektroniske bilag.

9.4.1 Bestemmelse af K , R_a , τ_c og B

Til at bestemme K , R_a , τ_c og B laves et forsøg, hvor motorens omdrejningshastighed ω og ankerstrøm I_a måles ved 12 forskellige ankerspændinger, V_s , mellem 1 og 12 volt, i spring af 1 volt. Ved hver målespænding noteres motorens omdrejningshastighed og ankerstrøm. Herefter laves to funktioner gennem regression af intervallet fra den ankerspænding, som overvinder den statiske friktion, τ_c , og op til 12 volt. Én funktion for omdrejningshastighed som funktion af ankerspænding, $\omega(V_s)$, og én for ankerstrøm som funktion af ankerspænding, $I_a(V_s)$. For hver af disse to funktioner, udregnes to punkter ift. de samme arbitrære ankerspændinger V_{s1} og V_{s2} , altså de følgende fire punkter:

$$(V_{s1}, \omega_1(V_{s1})), \quad (V_{s2}, \omega_2(V_{s2})), \quad (V_{s1}, I_{a1}(V_{s1})), \quad (V_{s2}, I_{a2}(V_{s2}))$$

Når omdrejningshastigheden ved en påtrykt spænding bliver konstant, bør strømmen I_a også være det, hvilket medfører at ligning (3) kan reduceres og omskrives til:

$$V_s = R_a \cdot I_a + K \cdot \omega \quad [V] \quad (9)$$

Ligning (4) reduceres og omskrives, med positiv sign(ω) indsat:

$$T_L = K \cdot I_a - B \cdot \omega - \tau_c \quad [\text{Nm}] \quad (10)$$

Ud fra to par af V_s og I_a , henholdsvis V_{s1} , I_{a1} og V_{s2} , I_{a2} , kan der fra (9) og (10) opstilles fire ligninger med fire ubekendte:

$$V_{s1} = R_a \cdot I_{a1} + K \cdot \omega_1, \quad V_{s2} = R_a \cdot I_{a2} + K \cdot \omega_2 \quad [V] \quad (11)$$

$$T_L = K \cdot I_{a1} - B \cdot \omega_1 - \tau_c, \quad T_L = K \cdot I_{a2} - B \cdot \omega_2 - \tau_c \quad [\text{Nm}] \quad (12)$$

Når motoren ikke belastes udefra er $T_L = 0$, og ovenstående ligninger kan løses for variablene: R_a , K , B og τ_c . Da systemet har to motorer laves forsøget på hver af disse. For de to motorer blev der observeret en relativ stor forskel mellem værdien for τ_c , som afveg med en faktor på 1,6. Derfor blev forsøget gentaget. Her var afvigelsen mellem de to motorers τ_c -værdier ganske lille; til gengæld var der nu en afvigelse på B -værdierne med en faktor 1,6, hvor der under forsøg 1 kun var en afvigelse på en faktor 1,2. Det blev derfor besluttet at bruge gennemsnittet for de to forsøg, da forskellen muligvis ligger i måleunøjagtigheder og variationer som motorernes interne temperaturer. I nedstående tabel ses værdierne for begge motorer, opdelt efter forsøg 1 og forsøg 2, samt deres gennemsnitsværdier:

Rammemotor				Gaffelmotor			
Forsøg 1				Forsøg 1			
R_a	K	B	τ_c	R_a	K	B	τ_c
8,9706	0,3367	0,001031	0,02426	8,5752	0,37897	0,0006281	0,03006
Forsøg 2				Forsøg 2			
R_a	K	B	τ_c	R_a	K	B	τ_c
6,8223	0,3456	0,0005136	0,09532	7,5448	0,3763	0,0005382	0,06087
Gennemsnit				Gennemsnit			
R_a	K	B	τ_c	R_a	K	B	τ_c
7,8965	0,3453	0,0007721	0,05979	8,06001	0,3776	0,0005832	0,04546

Tabel 5: Værdier for K , R_a , τ_c og B .

9.4.2 Bestemmelse af L_a

Forsøget for bestemmelse af L_a sker ved at påtrykke motoren et spring i ankerspænding; fra 0 volt til en værdi, der er lav nok til at den ikke begynder at rottere. Samtidig måles ankerstrøm ift. tid. Under opvoksningsforløbet angiver tidskonstanten forholdet $\tau = \frac{L_a}{R_a}$. Når ankerstrømmen er blevet konstant kan R_a bestemmes ved:

$$R_a = \frac{V_s}{I_a} \quad [\Omega]$$

Denne statiske værdi af R_a er sandsynligvis forskellig fra den, der blev bestemt, da motoren roterede, da modstanden kan variere under rotation. Derfor bruges den statiske R_a kun til at udregne L_a .

Gennemsnitsværdier, L_a og R_a , for gaffelmotor:

$$R_a = 5,10 \Omega \quad L_a = 0,0936 \text{ H}$$

Gennemsnitsværdier, L_a og R_a , for rammemotor:

$$R_a = 20,35 \Omega \quad L_a = 0,0351 \text{ H}$$

Der observeres en relativ stor forskel på de to motorers R_a -værdier, men dette er afprøvet på flere forskellige dage, med samme tendens, og kan muligvis skyldes forskelle i motorenes interne modstande, når de står stille. Fra forrige forsøg ses det dog, at de er relativt identiske når motorerne kører. R_a -værdierne for dette forsøg bruges kun til at bestemme L_a , men dette medfører alligevel en afvigelse mellem L_a for de to motorer.

9.4.3 Bestemmelse af motorens inertimoment, J

Der laves et spring i ankerspændingen, fra 0 til 10 volt, imens der på motoren måles omdrejningshastighed. Ud fra dette kan der laves en graf for ændringen af omdrejningshastighed

ift. tid. Tidskonstanten τ er givet som:

$$\tau = \frac{R_a \cdot J}{R_a \cdot B + K^2} \quad (13)$$

Tidskonstanten kan opskrives på denne måde ved at se bort fra L_a i overføringsfunktionen, således at den går fra en 2. ordens til en 1. ordens. Denne tilnærmelse kan gøres, da den elektriske tidskonstant er langt mindre end den mekaniske. Den elektriske del af overføringsfunktionen er betydeligt hurtigere end den mekaniske, og vil derfor kun bidrage ganske lidt til den samlede tidskonstant. Inertimomentet er den eneste ukendte variabel i denne ligning, og kan derfor bestemmes. Da motoren kører under dette forsøg, anvendes den gennemsnitlige dynamiske R_a fra tabel 5.

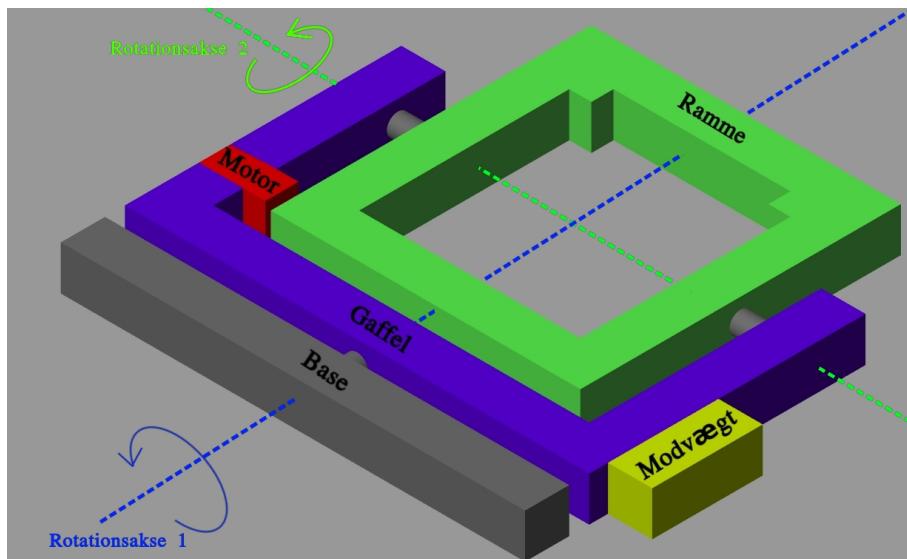
Gennemsnitsinertimoment for henholdsvis gaffelmotor og rammemotor:

$$J_{gaffelmotor} = 0,0003960 \text{ kg m}^2, \quad J_{rammemotor} = 0.00021730 \text{ kg m}^2$$

Igen observeres en relativ stor forskel mellem de to motorer, hvilket stemmer overens med, at der formentlig er forskel, da de f.eks. ikke nødvendigvis har samme alder eller driftstid.

9.5 Systemets inertimomenter

Yderligere detaljer om dette: se *Inertimomenter for systemet* under det elektroniske bilag.



Figur 26: Systemet med dets rotationsakser.

Drejningsmomentet T_L indgår i tilstandsmodellen, og for at bestemme denne, skal inertimomentet for selve rammen og gaflen udregnes. Systemet har to rotationsakser: rotationsakse 1 og rotationsakse 2, henholdsvis for gaflens og rammens motor. Disse har hvert deres inertimoment, som kan estimeres vha. en simplificeret model.

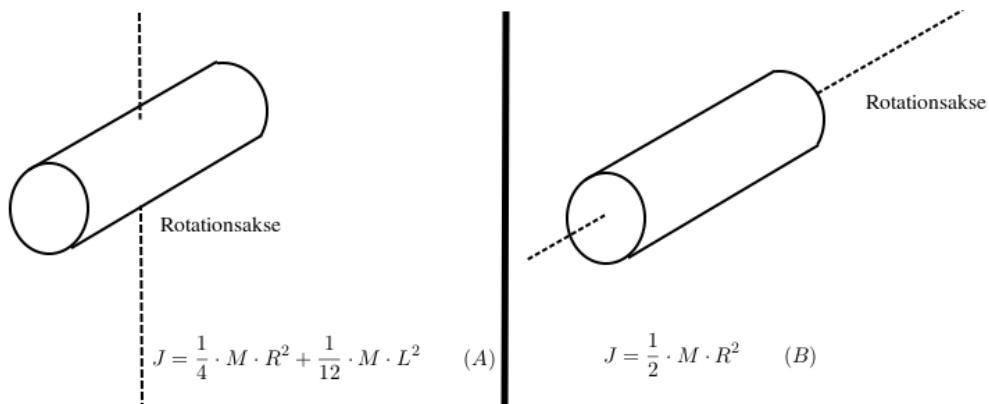
Motoren som driver rammen er monteret på gaflen (se figur 26), hvilket gør, at gaflen ikke er i balance omkring rotationsakse 1. Derfor er en modvægt tilføjet på modsatte side af rammen, som dermed afbalancerer gaflen.

Overordnet kan inertimomenter regnes på to måder: enten som massemidtpunkter eller som legemer. Når inertimomenter regnes som punktmasser, anvendes denne ligning:

$$J = M \cdot d^2 \quad [\text{kg} \cdot \text{m}^2] \quad (14)$$

hvor M er massen og d er afstanden fra massens midtpunkt til rotationsaksen.

Regnes der med kontinuerte legemer, kan der bruges forskellige formler alt efter legemernes facon og orientering ift. rotationsaksen. Formlerne for cylinderlegemer kan ses på nedenstående figur (27):



Figur 27: Formler for inertimoment, regnet som cylinderlegemer efter deres orientering til rotationsaksen [11], hvor R er radius og L er længde.

Forskellen ligger i, at hvis et legeme har massemidtpunkt i dets rotationsakse, altså når $d = 0$ i ligning (14), giver det intet inertimoment, modsat især formel (A) i figur 27.

Til en model for inertimomenterne via kontinuerte legemer, vælges det at se systemet som værende sammensat af cylinderlegemer. Dette er ikke helt tilfældet i virkeligheden, og de er heller ikke massive, men det forventes, at det trods alt giver en mere realistisk model, end udelukkende at regne med punktmasser.

Grunden hertil er, at rammens position vil ændre inertimomentet set fra gaffelmotoren på rotationsakse 1. Står rammen i vandret position, og der regnes med punktmasser, findes der intet bidrag fra rammens top og bund, da rotationsakse 1 går direkte gennem deres massemidtpunkt. Regnes der med legemer, vil der altid tages højde for hele systemet. Rammens position påvirker dog ikke inertimomentet for rammemotoren om rotationsakse 2.

Man kunne opstille en funktion for inertimomentet om rotationsakse 1, afhængig af vinklen på rammen, især regnet som punktmasser. Det er derimod mere vanskeligt, når rammen regnes som værende sammensat af cylinderlegemer, da der bruges to forskellige formler alt efter, hvordan legemet vender ift. rotationsaksen. Se figur 27.

I første omgang vælges det at undersøge inertimomenterne for de to yderpunkter af rammens position; nemlig når den er vandret med gaflen, som på figur 26, og når den er roteret 90 grader og står vinkelret på gaflen.

I tabel 6 herunder ses værdierne for inertimomenterne for de to forskellige regnemetoder, opdelt efter de to forskellige positioner. Med kombination menes, at de store uniforme dele (L-legemerne i *Inertimomenter for systemet*) er regnet som cylindermasser, og de mindre som punktmasser. De mindre dele, f.eks. vinkelbeslagene, er regnet som punktmasser.

Inertimoment for rammemotor [kg m^2]			
Punktmasser		Kombination	
0,0108		0,0171	
Inertimoment for gaffelmotor [kg m^2]			
Ramme lige (0°)		Ramme vinkelret (90°)	
Punktmasser	Kombination	Punktmasser	Kombination
0,0540	0,0656	0,0633	0,0810

Tabel 6: Systemets inertimomenter i forhold til forskellige positioner og regnemetoder.

Ud fra ovenstående resultater vælges metoden for kombination af massemidtpunkt og cylinderlegemer til at bestemme inertimomentet for rammemotoren om rotationsakse 2, da denne forventes at være mest korrekt.

For gaffelmotorens inertimoment om rotationsakse 1 bemærkes det, at der ligeledes er en signifikant forskel mellem de to regnemetoder. Fordelen ved at vælge punktmasser ville være, at det er relativt ligetil at opstille et udtryk for inertimomentet som funktion af vinklen. Ulempen ville være at den ved fuldt udsving, altså 90° , kun lige når op på samme værdi som kombinationsmetoden starter på, ved 0° . Ydermere forventes systemet kun at operere i vinkler inden for intervallet $[-30^\circ; 30^\circ]$, og derfor benyttes inertimomentet fra 0° -positionen udregnet med kombinationsmetoden for gaffelmotoren om rotationsakse 1.

Værdier for de to inertimomenter:

Henholdsvis rotationsakse 1, rammemotoren, og rotationsakse 2, gaffelmotoren:

$$J = 0,0171 \text{ kg m}^2, \quad J = 0,0656 \text{ kg m}^2$$

10 Regulering af systemet

Den matematiske model for systemet er nu bestemt; derfor foretages regulering af systemet ud fra positionen af motorerne. Dette afsnit omhandler, hvordan regulatorerne til de to motorer er designet og implementeret.

10.1 Krav

Design af en regulator kræver, at man har nogle krav til opførslen af systemet. Det ønskes, at regulatoren ikke har noget oversving og en indsvingningstid på omkring 1 sekund. Grunden til denne indsvingningstid er fordi spillet handler om præcision, og der ikke ønskes aggressive positionsændringer. Dog skal den ikke være så langsom, at man ikke kan nå at ændre kuglens kurs, men her vurderes 1 sekund som en passende tid.

Derudover ønskes en fasemargin på over 50 grader og en forstærkningsmargin på over 40 dB. Fasemargin beskriver, hvor meget systemet bliver påvirket af tidsforskydning, og forstærkningsmargin beskriver systemets evne til at håndtere støj.

10.1.1 I s-domænet

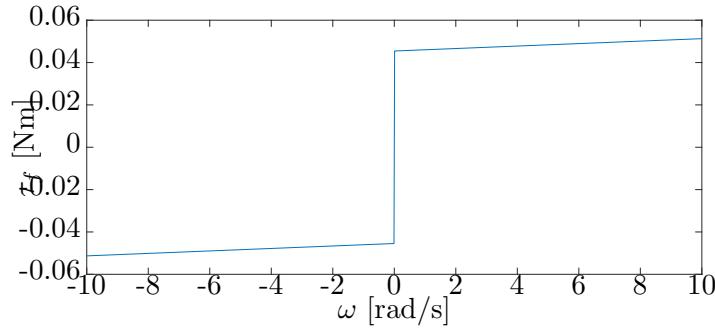
Når man kigger på kravene i s-domænet, kræver en indsvingningstid på 1 sekund, at polerne ligger til venstre for den lodrette linje, som skærer reel-aksen i punktet beregnet med følgende formel:

$$\sigma \geq \frac{4,6}{t_s}$$

Så i dette tilfælde skal polerne ligge under $Re(s) \leq -4,6$. For at sikre at der ikke er noget oversving, skal polerne for åben-sløjfe-systemet ligge på reel-aksen.

10.2 Klargøring af den matematiske model

Design af regulatoren kræver, at man har en lineær model, hvilket på nuværende tidspunkt ikke er tilfældet. Problemet er den statiske friktion τ_c , hvilket tydeliggøres på figur 28. τ_c skal kompenseres for, inden motorene begynder at opbygge hastighed.



Figur 28: Kraften hvormed friktionen τ_c modvirker rotationen i forhold til vinkelhastigheden

Der er to muligheder for linearisering af modellen. Første mulighed er at tage yderpunkterne og linearisere igennem origo. Dette vil give en dårlig tilnærmelse.

Den anden mulighed er at kompensere for den ulineariserede model på figur 28, så der ikke tages højde for dette af regulatoren. Der anvendes ligning (6), som også er gengivet herunder:

$$\frac{d}{dt}\omega = \frac{K_t}{J + g \cdot J_L} I_a - \frac{B}{J + g \cdot J_L} \omega - \frac{\tau_c}{J + g \cdot J_L} \cdot \text{sign}(\omega) \quad (6)$$

$K_t I_a$ udskiftes med τ_{in} :

$$\frac{d}{dt}\omega = \frac{\tau_{in}}{J + g \cdot J_L} - \frac{B}{J + g \cdot J_L} \omega - \frac{\tau_c}{J + g \cdot J_L} \cdot \text{sign}(\omega)$$

For at kompensere for $\tau_c \cdot \text{sign}(\omega)$ skal

$$\tau_{in} \equiv \tau_c \cdot \text{sign}(\omega) + K_t \cdot I_a$$

Eftersom det kun er spændingen V_s , som kan reguleres, skal kompenseringen føres videre. Derfor findes spændingen, hvorved der kommer en strøm, som ophæver $\tau_c \cdot \text{sign}(\omega)$. Dette kan gøres, hvis størrelsen af I_a bestemmes:

$$I_a = \frac{\tau_c \cdot \text{sign}(\omega)}{K_t} \quad (15)$$

Spændingen, som skal til for at ophæve den statiske friktion, kan så udregnes vha. ligning (6) ved at antage, at der ikke er nogen ændring i strømmen, og vinkelhastigheden $\omega = 0$:

$$\frac{d}{dt} I_a = \frac{1}{L_a} V_s - \frac{R_a}{L_a} I_a - \frac{K_v}{L_a} \omega \quad \Leftrightarrow \quad 0 = \frac{1}{L_a} \cdot V_s - \frac{R_a}{L_a} I_a$$

Hvis V_s isoleres, og I_a fra ligning (15) indsættes, fås udtrykket for spændingen:

$$V_s = R_a \frac{\tau_c \cdot \text{sign}(\omega)}{K_t} \quad (16)$$

Ved at benytte ovenstående ligning udregnes V_s for de to motorer, som vist i tabel 7:

Motor 1	0,9704 V
Motor 2	1,3673 V

Tabel 7: Spændingerne for at opnæve den statiske friktion i motorerne

Når man har kompenseret kan ligning (6) reduceres til:

$$\frac{d}{dt}\omega = \frac{K_t}{J + g \cdot J_L} I_a - \frac{B}{J + g \cdot J_L} \omega \quad (17)$$

10.2.1 Overførselsfunktion

Det er en fordel, at den matematiske model er beskrevet som en overførselsfunktion; derfor udføres Laplace-transformation på (5), hvor I_a isoleres:

$$\frac{d}{dt}I_a = \frac{1}{L_a}V_s - \frac{R_a}{L_a}I_a - \frac{K_v}{L_a}\omega \quad (5)$$

$$s \cdot I_a(s) = \frac{1}{L_a}V_s(s) - \frac{R_a}{L_a}I_a(s) - \frac{K_v}{L_a}\Omega(s) \quad \Leftrightarrow \quad I_a(s) = \frac{V_s(s) - K_v\Omega(s)}{L_a \cdot s + R_a} \quad (18)$$

Ligning (17) Laplace-transformeres, og $\Omega(s)$ isoleres. Herefter indsættes ligning (18):

$$\Omega(s) = \frac{K_t \cdot I_a(s)}{(J + g \cdot J_L) \cdot s + B} \quad \Leftrightarrow \quad \Omega(s) = \frac{K_t \cdot \frac{V_s(s) - K_v\Omega(s)}{L_a \cdot s + R_a}}{(J + g \cdot J_L) \cdot s + B} \quad (19)$$

Ligning (20) nedenfor beskriver overførselsfunktionen fra inputtet V_s til vinkelhastigheden Ω , men da der måles på position, skal funktionen integreres. Den overførselsfunktion, der benyttes i regulatoren, er angivet i ligning (21).

Alle mellemregninger kan ses i *Overførselsfunktion* under det elektroniske bilag.

$$\frac{\Omega(s)}{V_s(s)} = \frac{K_t}{(J \cdot L_a + J_L \cdot L_a) \cdot s^2 + (J \cdot R_a + B \cdot L_a + J_L \cdot R_a) \cdot s + (B \cdot R_a + K_t \cdot K_v)} \quad (20)$$

$$\frac{\Theta(s)}{V_s(s)} = \frac{K_t}{(J \cdot L_a + J_L \cdot L_a) \cdot s^3 + (J \cdot R_a + B \cdot L_a + J_L \cdot R_a) \cdot s^2 + (B \cdot R_a + K_t \cdot K_v) \cdot s} \quad (21)$$

10.3 Design af regulator

Efter den matematiske model er blevet klargjort, kan designprocessen begynde. Først bestemmes åben-sløjfe-polerne og -nulpunkterne. For at bestemme systemets opførsel anvendes en Proportional(P)-regulator. Til design af regulatoren er MATLAB anvendt.

Poler og nulpunkter af systemet

Da målet er at kontrollere vinkelposition, er der integreret og fremkommet en pol i origo. I tabel 8 kan man se en liste over poler og nulpunkter i den integrerede overførselsfunktion:

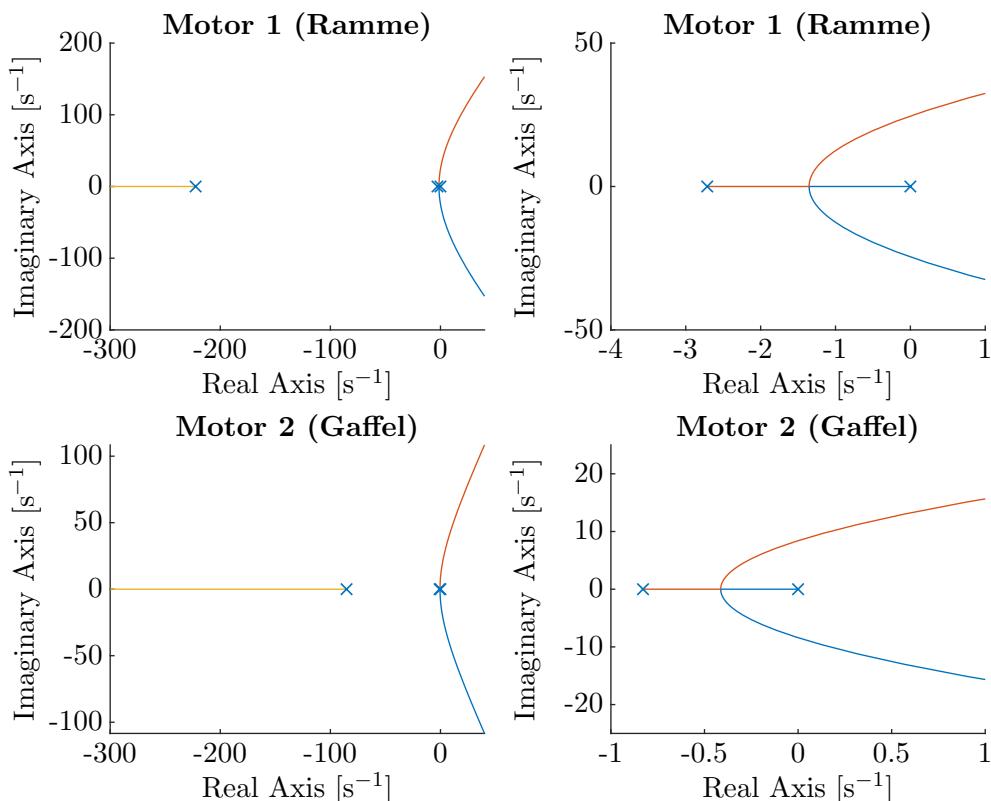
	Motor 1 (Ramme)	Motor 2 (Gaffel)
Poler:	0 -222,4 -2,713	0 -85,31 -0,8285
Nulpunkter:	-	-

Tabel 8: Nulpunkter og poler i systemet

Som man kan se har systemet ingen nulpunkter.

10.3.1 Valg af regulator

Den simpleste regulator er en P-regulator; altså en simpel forstærkning af fejlen ift. referencen. Opførslen af systemet kan studeres vha. åben-sløjfe-polernes bevægelse ift. forstærkningen fra P-regulatoren. For at se bevægelsen af polerne laves et rodkurve-plot. Se figur 29.



Figur 29: Polers bevægelse for systemet med en P-regulator

Som det ses for begge motorer er det ikke muligt at få pol-placeringerne til at ligge til venstre for $s = -4,6$ ved brug af en P-regulator. Det er dog muligt, hvis der indsættes et nulpunkt til venstre for polerne ved $-2,713$ hos motor 1 og $-0,8285$ hos motor 2.

Behovet for et integralled kan undersøges ved at anvende slutværdi-sætningen for systemet, der giver DC-forstærkningen:

$$\lim_{t \rightarrow \infty} (G(t)) = \lim_{s \rightarrow 0} (s \cdot G(s))$$

hvor $G(s)$ er overførselsfunktionen fra ligning (21), og $C(s)$ er P-regulatorens forstærkning.

Tages motor 2 som eksempel, findes DC-forstærkning på lukket-sløjfe-systemet således:

$$\lim_{s \rightarrow 0} \left(s \frac{C(s)G(s)}{1 + C(s)G(s)} \right)$$

$$G_{cl_DC} = \lim_{s \rightarrow 0} \left(s \frac{181,2s^3 + 1561s^2 + 1281s}{s^6 + 172,3s^5 + 7561s^4 + 1236s^3 + 2060s^2 + 1281s} \right) = \frac{0}{1281} = 0$$

Motor 2 har altså en DC-forstærkning på 0, hvilket også er tilfældet for motor 1. Hvis man så kigger på fejlen, når tiden går mod uendelig:

$$e_{ss} = (1 - G_{cl_DC})r = r$$

Dette betyder, at systemet altid vil nå sin reference, hvilket fjerner behovet for et integralled. Derfor fremstilles en proportional-derivative-regulator (PD) for begge motorer. Det forsøges at benytte den samme nulpunktplacering for begge motorer.

10.3.2 PD-regulator

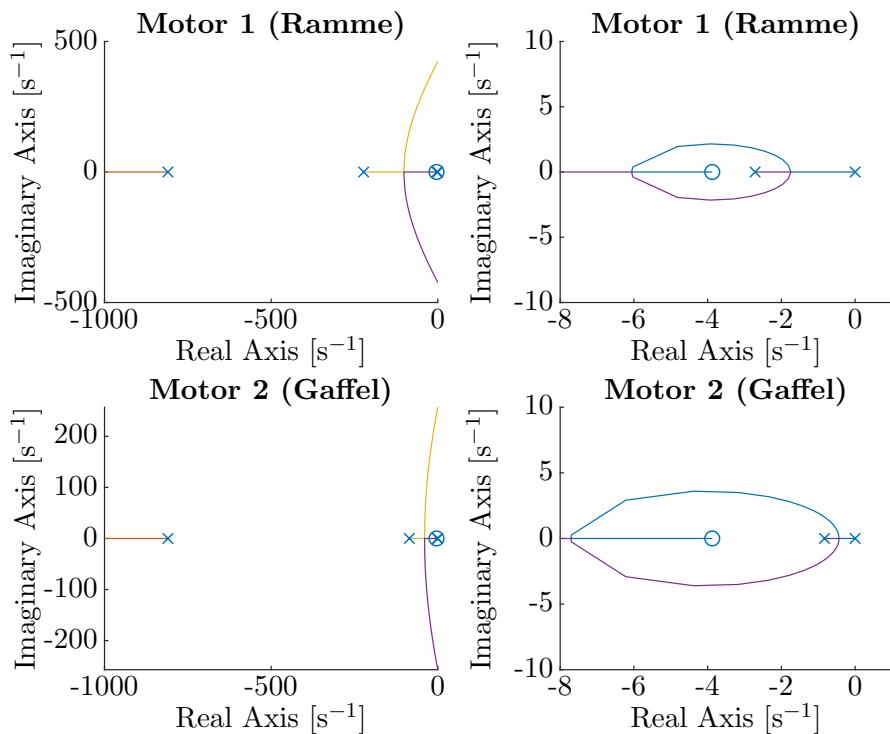
Når en PD-regulator benyttes, har den følgende matematiske form:

$$C(s) = K_P + K_D \cdot s$$

Problemet med denne form er, at den har en uendelig forstærkning ved høje frekvenser på grund af nulpunktet. Derfor skal der benyttes et filter, hvormed formlen er:

$$C(s) = K_P + \frac{K_D \cdot s}{T_F \cdot s + 1}$$

For at filtret ikke påvirker systemet, bliver filtrets poler placeret langt væk fra origo. Filtrets poler er valgt til $T_F = 809,7$. Efter forsøg med forskellige nulpunktplaceringer er det valgt at bruge et nulpunkt ved $s = -1,942$, hvilket svarer til $K_D = 2,1183$. Se figur 30 for de nye polbevægelser for systemet.



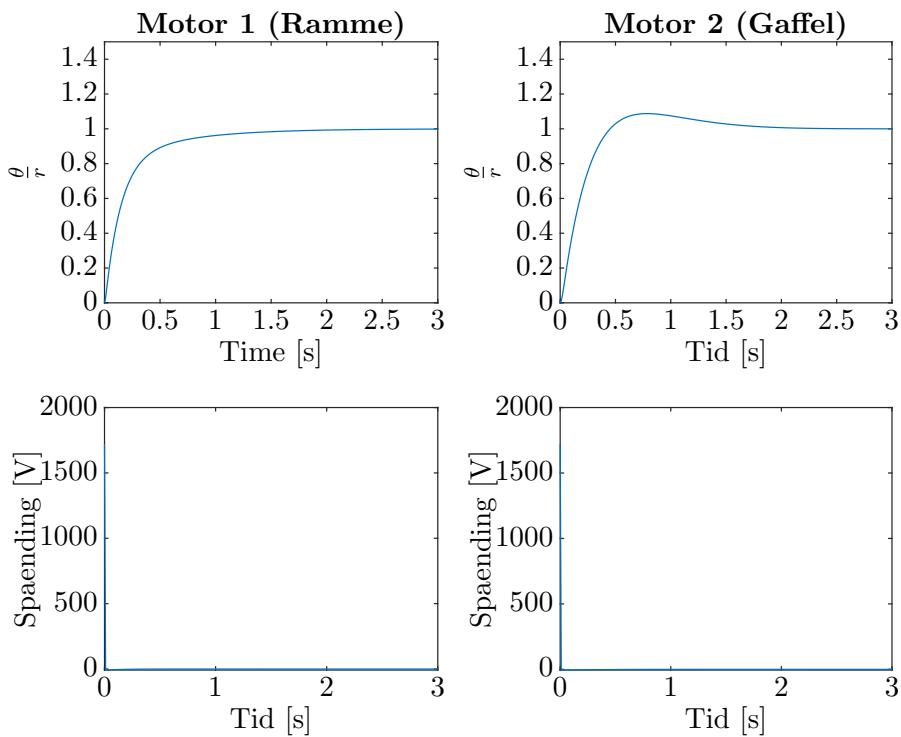
Figur 30: Forskellige nulpunktspladseringers påvirkning på systemets polbevægelser.

Derefter skal en værdi for K_P vælges, så regulatoren giver det ønskede respons.

Motor 1 (Ramme)	4,124
Motor 2 (Gaffel)	4,124

Tabel 9: K_p -værdier for de to motorer.

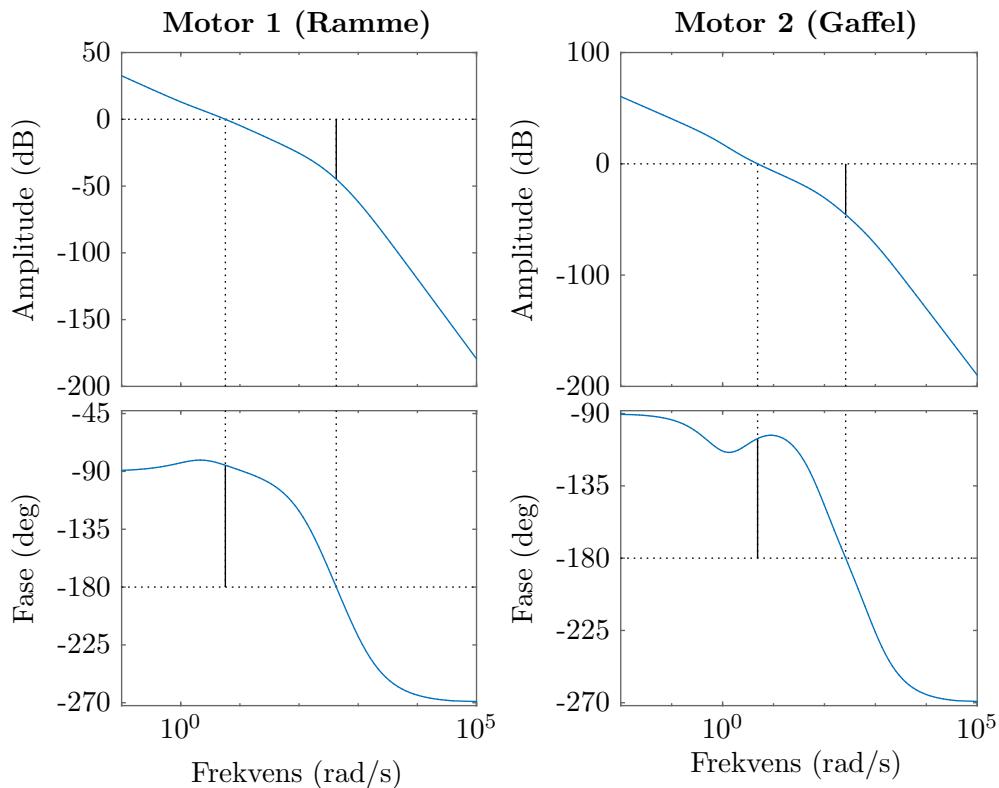
For at verificere forstærkningerne bliver der lavet et steprespons (se figur 31).



Figur 31: Stepresponset for systemet.

Som man kan se på gaffelmotoren på figur 31, har den en indsvingningstid på 1 sekund som ønsket. Regulatoren har dog brug for en spænding på omkring 1600 V, hvilket motoren ikke kan håndtere. Derfor skal der undersøges, om indsvingningstiden overholdes med begrænsning i spændingen, hvilket gøres i afsnit 10.4.

Derudover er der også lidt oversving, men dette forsvinder sandsynligvis når mætning tilføres spændingen. Rammemotoren når sin reference uden problemer, men har dog samme problem med spændingen, som gaffelmotoren.



Figur 32: Bode-plot med angivet fase- og forstærkningsmargin.

Herunder er fase- og forstærkningsmargin angivet. Værdierne overholder de opstillede krav.

	GM [dB]	PM [deg]
Motor 1 (Ramme)	44,82 (425,3 rad/s)	94,797 (5,674 rad/s)
Motor 2 (Gaffel)	45,67 (260,9 rad/s)	74,47 (4,865 rad/s)

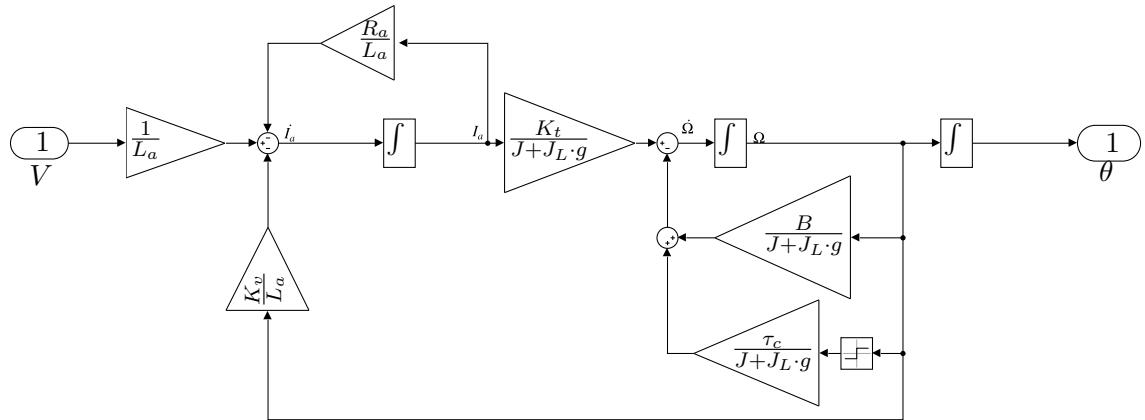
Tabel 10: Forstærknings- og fasemargin til motorerne, angivet i figur 32.

10.4 Simulering af systemet

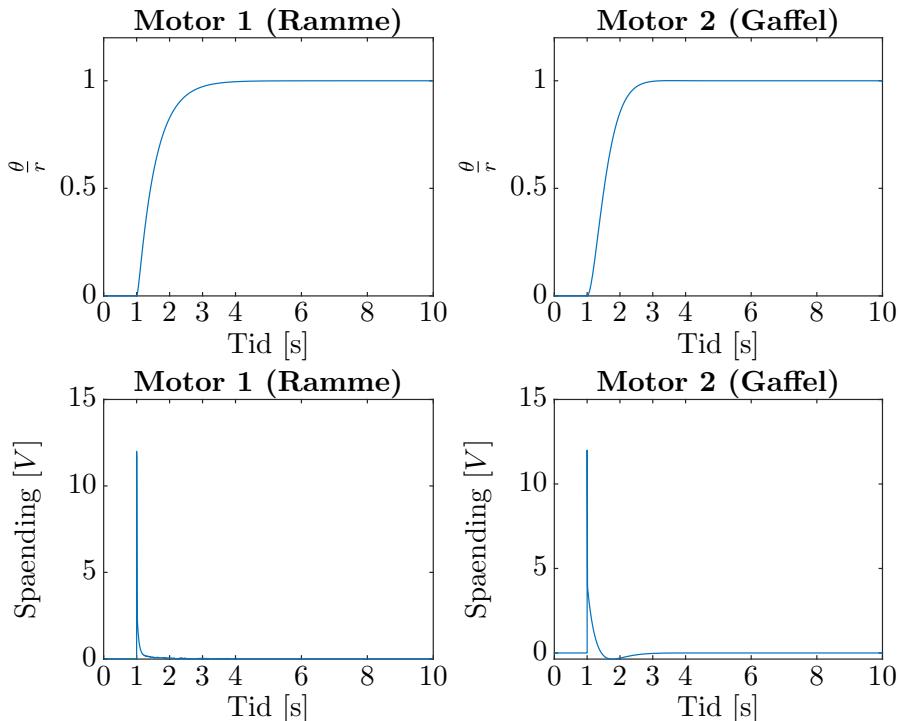
To simuleringer i en MATLAB-udvidelse, **Simulink**, er lavet vha. de indbyggede værktøjer. Den ene er en simpel simulering, som benytter den matematiske model med tilføjet mætning for input-signalet. Den anden er lavet vha. **Simscape Multibody** og dele af den matematiske model.

10.4.1 Simulering af den matematiske model

Figur 33 og 34 viser henholdsvis simuleringen i **Simulink** og tilhørende steprespons:



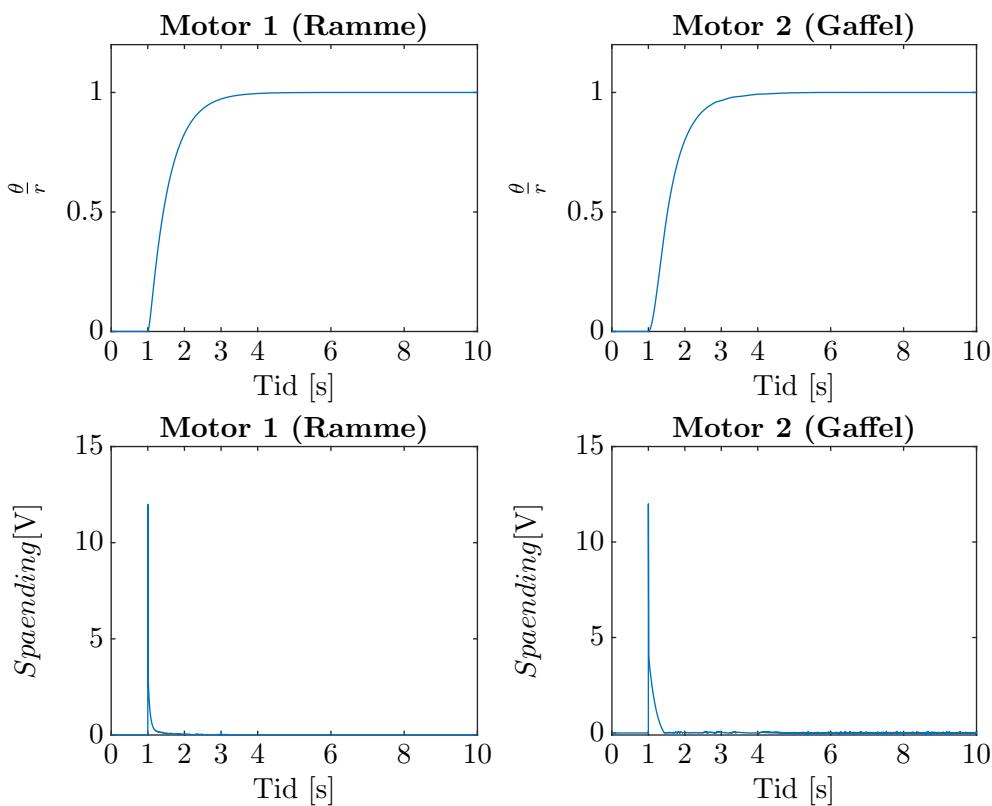
Figur 33: Matematisk model udtrykt i Simulink.



Figur 34: Steprespons for modellen.

Sammenlignes figur 34 og figur 31 kan man se, at regulatoren med begrænsninger for inputtet u opfører sig som ønsket ift. oversving; dog kan det ses, at indsvingningstiden er langsommere end 1 sekund.

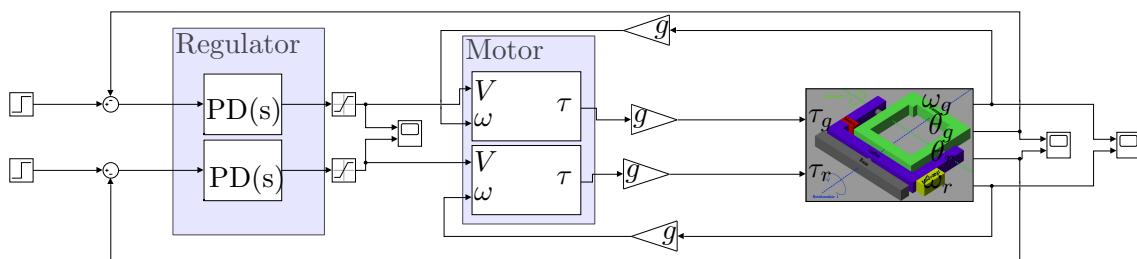
Det undersøges desuden, hvordan regulatoren håndterer den matematiske model med kompensering for den statiske friktion. Herunder ses stepresponserne for de to motorer:



Figur 35: Steprepsonsent med den matematiske model, hvor der kompenseres.

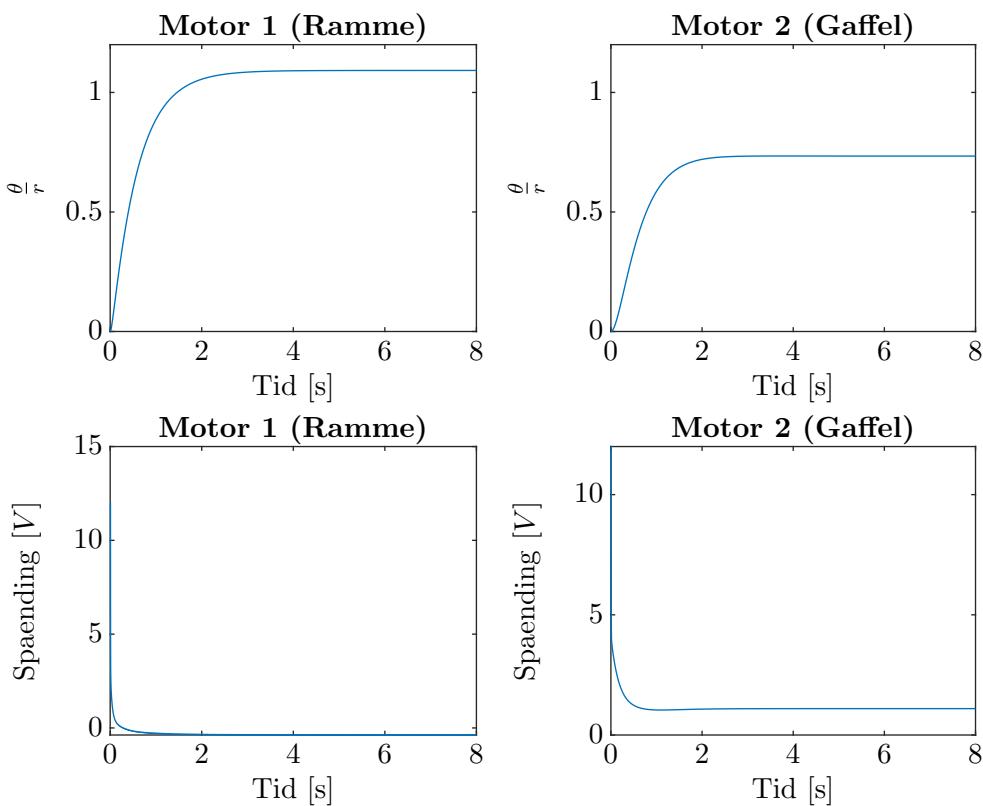
Velvidende at indsvingningstiden er større end 1 sekund, afprøves denne regulator nu i Simscape.

10.4.2 Simscape Multibody-simulering



Figur 36: Kobling mellem **Simulink** og **Simscape**.

Systemet er modelleret i **Simscape**, som er en udvidelse af **Simulink**. Herved opnås endnu en model til at teste, hvordan systemet ideelt set påvirkes af den designede regulator. Fordelen ved **Simscape** er, at her tages højde for tyngdekraftens påvirkning, modsat den matematiske model i **Simulink**.



Figur 37: Stepresponset fra Simscape-modellen, hvor der kompenseres, derudover steppes ved 0 sekunder.

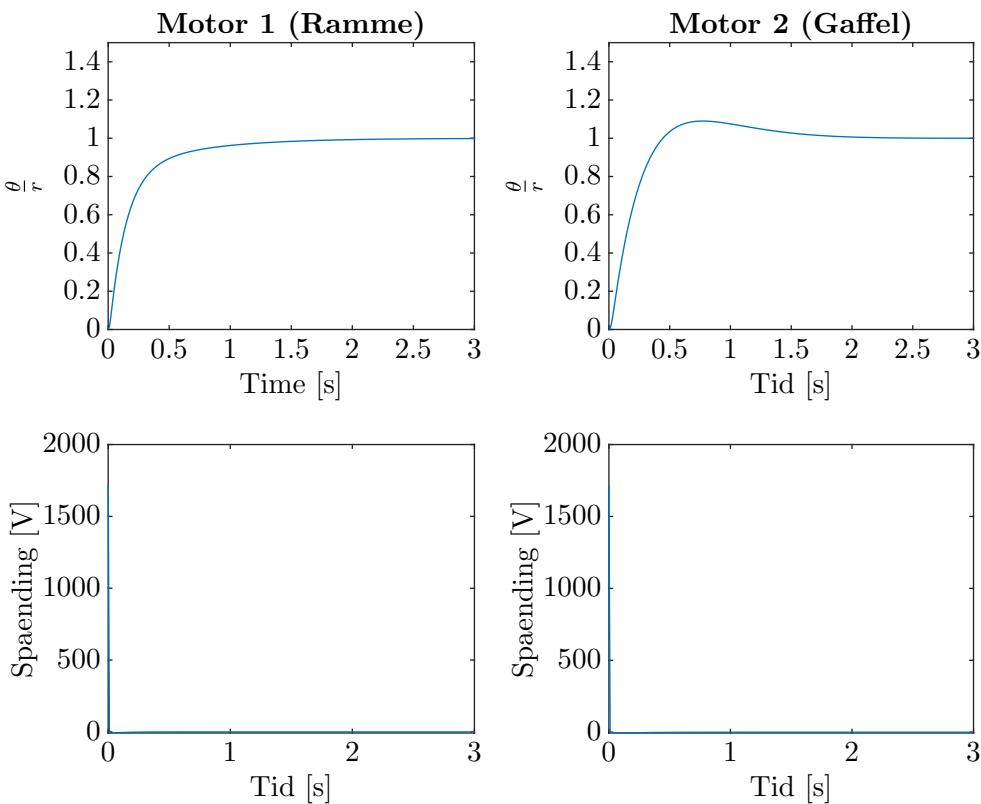
Som det ses på figur 37 er Simscape-modellen ikke identisk med den matematiske model fra Simulink på figur 35. Da begge blot er modeller, undersøges det nu, hvordan pan-tilt systemet i virkeligheden reagerer på samme stepsignal.

10.5 Implementering

Implementering af regulatoren på mikrocontrolleren kræver konvertering fra kontinuert til diskret tidsdomæne, hvorefter den kan implementeres i C. Det er valgt at bruge en periodetid på 10 ms. Før overførselsfunktionen kan implementeres, skal det undersøges, hvorvidt forsinkelsen har nogen indflydelse på responset. En forsinkelse i s-domænet kan udtrykkes som:

$$G_s(s) = \frac{1}{\frac{T}{2}s + 1}$$

hvor T er samplingsperioden.



Figur 38: Stepresponset af det forsinkede system, uden mætning.

Systemet opfører sig ens i figur 38 og figur 31, og derfor kan man der fortsættes med at implementere regulatoren på mikrocontrolleren. Dette ses også på fasemarginen for systemet, da denne beskriver, hvor godt systemet håndterer forstyrrelse i timingen. Begge systemer har et god fasemargin (jf. tabel 10 på side 38), hvilket understøtter figur 38, som viser, at forsinkelsen ikke har nogen betydning.

10.5.1 Implementering i C

To implementeringer af PD-regulatoren blev lavet i C. I den ene implementeres regulatoren som differensligninger. Problemet med denne metode er, at det er svært at ændre på de forskellige forstærkninger. Derfor bruges implementeringsmetoden beskrevet i *Feedback Systems* kapitel 10.5.^[12] I implementeringsmetoden (figur 39) bliver der brugt h for samplingsperiode og b for vægtningen af referencen, hvilket i dette tilfælde er sat til at være $b=1$.

```

kp,b;
ad = Tf/(Tf+h);
bd = kd/(Tf+h);

while (running) {
    r = ADC_in;
    y = enc_in;
    P = kp*(b*r-y);
    D = ad*D-bd*(y-y_old);
    v = P+D;
    u = sat(v,u_low,u_high);
    SET(u);
    y.old = y;
    wait(...); }
```

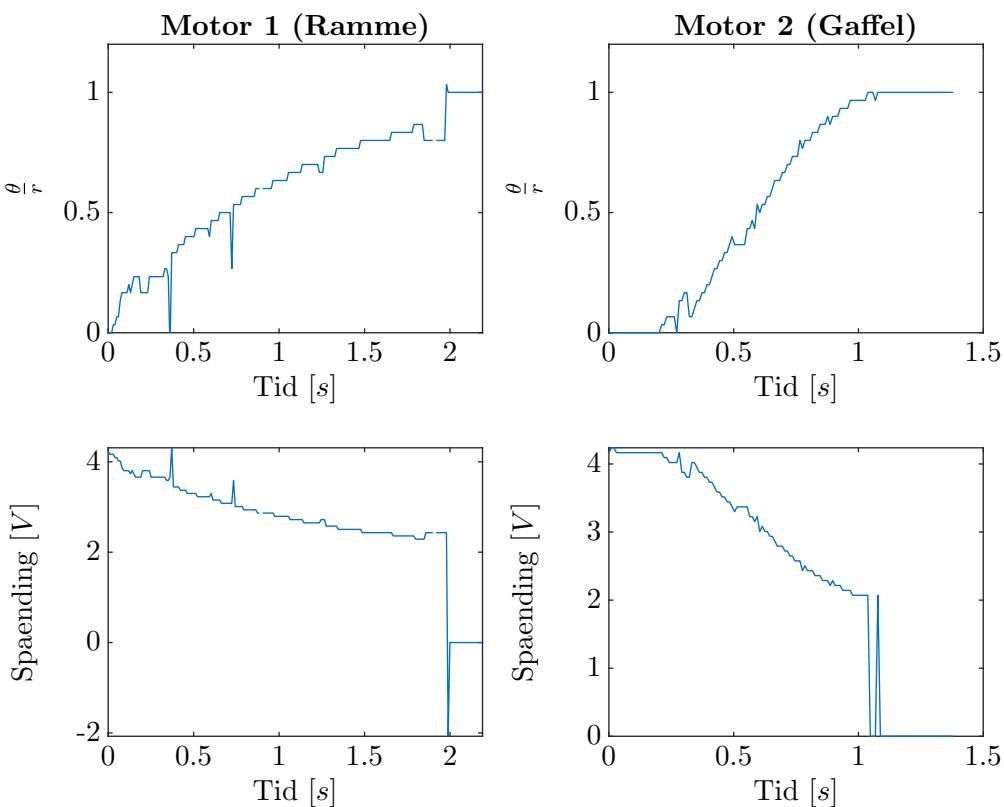
Figur 39: Pseudokode for implementering af en PD-regulator.

Inden sløjfen starter, beregnes regulatorens koefficenter, så der ikke skal bruges tid på dette mens regulatoren kører. Det er dermed også meget nemt at tune regulatorens forstærkninger.

Inde i sløjfen opdateres referencen r , læst fra joysticket, og positionen y læses fra enkoder-værdierne fra FPGA'en. Dernæst opdateres P- og D-leddet, og input før mætning, v , opdateres med de nye værdier. Input u opdateres med v tilføjet mætning og sendes til motorerne. y læses ind i y_old , og der ventes til næste gang regulatoren skal køres.

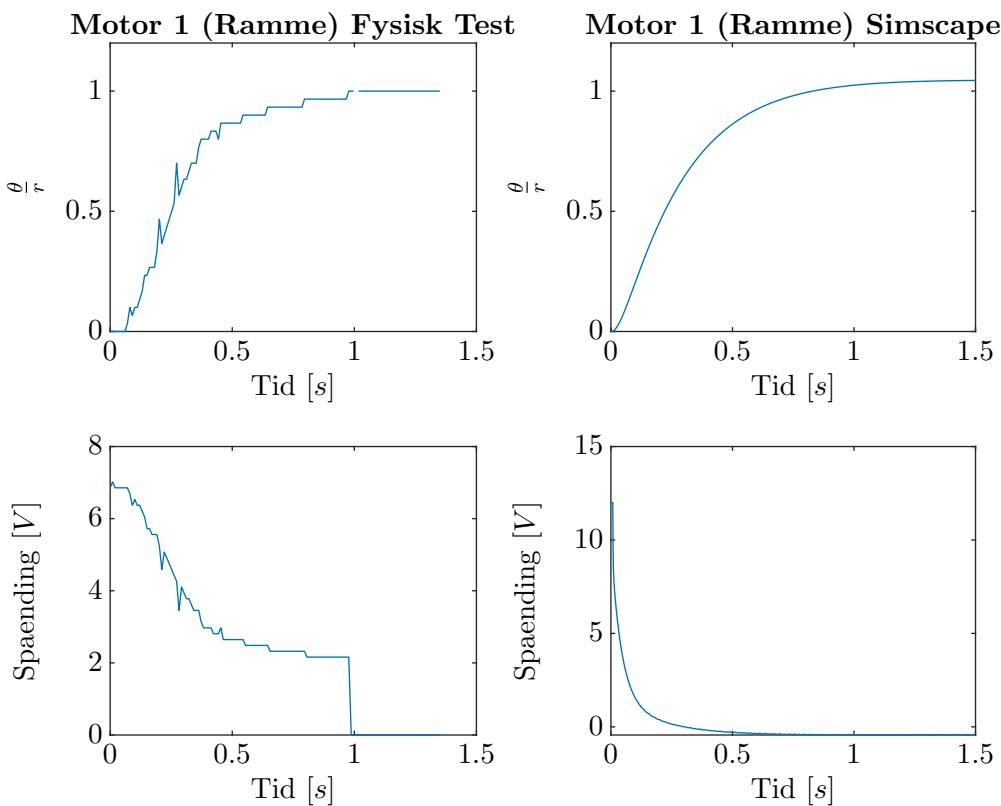
10.6 Test af systemet

Ved første test på systemet var det tydeligt, at pan-tilt systemet havde svært ved at nå referenceværdierne. Derfor blev spændingskompenseringen opjusteret til 2 volt for begge motorer. Herunder ses steprespons for pan-tilt systemet med den nye kompensering:



Figur 40: Steprespons for pan-tilt systemet, med ny kompensering.

Som det ses havde det fysiske system en højere indsvingningstid for rammemotoren. Derfor blev K_P for rammemotoren skruet op fra 4,124 til 9, og forsøget blev gentaget som vist nedenunder:



Figur 41: Nyt steprespons for rammemotoren på pan-tilt systemet, med den nye K_p , og samme værdier i Simscape.

Ud fra dette bekræftes det, at rammemotoren har behov for en ca. dobbelt så høj K_p for at opnå en indsvingningstid på 1 sekund, end gaffelmotoren har. Dette skyldes muligvis, at netop rammen er monteret med to led, modsat gaflen med kun ét led, der hver bidrager med en friktion, som ikke er medregnet i modellerne. Det bemærkes samtidigt, at en simulering af rammemotoren med samme værdier i Simscape (se højre side af figur 41) giver et omtrent korrekt steprespons, dog overstiges referencen på 1. Simscape-stepresponset for gaffelmotoren er stadig uændret fra figur 37, på trods af at stepresponset fra den fysiske gaffelmotor overholder specifikationen med en indsvingningstid på ca. 1 sekund.

11 Konklusion

Applikationen, SPI-protokollen, kommunikationsprotokollen og reguleringsløjfen er programmeret i C med anvendelse af RTCS som styresystem, og implementeret på mikrocontrolleren. Denne kommunikerer med FPGA'en, vis komponenter er programmeret i VHDL. FPGA'en modtager kommandoer til at styre de to DC-motorer efter ønske ved brug af PWM gennem en dobbelt H-bro. Enkodere og hall-sensorer giver feedback omkring motorernes position, hvorfed systemet reguleres herefter. Herunder sammenlignes de fire overordnede fokusområder fra projektbeskrivelsen med den tilhørende konklusion.

Det klassiske labyrintspil skal realiseres vha. pan-tilt systemet

- Et joystick er konstrueret, således at banens orientering kan styres i begge frihedsgrader vha. dette brugerinput. Applikationen til spillet er programmeret til at detektere om spillet er tabt eller vundet vha. et elektrisk kredsløb med ni sensorsystemer og to latches.
- En LCD-skærm viser tiden for den igangværende spillerunde, samt den nuværende rekordtid. Dermed opfyldes kravet om brugerinput og brugerfeedback.

Kommunikation mellem systemets enheder

- SPI-protokollen er oprettet på mikrocontroller og FPGA, hvilket muliggør kommunikation. De opsættes efter master/slave-princippet.
- En kommunikationsprotokol er oprettet for at håndtere beskeder gennem SPI-protokollen. Joysticket sender værdier til regulatoren, der herefter afsender værdier gennem SET-kommandoen til de to motorer. GET-kommandoen anvendes til at anmode om værdier fra enkoderne og hall-sensorerne, hvor der besvares med en REPLY-kommando. REPLY sender motorernes position i grader og hall-sensorernes reset-værdi; dvs. om en motor er tilbage i udgangspositionen.

Motorstyring

- Motorerne styres gennem en dobbelt H-bro, som giver mulighed for styring af begge to samtidigt. H-broen får tilsendt PWM-signaler fra FPGA'en, som regulerer hastigheden på motorerne. Ydermere bestemmes positionen på motorerne gennem enkoderne.

Pan-tilt systemet skal reguleres

- Pan-tilt systemet blev modeleret i Simscape, hvor der blev afprøvet forskellige regulatorer med forskellige forstærkningsværdier. Disse forstærkningsværdier giver ikke

samme steprespons på det virkelige pan-tilt system, som de gør i Simscape. Dette kan skyldes, at det fysiske system ikke er sammensat af friktionsløse led, som Simscape modellen i øjeblikket er.

- Den mest optimale regulator til reguleringssløjfen endte med at være en PD-regulator for at undgå oversving og opnå en indsvingningstid på ca. 1 sekund.
- Pan-tilt systemet kan reguleres ud til en position via en debugger. Desværre kan der på nuværende tidspunkt ikke reguleres til positioner via joysticket.

12 Perspektivering

Dette afsnit omhandler idéer til mulige forbedringer eller alternativer til de valgte løsninger.

Beskeder

SET-kommandoen kan alternativt overtage funktionaliteten for GET-kommandoen, således at FPGA'en hele tiden sender informationer tilbage når en SET-besked modtages, og dermed gøre GET overflødig. På denne måde ved systemet hele tiden, hvad positionen på de to motorer er, også selvom det ikke er ønsket. Det giver muligvis overflødige beskeder med REPLY, men ikke forringelse af funktionalitet. Det giver i derimod færre kommandoer at arbejde med.

Scheduleringsmetode

Valg af scheduleringsmetode blev RTCS. Denne metode er simpel, og anvender de basale funktioner som semaforer, køer og non-preemptive scheduling. Det er ikke nødvendigt for dette projekt at anvende preemptive scheduling. Ønskedes anvendelse af preemptive scheduling, kunne man alternativt anvende FreeRTOS. Denne scheduleringsmetode er mere kompliceret end RTCS; dog er der meget information omkring denne på internettet ift. RTCS, som er skrevet af vores underviser i faget RB-EMD4.

Regulering af position

At få systemet til at regulere positioner via joysticket blev påbegyndt ret sent i projektet. Kompleksiteten af at sammensætte forskellige kodedele, blev muligvis undervurderet; især at regulere position via joystick, da det at regulere PWM-signalet via joysticket og regulatoren virkede separat.

13 Litteraturliste

- [1] Christoffer Sloth: Projektbeskrivelse.
- [2] Run to completion scheduling. Udgivet af Wikipedia. Internetadresse:
https://en.wikipedia.org/wiki/Run_to_completion_scheduling - Besøgt d. 11.05.2019
- [3] Preemptive and Non-Preemptive Scheduling. Udgivet af GeeksforGeeks. Internetadresse: <https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/> - Besøgt d. 11.05.2019
- [4] SPI-bus, Wikipedia. https://da.wikipedia.org/wiki/Serial_Peripheral_Interface-bus. 23.05.2019.
- [5] Serial Peripheral Interface-bus. Udgivet af Wikipedia.
Internetadresse: https://da.wikipedia.org/wiki/Serial_Peripheral_Interface-bus - Besøgt d. 03.05.2019
- [6] Texas Instruments - Tiva TM4C123GH6PM Microcontroller: Datablad.
- [7] Ünsalan, Cem og Bora Tar: Digital System Design with FPGA - Implementation Using Verilog and VHDL. 1. udg. McGraw-Hill Education, 2017. (Bog)
- [8] Encoder, EmbeddedRelated. <https://www.embeddedrelated.com/showarticle/158.php>. 23.05.2019.
- [9] Andersen, Palle og Tom S. Pedersen: Modeldannelse (Kapitel 2). I: BlackBoard, 02.02.2010, s. 14-19
- [10] DC Motor Construction. Electrical Engineering Portal. <https://electrical-engineering-portal.com/4-types-of-dc-motors-and-their-characteristics>. 23.05.2019.
- [11] Afsnit 10.5: Halliday: Principles of Physics. 10. udg. Wiley, 2013
- [12] Afsnit 10.5: Åström, Karl Johan og Richard M. Murray: Feedback Systems: An Introduction for Scientists and Engineers, 2008.

14 Oversigt over elektronisk bilag

Journaler:

- Filer til journal
- Bestemmelse af motor-konstanter

Kode og diagrammer:

- Tiva: Header- og C-filer
- FPGA: VHDL-filer
- Taskdiagram

Øvrige bilag:

- Inertimomenter for systemet
- Simulink og Simscape
- Overførselsfunktion
- Projektoplæg