

ELE 217 Prosjekt MyJet
Gruppe 2

av

Morten Jacobsen, Jan Egil Fredriksen og Mohammed Al-Rawi

i

ELE217-G 20V
Mikrokontrollere og styresystemer

Veileddet av Geir Jevne og Ken Henry Andersen

Fakultet for teknologi og realfag
Universitetet i Agder

Grimstad, Januar 2020

Sammendrag

Denne rapporten beskriver designe løsningen på en drone styring. Som skal styres fra en BLE app og uart. Det er mulig å lese ut data og kontrolleren en styre enhet. Løsningen har også med seg strømmåling og styring ved hjelp av akselerometerdata og gyroskop data fra MPU6050. Det blir brukt både SPI og TWI protokoll for å overføre data til slaveenheten.

Obligatorisk gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemiddler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

1.	Vi erklærer herved at vår besvarelse er vårt eget arbeid, og at vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	Ja
2.	Vi erklærer videre at denne besvarelsen: <ul style="list-style-type: none">• Ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands.• Ikke refererer til andres arbeid uten at det er oppgitt.• Ikke refererer til eget tidligere arbeid uten at det er oppgitt.• Har alle referansene oppgitt i litteraturlisten.• Ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	Ja
3.	Vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	Ja
4.	Vi er kjent med at alle innleverte oppgaver kan bli plagiatskontrollert.	Ja
5.	Vi er kjent med at Universitetet i Agder vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk.	Ja
6.	Vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider.	Ja
7.	Vi har i flertall blitt enige om at innsatsen innad i gruppen er merkbart forskjellig og ønsker dermed å vurderes individuelt. Ordinært vurderes alle deltakere i prosjektet samlet.	Nei

Publiseringsavtale

Fullmakt til elektronisk publisering av oppgaven Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).

Oppgaver som er unntatt offentlighet eller taushetsbelagt/konfidensiell vil ikke bli publisert.

Vi gir herved Universitetet i Agder en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	Ja
Er oppgaven båndlagt (konfidensiell)?	Nei
Er oppgaven unntatt offentlighet?	Nei

Forord

Denne rapporten av utviklingsprosessen, er en del av ELE217-G 19H Mikrokontrollere og styresystemer prosjektet.

Gruppen vil takke deres veiledere Universitetslektor Geir Jevne og Universitetslektor Ken Henry Andersen, som foreleser ved Universitetet i Agder for riktig veiledning gjennom prosjektperioden.

Grimstad

19. desember 2019

Morten Jacobsen, Jan Egil Fredriksen og Mohammed Al-Rawi.

Forkortelser

Forkortelse	Hele uttrykket
BLE	Bluetooth Low Energy
I2C	Inter-Integrated Circui
MPU	
PID	Propotional Integral Derivative
PWM	Pulse width modulation
SDK	Software Development Kit
SoC	System on a Chip
SPI	Serial Peripheral Interface

Tabell 1: Tabell av forkortelser

Innhold

1 Innledning	1
2 Teknisk bakgrunn / teoretisk bakgrunn	2
2.1 nRF52832	2
2.2 MPU6050	2
2.3 Inter-Integrated Circuit (I2C)	3
2.4 Serial Peripheral Interface (SPI)	3
2.5 Blåtann Lav Energi (BLE)	4
2.6 PWM Pulse width modulation	4
2.7 PID (Propotional Integral Derivative) controller	4
3 Testing og validering	6
3.1 Validering av størmålings krets	6
3.2 SPI	6
3.3 PWM	7
4 Diskusjon	8
4.1 Kommunikasjon med MPU6050	8
4.2 PID	10
4.3 PWM signal til LED	10
4.4 SPI protokoll	12
4.5 RTOS Løsning	13
4.6 BLE kommunikasjon	15
4.7 Gjennomsnitts strøm måling (ADC)	16
5 Konklusjon	21
5.1 Prosess	21
5.2 Produkt	21
5.3 Utfordringer	21
5.4 Videre arbeid	21
Tillegg A Timeplan og Tidsestimat	23
Tillegg B SW og HW SDL av kontrollenhet	25
Tillegg C SW og HW SDL av styreenhet	27
Tillegg D SDL for nRF52 1 styreenheten	28
Tillegg E SDL for Main i nRF52 Kontrollenhet	30
Tillegg F SDL for i2c register read og write i nRF52 Kontrollenhet	31
Tillegg G SDL for accelerometer og gyroskope read nRF52 Kontrollenhet	33
Tillegg H SDL for MPU6050 Calibration nRF52 Kontrollenhet	35
Tillegg I SDL for MPU6050 tråd i nRF52 Kontrollenhet	37
Tillegg J SDL for MPU Angle Math	38
Tillegg K SDL for PID i nRF52 2 kontrollenhet	39

Tillegg L SDL for PWM init nRF52 Kontrollenhet	40
Tillegg M SDL for PWM i nRF52 2 kontrollenhet	41
Tillegg N SDL for SPI master funksjon	42
Tillegg O SDL for SPI master event handler	43
Tillegg P SDL for SPI master init	43
Tillegg Q SDL for SPI slave Init	44
Tillegg R SDL for SPI slave tråd	45
Tillegg S SDL for SPI slave event handler	46

Figurer

1	Hoved Blokk Diagram	1
2	nRF52832 SoC[10]	2
3	MPU-60X0 blokk diagram[5]	3
4	PID closed loop	4
5	PID ARM funksjon	5
6	Live oppsett analog krets	6
7	Test design av kretsen på et bread board	6
8	SPI-testing	7
9	PWM module	10
10	PWM sequence	11
11	PWM motor speed regulation	11
12	PWM init	12
13	PWM bidrag	12
14	PWM sekvens oppdatering	12
15	Data som blir sendt og mottatt over SPI	13
16	SPI protocol	13
17	Oversiktsbilde, tråder i RTOS	14
18	MailQ	15
19	for loop i main	16
20	Switch Case kodden for NUS hendelse	16
21	Strøm målings krets	19
22	Simulert inngangs spenning ved forskjellige pulsvidde fra PID	20

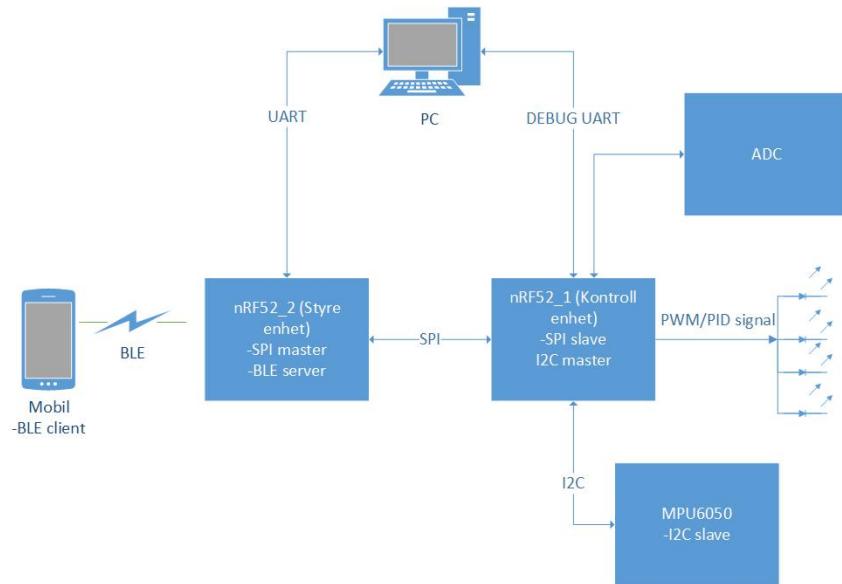
Tabeller

1	Tabell av forkortelser	iii
---	----------------------------------	-----

Listinger

1 Innledning

Prosjektet har som mål å kunne lage et styresystem til en drone. Til å gjøre dette har vi fått noen ramme faktorer i prosjekt oppgave som at det skal benyttes 2 stk nRF52832DK kort skal kommunisere over SPI buss. Og en MPU6050 som det skal hentes data utfra. Vi har velgt å kalle nRF52832 kortene for «styreenhets» og «kontrollenhet».



Figur 1: Hoved Blokk Diagram

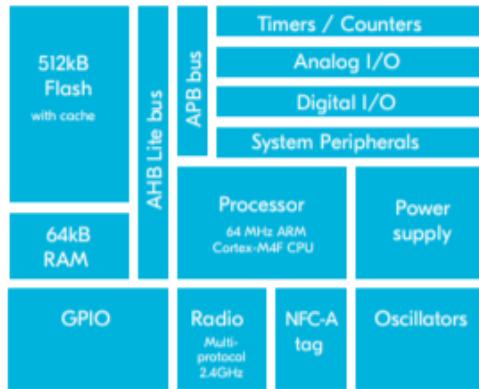
Kontrollenhet skal hente akselerometerdata og gyroskop data fra MPU6050 over TWI kommunikasjon. Som vist i figur1. Daten skal omregnes til vinkler og styrer motorer i dronen som er simulert med 4 lysdioder. Vinkelen kan settes av styreenheten via mobil. Denne enheten måler og beregner en verdi for endring i strøm til hver motor, eller diode som i dette prosjektet. Strømmen som måles skal måles og digitaliseres med ADC'en til nrF52 kortet. All data som går mellom styreenheten og kontrollenheten overføres med SPI protokollen.

Styreheten skal hente og gi data over SPI til kontrollenheten. Og skal kunne gi hentet data til mobilen når den etterspørres av mobilen som vist i figur1. Samt gi ønsket vinkel på drone fra mobil til kontrollenhet. Kommunikasjonen med mobilen skjer via BLE protokollen. Enhetene skal kunne kommunisere til en komputer over uart og skal kunne gi den samme dataen til uart som den kan gi til mobilen.

2 Teknisk bakgrunn / teoretisk bakgrunn

2.1 nRF52832

nRF52832 chipen er en SoC (System on a Chip)[10], slike enheter inneholder alt en mikrokontroller trenger på en chip. Figur2 viser hva denne SoC'en inneholder, slik som minne, innganger, utganger, tellerer, prosessor og radio. Fordelen med slike systemer er kort tidsforsinkelse og billig å produserer og egner seg meget godt til embedded system utvikling.



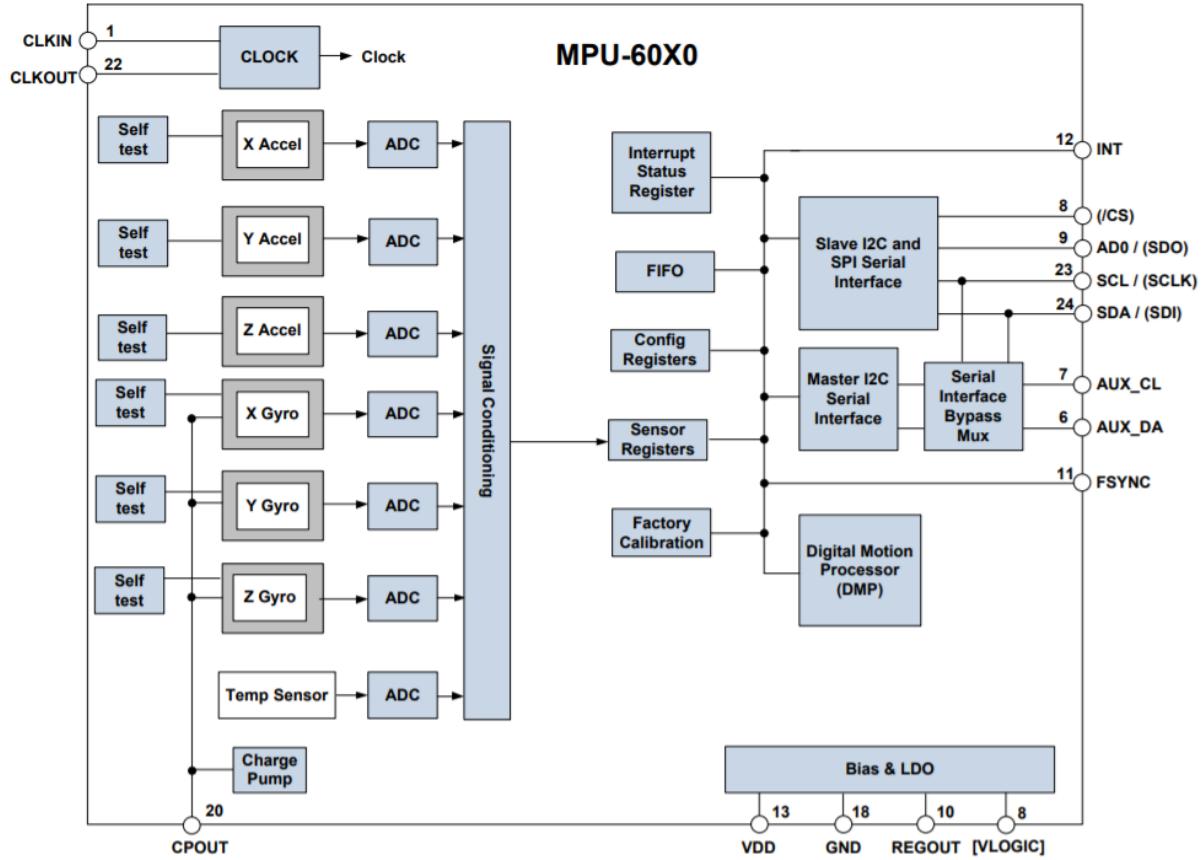
Figur 2: nRF52832 SoC[10]

Denne chipen sitter på et utvikler kort som er laget av Nordic semiconductors. Den er ment brukt til utprøving design av kontroll og styresystemer for utviklere[10]. Det gjør at disse utvikler kortene inneholder flere hardware enheter og funksjonaliteter en vårt design egentlig trenger, da de kan brukes til et bredt spekter av applikasjoner.

2.2 MPU6050

Invensense MPU-6050 er relativt billig med tanke på at det både er et akselerometer, gyroskop og temperatur sensor. MPU-6050 består av et MEMS akselerometer samt et MEMS gyroskop i en enkel chip. Den skryter på seg å være meget nøyaktig[12] ved å ha 16-bits analog til digital konvertering[5] av 3 gyro og 3 akselerometer kanaler.

Konverteringen skjer parallelt i det respektive X, Y og Z planet for begge enhetene. MPU'en benytter en I2C-buss for å kommunisere med nRF52 Kontrollenhet. MPU-6050 er satt opp som I2C-slave og NRF52 kontrollenhet som I2C-master for kommunikasjon mellom dem på I2C bussen.



Figur 3: MPU-60X0 blokk diagram[5]

Akselerometeret bruker flere piezoelektriske sensor som registrerer endring i akselerasjon. Denne kan settes opp med ulik oppløsning som settes i registeret ACCEL_CONFIG [MPU6050·reg] fra $\pm 2g$ til $\pm 16g$. Når gyroskopet roteres om en av sanseaksene, forårsaker Coriolis-effekten en vibrasjon som blir oppdaget ved en kapasitiv avleddning. Følsomheten på denne kan settes mellom $\pm 250^\circ$ til $\pm 2000^\circ/\text{sekund}$ (dps) i GYRO_CONFIG[MPU6050·reg] registeret.

2.3 Inter-Integrated Circuit (I2C)

MPU6050 er satt opp som I2C-slave med presatt adressene 0x68 og 0x69[MPU6050·reg]. Adressen til MPU-6050 styres ved å sette AD0 pinnen på MPU'en lav eller høy. I2C-Master er nRF52 Kontrollenhett. Den kommuniserer over I2C bussen vha. protokollen Nordic Semiconductors kaller TWI (Two wire interface). Gjennom denne protokollen styres 3 pinner: CS (chipselect) - som velger hvilken slave enhets som det snakkes med, SDL (serial data) og SCL (serial clock) synkroniserer data overføringen mellom enheten.

2.4 Serial Peripheral Interface (SPI)

Kommunikasjonen mellom nRF52 kortene skjer over en SPI protokoll. Denne protokollen er en synkron serie kommunikasjon. Med fire linjer, en til data overføring fra master til slave (MOSI), en til data fra slave til master(MISO), en klokke linje for synkronisering (SCLK) og en linje til å velge hvilken slave som det snakkes med (SS). nRF52832 enheten som er koblet til MPU6050 er slaven.

2.5 Blåtann Lav Energi (BLE)

Bruktes på kommunikasjon mellom mobil enheten og styre enheten nRF52_2 kortet. Protokollen er en to veis radio kommunikasjons protokoll[17, s. 3]. Den brukes på enheter som kun trenger kort rekkevidde og sender i et lisensfritt ISM-band (Industrial, Scientific and Medical Band) mellom 2,402 GHz og 2,480 GHz[7, s. 200]. De kan brukes over hele verden uten spesielle tillatelser. Nordic sin nRF52832 chipen støtter Bluetooth 5.1 og BLE[10].

Mobilens er client/central enheten og styre enheten er serveren/peripheral enhet i dette designet. Oversikten på dette kan ses i figur 1.

2.6 PWM Pulse width modulation

Pulsbreddemodulasjon (PWM), eller pulsvarighetsmodulasjon (PDM) er en metode for å redusere den gjennomsnittlige kraften som leveres av et elektrisk signal, ved effektivt å hugge det opp i diskrete deler. Gjennomsnittsverdien på spenning (og strøm) sendt til lasten styres ved å slå bryteren mellom tilførsel, og last av og på med en bestemt hastighet. Jo lengre bryteren er på sammenlignet med av-perioder, desto høyere blir den totale strømmen som tilføres belastningen.[18]

2.7 PID (Propotional Integral Derivative) controller

PID-kontrollerens er den vanligste kontrolleren i industrien, algoritmen til denne teknikken brukes for å gi et bedre nøyaktig signal angående ønsket verdi og for å spare energi ved å gi bare det systemet trenger. I denne transførfunksjonen med lukket sløyfe, er inngangssignalet til PID-kontrolleren den ønskede (refereanse) verdien blir subtrahert av feedback-signalet fra utgangen som kalles feil, se figure[4]. Kontrolleren mottar nåværende roll og pitch vinkler fra MPU6050 og etter en funksjonskall[5] får vi ut bidraget til ledene (motorene).

Algoritmen:

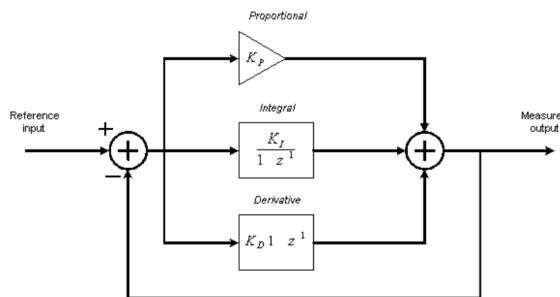
$$y[n] = y[n - 1] + A0 * x[n] + A1 * x[n - 1] + A2 * x[n - 2]$$

$$A0 = K_p + K_i + K_d$$

$$A1 = (-K_p) - (2 * K_d)$$

$$A2 = K_d$$

der K_p er proposjonal konstant, K_i er Integral konstant og K_d er Derivativ konstant.[8]



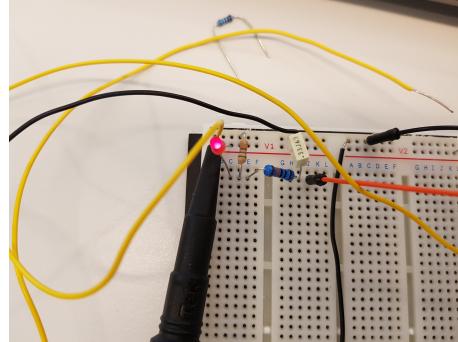
Figur 4: PID closed loop

```
__STATIC_FORCEINLINE float32_t arm_pid_f32(arm_pid_instance_f32 * S,  
                                         float32_t          in  
)  
  
Parameters  
  [in,out] S is an instance of the floating-point PID Control structure  
  [in]     in input sample to process  
Returns  
  processed output sample.
```

Figur 5: PID ARM funksjon

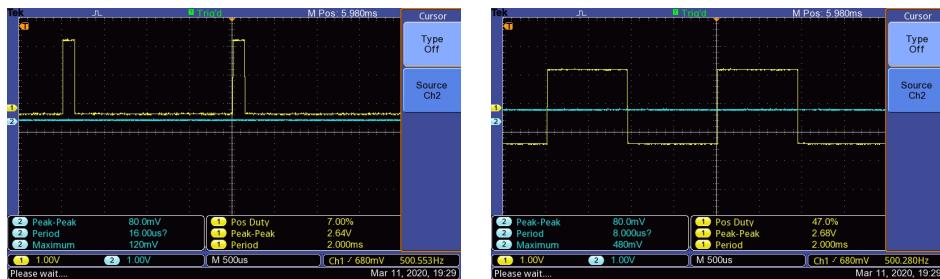
3 Testing og validering

3.1 Validering av størmålings krets



Figur 6: Live oppsett analog krets

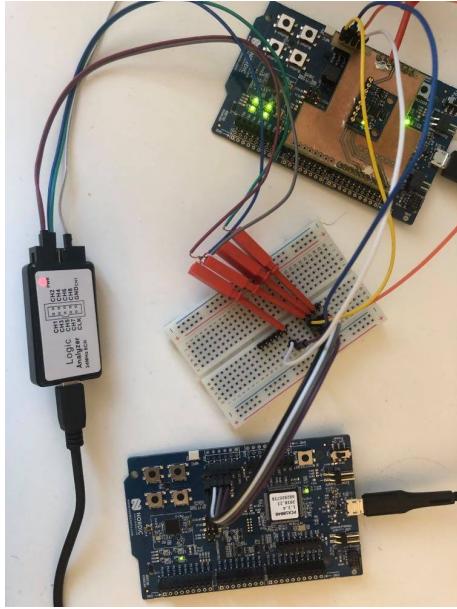
I figuren 7 målte vi på PWM pulsen og utgangen av gjennomsnitts strøm kretsen og for å få validert at denne krets koplingen funger slik den var ment. Den analoge kretsen ble koplet opp på et bread board vist i figur 6. Brukte et osciloskop og nRF52 kort med opplastet kode for PWM. Resultatet av denne testen kan vi se at puls bredden er direkte proporsjonal med utgangen. og kretsen fungerer slik vi har tenkt.



Figur 7: Test design av kretsen på et bread board

3.2 SPI

SPI transføren ble testet med en logic analysator. Den funker som den skal.



Figur 8: SPI-testing

3.3 PWM

Siden “motorene” ble emulert med de fire innebygde ledene på nRF52 kortet, ble det testet med ledene. Se video på Youtube [14].

4 Diskusjon

4.1 Kommunikasjon med MPU6050

Da denne oppgaven ble gitt hadde gruppen lite erfaring med å jobbe med en MPU6050. I starten ble oppgaven planlagt ut fra InvenSense sitt Datasheet[5] samt Register-Map[**MPU6050’reg**] hvor det ble forklart hvilke bruksområder og registre som var nødvendig for å løse denne oppgaven. Et praktisk eksempel løst ved hjelp av Arduino ble ervervet fra Joop Brokking[4].

Kommunikasjonen mellom MPU6050 og nRF52 Kontrollenhet, foregår gjennom I2C/TWI, hvor MPU6050 er satt opp som I2C-slave og Kontrollenheten I2C master.

Identiteten til MPU6050 «WHO_AM_I» er 0x68 når pinne AD0 er lav og 0x69 når høy [**MPU6050’reg**]. For å initialisere MPU'en må vi «våkne den opp». Dette løses ved å skrive ZERO (0) til PWR_MGMT_1 registeret [**MPU6050’reg**].

For å få ut rådata til akselerometeret og gyroskopet leser vi ut data fra X/Y/Z 8bits registre for respektive H og L bits[**MPU6050’reg**]. Da disse dataene er delt opp i høye- og lave-bits kombinerer vi disse sammen til en Uin16_t. Rådataene vi får ut er på 2'ers kompliment form, og vil på denne måten beholde sin verdi.

$$int16_t(x/y/z_data = (data[0/2/4] << 8)|x/y/z_data[1/3/5]) \quad (1)$$

Før vi stoler på funksjonen til å lese ut data fra akselerometer og gyroskopet , ønsker vi å kalibrere sensorene for å minimere eventuelle avviksfeil. Dette ble gjort gjennom en Inertial Measurement Unit Error calibration funksjon (calculate_IMU_error). Denne funksjonen leser av verdiene fra både akselerometer og gyroskopet 2000 ganger og finner gjennomsnittsavviket for begge, og lagrer dette avviket som respektivt Akselerasjons- og gyroskop-avvik. Dette avviket trekkes fra rådataen i lesefunksjonen i MPU6050 tråden.

Opprinnelig ba oppgaven oss om å kun bruke akselerometer delen av MPU6050. Vår gruppe valgte derimot å kombinere akselerometeret og gyroskopet for en mer presis måling samt tilrettelegge for utvidelse av funksjonalitet. Implementasjonen ble gjennomført ved å kombinere en tilbakekobling fra en porsjon av akselerometerverdien sammen med gyroskopverdien for å motvirke «drift» [3]. Dette gjøres ved hjelp av et komplementær filter hvor vi kombinerer de tre gebevegelsene fra akselerometeret med de raske bevegelsene til gyroskopet. Akselerometeret alene gir en god indikasjon på orienteringen i statiske forhold, og gyroskopet gir en god indikasjon på tilt i dynamiske forhold. Ideen bak dette er å sende akselerometersignalene gjennom et lavpass filter og gyroskopsignalet gjennom et høypass filter, og kombinere sluttresultatene for en endelig verdi.

Påfølgende steg var behandling av rådataen vi hadde lest ut fra akselerometeret og gyroskopet. Akselerometeret måler forskjellen mellom lineær-akselerasjon i akselerometerets referanseramme og jordens gravitasjonsfelt vektor[13, s. 2]. En positiv verdi tilsier her en økning i fart og akselerasjon, og en negativ verdi tilsier en nedgang. Data ut fra gyroskopet gir forteller oss om rotasjon om X/Y/Z aksene i °/s (dps).[5, s. 25]. Da vi ønsker å vise nåværende vinkel, er vi avhengige av å fjerne tidenparameteren fra gyroskop verdien. Dette løses ved å multiplisere gyroskopverdien

med differansen i tid mellom hver lesing (dt). Tidtaker funksjonen ble skrevet av Ken Henry Andersen og publisert på UIA Canvas for oss å bruke.[1].

Deretter behandles rådataen gjennom matematiske kalkulasjoner med bakgrunn i «Aerospace Rotation Sequence» og «Euler's Angles». X/Y/Z blir heretter referert til som «Roll»/«Pitch»/«Yaw».

Gyro_pitch settes til summen av rådataen fra gyroskopets X-register dividert med gyroskopets oppløsning og til slutt multiplisert med dt. Operasjonen er tilsvarende for Y registeret.

$$GyroPitch/Roll = \left(\frac{X/Y_{GYR}}{GYR_RESOLUTION} \right) * dt \quad (2)$$

Deretter setter vi betingelsen: om IMU har rotert om ledelinjen (yaw, z-akse), legger vi Roll vinkelen til Pitch vinkelen, og motsatt trekker vi fra Pitch fra Roll.

$$Gyro_Pitch += Gyro_Roll * \sin\left(\frac{Z_{GYR}}{GYRO_RESOLUTION}\right) * \left(\frac{\pi}{180}\right) \quad (3)$$

$$Gyro_Roll -= Gyro_Pitch * \sin\left(\frac{Z_{GYR}}{GYRO_RESOLUTION}\right) * \left(\frac{\pi}{180}\right) \quad (4)$$

For å gjøre vinkel beregninger på rådata fra akselerometeret, må vi først regne ut den totale akselerometer vektoren. Dette gjøres ved å ta kvadratrotten av X/Y/Z verdiene.

$$Accelerometer_total_vector = \sqrt{(X_{ACC}^2 + Y_{ACC}^2 + Z_{ACC}^2)} \quad (5)$$

Deretter regner vi ut akselerometer Pitch og Roll vinkler:

$$Acc_Pitch = \arcsin\left(\frac{Y_{ACC}}{Acc_total_vector}\right) * \left(\frac{180}{\pi}\right) \quad (6)$$

$$Acc_Roll = \arcsin\left(\frac{X_{ACC}}{Acc_total_vector}\right) * \left(\frac{180}{\pi}\right) \quad (7)$$

Første gang løkken kjører setter vi Gyr_Pitch = Acc_Pitch, og Gyr_Roll = Acc_Roll. Deretter setter vi et IMU flagg som indikerer at løkken nå har kjørt sin første runde.

Så implementerer vi et komplementærfilter som tar 90% av Gyroskopverdien og kombinerer dette med 10% av Akselerometer verdien, dette for å forhindre «drift» fra gyroskopet.

$$Angle_Pitch_Output = Gyr_Pitch * 0.9 + Acc_Pitch * 0.1 \quad (8)$$

$$Angle_Roll_Output = Gyr_roll * 0.9 + Acc_pitch * 0.1 \quad (9)$$

4.2 PID

Med transferfunksjon som er gitt av CMSIS DSP software library [8], fant vi ut at k_i har ikke så mye effekt på algoritmen. I tillegg må k_p være større enn $2 \times k_d$. Da ble det valgt følgende:

$$k_p = 21.9$$

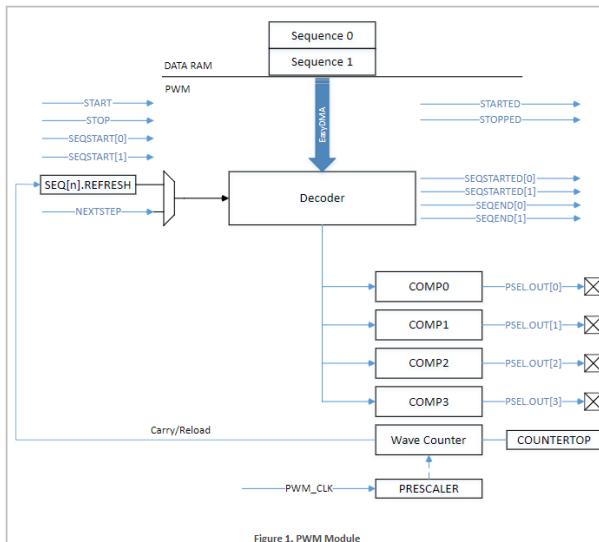
$$k_i = 0.00015$$

$$k_d = 10$$

I utgangspunktet så referansepunktet er å holde "dronen" stabil, dvs. referanse er "0". Hvis det kommer noen ønsket vinkel fra bruker så blir referansen satt til den verdien som er mottatt fra "SPI slave event handler" via Mail. Bidraget er satt til å være maks 400 siden alle "motorene" starter med $1500 \mu\text{S}$. Retur verdien fra PID ARM funksjonen[5] sendes til PWM tråden.

4.3 PWM signal til LED

nRF52832 har 3 PWM hardware blokker som gjør det mulig å generere puls breddemodulerete signaler på GPIO. Hver PWM module kan gi opptil 4 PWM-kanaler med individuell frekvensstyring i grupper på opptil fire kanaler. Innebygd dekoder og EasyDMA funksjoner gjør det mulig å manipulere PWM duty-cycle'ene uten CPU-inngrep. Vilkårlige duty-cycle sekvenser leses fra Data RAM og kan lenkes for å implementere ping-pong-bufering eller gjentas i komplekse løkker.



Figur 9: PWM module

Først definerer vi en sekvens for PWM duty-cycle'ne:

```

static nrf_drv_pwm_t m_pwm0 = NRF_DRV_PWM_INSTANCE(0);
static nrf_pwm_values_individual_t m_PWM_seq_values = {0, 0, 0, 0, 0};
static nrf_pwm_sequence_t const m_PWM_seq =
{
    {
        .values.p_individual      = &m_PWM_seq_values,
        .length                   = NRF_PWM_VALUES_LENGTH(m_PWM_seq_values),
        .repeats                  = 0,
        .end_delay                = 0
    };
};

uint16_t PWM_output[4] = {0};

```

Figur 10: PWM sequence

I vår applikasjon ble det valgt PWM-basefrekvens på 1 MHz, ESC må ha minst et PWM signal med 2,041 ms periodetid ($f = 490$ Hz). Wave counteren teller opp med 490 Hz klokka frekvens, for å finne top verdien til telleren brukte vi:

$$T_{PWM(up)} = T_{PWM(clk)} \times COUNTERTOP \quad (10)$$

$$T_{PWM(up)} = 2,041\text{ms}$$

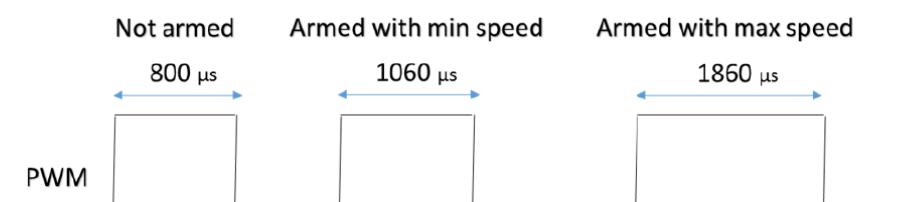
$T_{PWM(clk)} = 1\mu\text{s}$ da blir $COUNTERTOP = 2041$ [15].

Vi valgte å oppdatere duty-cycle'en individuelt til hver kanal. Utgangen av hver PWM kanal er koblet til en led på nRF52832.

$$D = \frac{T_{on}}{T}, \quad T = T_{on} + T_{off} \quad (11)$$

Duty-cyclen til PWM bør ikke være større enn $1860\mu\text{s}$, se figur 11 [16].

Figure 17. PWM input signal for motor speed regulation



Step 13. The motor starts to rotate between the minimum and maximum speed.

Figur 11: PWM motor speed regulation

Etter å ha initialisert PWM hardware blokka, blir den startet med en funksjonskall:

```

void PWM_init(void)
{
    nrf_drv_pwm_config_t const config0 =
    {
        .output_pins =
    {
        Motor_1 | NRF_DRV_PWM_PIN_INVERTED,           // channel 0
        Motor_2 | NRF_DRV_PWM_PIN_INVERTED,           // channel 1
        Motor_3 | NRF_DRV_PWM_PIN_INVERTED,           // channel 2
        Motor_4 | NRF_DRV_PWM_PIN_INVERTED,           // channel 3
    },
        .irq_priority = APP_IRQ_PRIORITY_LOWEST,
        .base_clock  = NRF_PWM_CLK_1MHz,                // 1 MHz klokke
        .count_mode  = NRF_PWM_MODE_UP,
        .top_value   = PWM_DEFAULT_CONFIG_TOP_VALUE,   // 2041
        .load_mode   = NRF_PWM_LOAD_INDIVIDUAL,         // Individuell load.
        .step_mode   = NRF_PWM_STEP_AUTO
    };

    nrf_drv_pwm_init(&m_pwm0, &config0, NULL);
    (void)nrf_drv_pwm_simple_playback(&m_pwm0, &m_PWM_seq, 1, NRF_DRV_PWM_FLAG_LOOP);
}

```

Figur 12: PWM init

PWM tråden venter på bidraget fra PID tråden. Etter å ha fått mailen kalkuleres det den endelige bidraget til hver av “motorene”:

```

PWM_output[0] = RUN_PWM + From_PID->Roll - From_PID->Pitch ;//FRONT RIGHT CCW + & - LED 1 3 1 X
PWM_output[1] = RUN_PWM - From_PID->Roll - From_PID->Pitch ;//REAR RIGHT CW - & - LED 2 4 Y<--|--
PWM_output[2] = RUN_PWM + From_PID->Roll + From_PID->Pitch ;//FRONT LEFT CCW + & + LED 3 2
PWM_output[3] = RUN_PWM - From_PID->Roll + From_PID->Pitch ;//REAR LEFT CW - & + LED 4 1

```

Figur 13: PWM bidrag

Så blir PWM-COMP registrene oppdatert:

```

m_PWM_seq_values.channel_0 = PWM_output[0];
m_PWM_seq_values.channel_1 = PWM_output[1];
m_PWM_seq_values.channel_2 = PWM_output[2];
m_PWM_seq_values.channel_3 = PWM_output[3];

```

Figur 14: PWM sekvens oppdatering

4.4 SPI protokoll

Det var to mulige løsninger for å kommunisere mellom masteren og slaven, den ene var å polle, og den andre var å implementere en ekstra pinne for å sende et interrupt fra slaven til masteren hver gang det kom en ny data fra IMU'en (MPU-6050). I vår applikasjon ble det enighet om at SPI masteren skulle polle hver 100 millisekunder, så masteren får oppdatert dataene sine 10 ganger i sekundet. Hver gang masteren forespør data, sender slaven all data i en strukt [figure 15]. SPI'en kjøres på MODE0, SCLK er lav når den er inaktiv og data samples på stigende flanke og sendes på fallende flanke av SCLK. Det er MSB som sendes først. SPI Frekvensen er 4 MHz.

```

typedef struct
{
    int16_t x_data;
    int16_t y_data;
    int16_t x_angle;
    int16_t y_angle;
    uint16_t m1_thrust;
    uint16_t m2_thrust;
    uint16_t m3_thrust;
    uint16_t m4_thrust;
    int16_t m1_current;
    int16_t m2_current;
    int16_t m3_current;
    int16_t m4_current;
}RX_Mas_struct;

```

(a) RX Master struct

```

typedef struct
{
    int16_t x_angle_desired;
    int16_t y_angle_desired;
}TX_Mas_Data_struct;

```

(b) TX Master struct

```

typedef struct
{
    int16_t x_angle_desired;
    int16_t y_angle_desired;
}RX_SLAVE_Data_struct;

```

(c) RX Slave struct

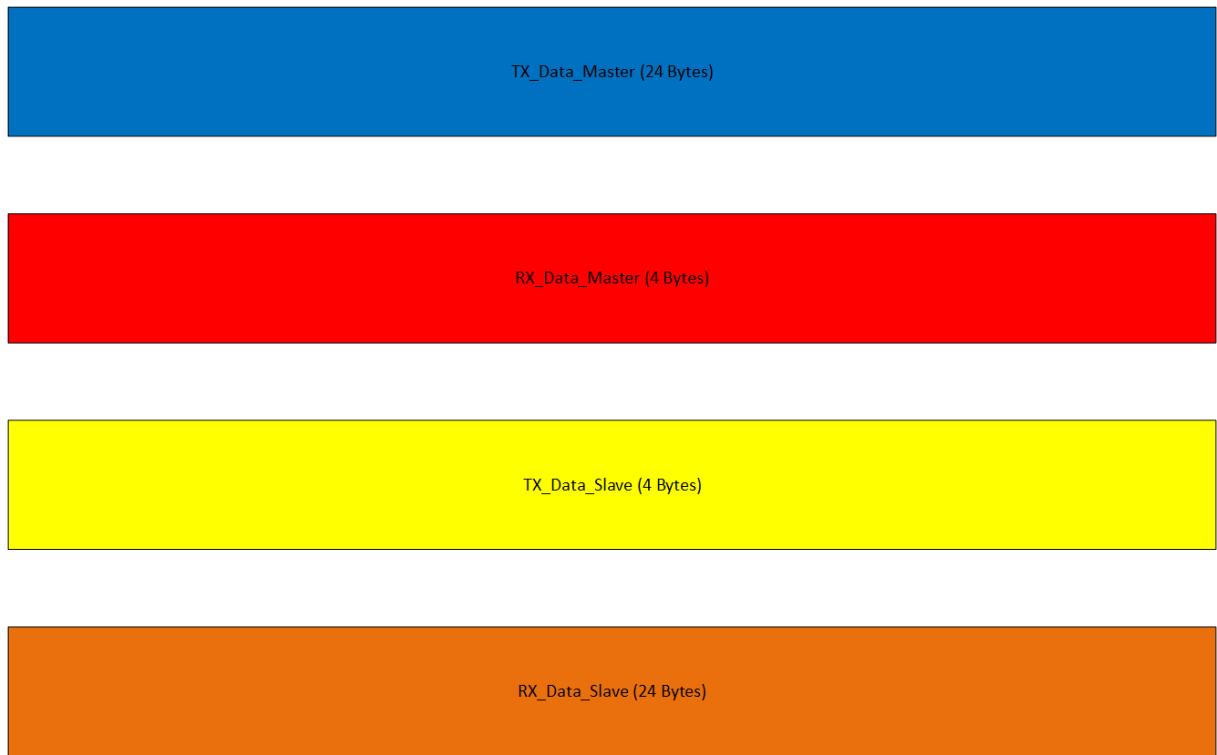
```

typedef struct
{
    int16_t x_data;
    int16_t y_data;
    int16_t x_angle;
    int16_t y_angle;
    uint16_t m1_thrust;
    uint16_t m2_thrust;
    uint16_t m3_thrust;
    uint16_t m4_thrust;
    int16_t m1_current;
    int16_t m2_current;
    int16_t m3_current;
    int16_t m4_current;
}TX_Slave_struct;

```

(d) TX Slave struct

Figur 15: Data som blir sendt og mottat over SPI



Figur 16: SPI protocol

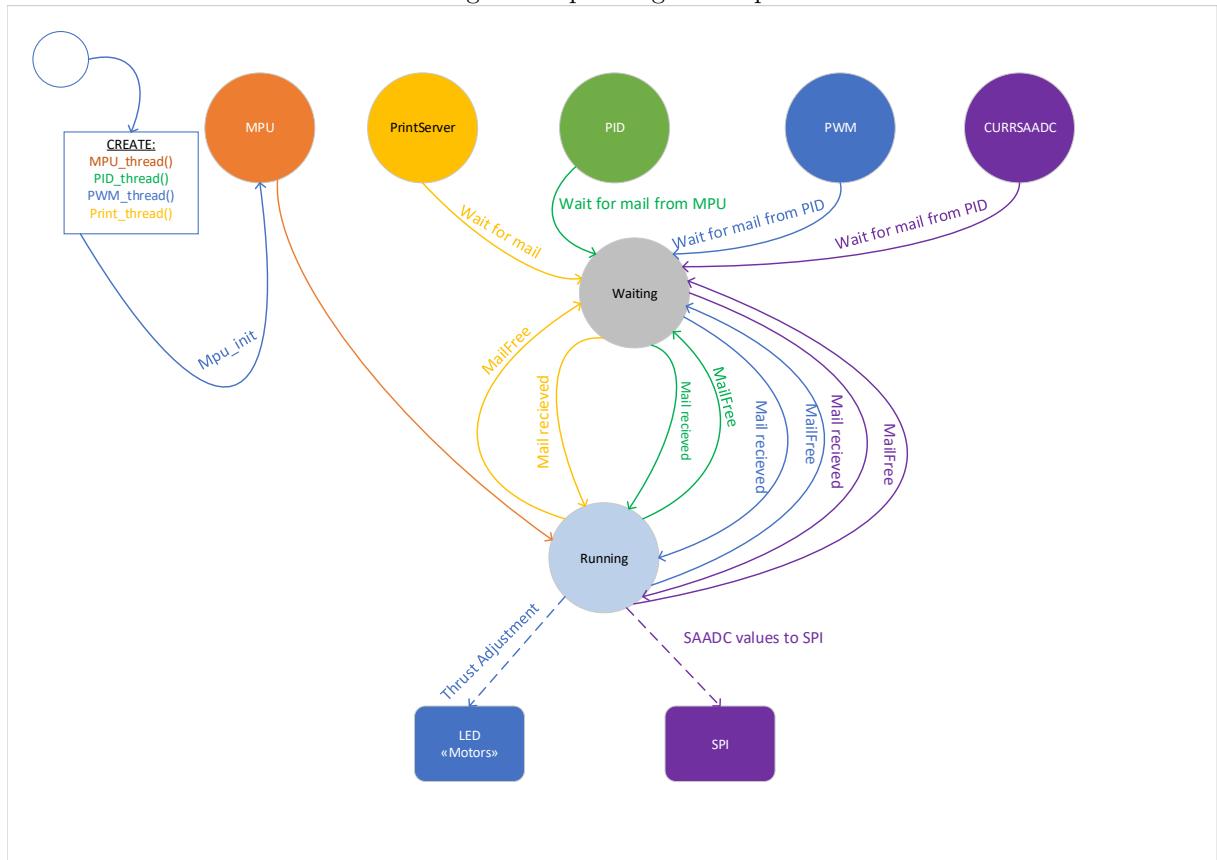
4.5 RTOS Løsning

For å sikre en god flyt i prosjektet ble det tidlig bestemt å implementere CMSIS-RTOS («Cortex Microcontroller Software Interface Standard Real Time Operating System»). CMSIS RTOS tilrettelegger for å implementere en standardisering for å gjøre programvaren flyttbar mellom ulike mikrokontrollere og utviklingsmiljø, likeledes forenkler den prosessen med å gjenbruke programvare fra andre prosjekter og forenkler integrasjon av programvare fra tredjeparts leverandører.

RTOS er nyttig når man har et prosjekt med flere oppgaver som skal kjøre i tilsvrelatende parallel og tilrettelegger for eksekvering av bestemte oppgaver når gitte hendelser inntreffer, med et fokus på raskt bytte av oppgaver - såkalt «Context switching».

I vårt prosjekt ble RTOS benyttet i forbindelse med trådene for PID, PWM, SPIS, CURRSAADC og Printserver(printe til UART).

diagram v4.pdf diagram v4.pdf



Figur 17: Oversiktsbilde, tråder i RTOS

Det ble brukt Mail Queue til kommunikasjon mellom trådene, se figur[18]. I første omgang leses verdiene ut fra MPU6050, deretter blir resultatet kalkulert til vinkler i MPU tråden før resultatet sendes til PID for å beregne pådrag eller reduksjon i motorene, før resultatet av dette blir sendt til PWM tråden for å generere en visuell representasjon av PIDens resultat. All data fra PWM sendes til slutt til CURRSAADC tråden. Der blir strøm målingene lagt til strukten og sendt til SPIS tråden for å overføres til SPI Master enheten.

Mail Queue mellom trådene



Figur 18: MailQ

4.6 BLE kommunikasjon

For å sette opp en BLE kommunikasjon mellom en mobil og nRF52_2 styreenhet. Ble det brukte eksempel kode fra **Nordic semiconductor**. Eksemplet vi så på og tok som utgangs punkt i for å lage vår egen BLE kommunikasjon, var eksempel for en peripheral enhet kaldt ble_app_uart. Dette eksemplet er hentet fra nRF5 SDK versjon 15.3.0[9].

Når vi hadde blitt kjent med dette eksemplet valgte vi å få løftet dette eksemplet opp i path's strukturen lik med koden vi har lavet for SPI kommunikasjonen. Dette for å enkle sammen syingen av forskjellige filer og at det ble enklere for GIT å følge di forskjellige delene i prosjektet ligger i samme mappe og med lik mappestruktur. For å gjøre dette måtte vi endre includpath's og finne hvilke c filer som dette bygget trenger for å bygge og kopiere med disse inn i mappestrukturen.

Videre valgte vi å splitte denne koden opp slik at ikke alt av kode ligger i en c fil, man at main er ryddig og oversiktlig med kun funksjonskall til initialisering og for selve *for*

loopen. Det ble laget en egen c fil hvor alle funksjonen til BLE ble lagt, med en tilhørende h fil.

Så ble det laget en for loop i main som figur 19 viser. Denne sjekker for om det har vært timer event eller NUS event. Når timer event kommer skal det gå en SPI kommunikasjon. Sendes en struct med data over til kontroll enheten. Tilbake får en en mengde data som lagres i en struct som telefonen kan hente data fra.

```

for (;;)
{
    idle_state_handle();
    if (SPI_TIMER_EVENT)
    {
        SpiMaster();
        SPI_TIMER_EVENT = false;
        TX_DATA.data_from_ble[0] = 0;
    }
    if (NUS_DATA_EVENT)
    {
        do_ble_cmd();
        NUS_DATA_EVENT = false;
    }
}

```

Figur 19: for loop i main

Når NUS event trigges i for loop'en så startes en funksjon som inneholder en switch case som velger handling utfra hvilken kommando som kommer fra telefon kommunikasjonene.

```

switch (TX_DATA.data_from_ble[0])
{
    //if cmd from phone get angle
    case 1:
        sprintf(send_buff, 4,"%d%d", RX_DATA.x_angle, RX_DATA.y_angle);
        err_code = ble_nus_data_send(&m_nus, (uint8_t*)send_buff, &length, m_conn_handle);
        break;
    //if cmd from phone get accelerometer data
    case 2:
        sprintf(send_buff, 4,"%d%d", RX_DATA.x_data, RX_DATA.y_data);
        err_code = ble_nus_data_send(&m_nus, (uint8_t*)send_buff, &length, m_conn_handle);
        break;
    //if cmd from phone get thrust data
    case 3:
        sprintf(send_buff, 4,"%d%d%d%d", RX_DATA.m1_thrust, RX_DATA.m2_thrust, RX_DATA.m3_thrust, RX_DATA.m4_thrust);
        err_code = ble_nus_data_send(&m_nus, (uint8_t*)send_buff, &length, m_conn_handle);
        break;
    //if cmd from phone get current data
    case 4:
        sprintf(send_buff, 4,"%d%d%d%d", RX_DATA.m1_current, RX_DATA.m2_current, RX_DATA.m3_current, RX_DATA.m4_current);
        err_code = ble_nus_data_send(&m_nus, (uint8_t*)send_buff, &length, m_conn_handle);
        break;
    //if cmd from phone sett angel
    case 5:
        TX_DATA.x_angle_desired= atoi(&TX_DATA.data_from_ble[1]);
        TX_DATA.y_angle_desired= atoi(&TX_DATA.data_from_ble[3]);
        SpiMaster();
        break;
    default:
        //Do nothing.
        break;
}

```

Figur 20: Switch Case kodden for NUS hendelse

4.7 Gjennomsnitts strøm måling (ADC)

Strømmålingen skal sendes til SPI TX buffer for å kunne sendes til styreenheten som lagrer denne verdien og kan sende den til mobil enheten eller ut på uart ved forespørsel. Som illustrert i figur1

For å realisere dette valgte vi å bruke den inne bygget SAADC på nRF52 utvilkert kortet. Denne blir brukt til å digitalisere et analogt spennings signal som er proporsjonalt med strømmen i lysdioden, til et digitalt signal som kan konverteres til en forståelig verdi. Denne spenningen kommer fra en shunt motstand og et lavpass filter,

som lager en gjennomsnittsspenning av dette pulsede signalet fra PWM enehetn. Dette kan ses i figur21.

Måten koden løser dette på er ved at en timer får SAADC'en til digitaliserer verdien på de fire kanalene. Dette skaper at en interrupt handler setter et signal hos SADDC tråden som behandler verdiene fra ADC'en. Denne tråden våkner da fra idle state når det er dens tur og omformer disse verdien til verdier som settes i TX bufferet til SPIS.

Som en et utgangspunkt benyttes Nordic stackens peripheral/saadc eksempel kode. Denne går i one-shot mode hvor en enkelt kanal samples. I dette prosjektet skal det samples på 4 kanaler siden det er 4 lysdioder på kontrollenheten. Moden som kan gjøre dette kalles scan-mode. Enheten går automatisk over i scan-mode hvis flere kanaler er muliggjort[11].

Scan-mode funger slik; når den er aktivert og en *SAMPLE task* blir mottatt; så samples det en gang på alle aktiverete kanaler. Når buffere er fylt opp så sendes et EVENT_END[11] signal, vi bruker EVENT_DONE signal siden buffere er like stort som 4 kanalen. Så buffere er fult ved samme tid som konverterings tasken er ferdig. Hadde buffere vært dobbel størrelse av antall kanaler ville vi fått to DONE signaler før buffere var fult. Mens END signalet vil kun kommet en gang. Så i dette designet kan vi bruke begge. (Vi benytter DONE)

For å initialisere en ADC kanal må denne settes opp med flere parametre som vist i punkt listen under. Fleste parametre er predefinert. Så en trenger bare å sjekke om noen er satt feil i forhold oppgavens behov. De to første punkteen handler om Resistor Ladder som brukes til å løfte eller senke inngangen mot VDD eller GND[11]. Vi har valgt å ikke bruke ladderen da flyter inngangen. Og målingen blir tatt i forhold til interne GND. Er alt satt i predefinerte strukten.

- I. resistor_p
- II. resistor_n
- III. gain
- IV. reference
- V. acq_time
- VI. mode
- VII. burst
- VIII. pin_p
- IX. pin_n

For å sette GAIN verdien må en finne maksimal signal verdi, som det skal måles på. Ut fra dokumentasjonen kan PWM gå opp til 3.3volt. Da må vi bruke en Gain på 1/6. Da er maksimalt spenningskilden som det kan måles på være 3.6v. Utregningen kan ses i likningen under[11].Denne parameteren er også predefinert til dette.

$$InngangsVoltageRange = \frac{(0.6v)}{\left(\frac{1}{6}\right)} = 3.6v \quad (12)$$

Referanse kan velges mellom intern og VDD. I vårt design settes denne til intern referanse på 0.6v, denne er mere stabil en den eksterne på VDD og drifter mindre.

Akkvisisjon tid er hvor lenge svitsjen til kondensatoren holdes inne slik at kondensatoren kan lades opp. Denne tiden blir bestemt av input resistansen. Høy resistans fra måle objektet lengre tid holdes svitsjen inne. I vårt design er det 100kohm som kan ses i figur 21. Da kan vi finne fra Table 2. Acquisition time i nordic dokumentasjonen at denne tiden kan være 15us[11]. I linjen under kan det ses en eksempel kode på at en endrer denne verdien til ønsket verdi.

```
channel_0_config.acq_time = NRF_SAADC_ACQTIME_15uS;
```

Med mode kan en velge mellom nornal og differensial måling. for å måle differensial tilfører en to signaler og måler differansen mellom di. I normal mode kortsluttes den negative inngangen til jord når aktivert. Dette er den moden vi ønsker å måle med og er alt blitt definert i forhånds defineringen.

Burst mode er at SAADC tar to raske målinger og tar gjennom snitte av dette og legger i RAM[11]. Vi ønsker ikke denne aktiv så den er deaktivert. dette er alt definert i den forhånds definerte struckten.

Med pin_p og n velger en om det skal være positiv eller negativ input pinne signalet skal måles på[11]. Her ønsker vi kun å måle på den positive input pinnen og disable den negative.

For å sette opp en timer som trigger saadc målinger hver 100ms. Endrer vi kun eksempel koden som vit under med å sette inn 100 i stede fro den verdien som var satt. Med denne koden under setets det opp en teller som teller og blir resatt når ønsker verdi er nådd. Dette trigger så et event signal via PPI som går til SAADC'en som tar en måling på hver analog input som er konfigurert.

```
uint32_t ticks = nrf_drv_timer_ms_to_ticks(&m_timer, 100);  
nrf_drv_timer_extended_compare(&m_timer,  
    NRF_TIMER_CC_CHANNEL0,  
    ticks,  
    NRF_TIMER_SHORT_COMPARE0_CLEAR_MASK,  
    false);
```

SDK config må settes opp slik at den gjør en 12bits konverttereing og dette gir en oppløsning på $3.6v/2^{12}=0.396v$.

Så ender SAADC en et event done signal når bufferet er fullt som trigger NVIC som har med alle interrupt handlere, den starter callback funksjonen til SAADC. Denne skal inneholde kode som konverterer data'en fra de analoge inngangene og legge det fra bufferet inn i struckten som sendes over SPI bussen til styre enheten.

```
void Saadc$Callback(nrf_drv_saadc_evt_t const * p_event){  
    if (p_event-> type == NRF_DRV_SAADC_EVT_DONE)  
    {  
        ret_code_t err_code;
```

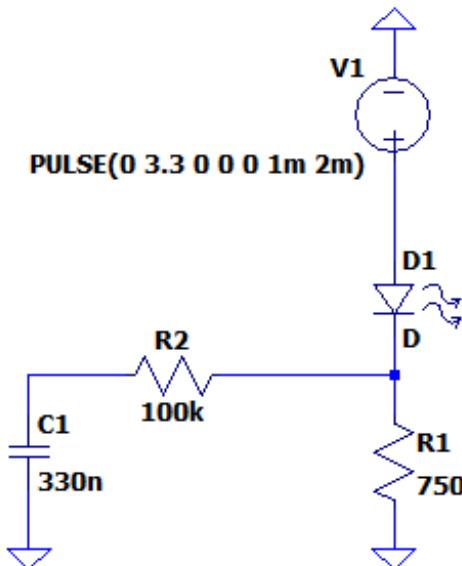
```

err_code = nrf_drv_saadc_buffer_convert(p_event->data.done.p_buffer,
SAMPLES_IN_BUFFER);
for (int i = 0; i < SAMPLES_IN_BUFFER; i++)
{
    m_current[i] = (p_event->data.done.p_buffer[i]*3600)/4096;
}
}

```

Som en ser i koden over så henter callback funksjonen event data pekeren som peker hvor bufferet ligger i esay dma minne. Så sjekker koden om saadc event done er satt. Hvis denne er satt leser den data'en ut og frigjør minne og resetter flag. Så konverteres data'en til millivolt og legges på data strukturen til SPI bussen. Regne stykket er litt opplösning ganger verdien til data under test.

For å måle på strømmen som går i lysdiodene designet vi en måle krets med en shunt motstand hvor spenningen over denne vil være proporsjonal med strømmen i kretsen. Dette ble simulert i LTspice som kan ses i figur 21 og 22.



Figur 21: Strøm målings krets

For å begrense strømmen i lysdioden måtte vi finne i data blad hvor sterkstrøm en lysdiode tåler. Fant typisk spenning verdi på 2.1V[2, s. 2] og en maks strøm den tåler 7mA[2, s. 5]. Ved ohmslov $R = \frac{V}{I}$, får vi at motstanden minimum må være 215ohm, men setter strømmen til maks 2mA og får nærmeste motstands verdi lik 750ohm. Kretsens design kan ses i figur 21.



Figur 22: Simulert inngangs spenning ved forskjellige pulsvidde fra PID

For å få en gjennomsnitts strømmåling av PWM pulsene som går gjennom dioden må vi lage et filter som lager en gjennomsnitt. Satte C1 til 330nF, og beregnet verdien til R2 ved formelen under. Ved å designe et lavpass filter[6, s. 58] med en mye lavere grense frekvens enn PWM har. Vil en få lite rippel og en verdi som bygger seg opp til strøm trekkets gjennomsnitt selv om det er pulser som blir tilført. Vi satte denne verdien til 5Hz etter litt prøving i LTspice det gå måle resultatene som kan ses i figur 22.

$$R_1 = \frac{1}{2\pi \cdot *f \cdot C} [6, \text{s. } 60] \quad (13)$$

5 Konklusjon

5.1 Proses

(Planlegging, prioritering, Git. Hvordan samarbeidet har fungert.)

Gruppen ble tildelt oppgaven 07.01.2020 i faget «ELE217 - Mikrokontrollere og Styresystemer». Umiddelbart ble oppgaven delt opp i mindre seksjoner og fordelt utover gruppemedlemmene. Jan Egil Fredriksen hadde fra tidligere erfaring med både LaTex og SCRUM, og ble derfor et naturlig valg som SCRUM-MASTER. Vi ble enige om å bruke Bitbucket i kombinasjon med Git som et medie for opp- og nedlasting samt lagring for ulike software-versjoner. I tillegg ble det brukt Jira for å føre timer og lage sprints utover i utviklingsfasen. Dette viste seg å fungere meget bra for medlemmene i gruppen, og ble spesielt nyttig etter nedstengingen av UiA Grimstads Campus når vi ble hindret fra å møtes fysisk for å diskutere og jobbe med prosjektet. Medlemmene har sett seg fornøyd med oppgavefordelingen hvor vi hver for oss har designet ulike blokker og testet disse før blokkene ble sydd sammen som ett felles prosjekt. I starten ble hovedprioriteringene satt til å designe utlesing av MPU6050, BLE kommunikasjon mellom telefon og nRF52 kort samt SPI kommunikasjonen. Deretter jobbet gruppen med å utfylle det nødvendige under disse hovedpunktene, før prosjektet til slutt ble sydd sammen i to enheter.

5.2 Produkt

Målet var å lage et “Flight control board” som får informasjon fra et akselerometer, lager PWM signal til de fire “motor” kontroll enhetene, samt kommuniserer med en smarttelefon.

Vi delte målet til to deler. Den ene delen kalte vi “Master”, mens den andre ble kalt “Slave”.

Master delen inneholder BLE kommunikasjon, UART og SPI.

Slave delen hadde MPU6050 akselerometeret koblet til, DEBUG_UART, PWM signal og gjennomsnitt strøm måling til de fire “ledene”.

Begge del-målene ble testet og det var feilfritt, men grunnet spesielle omstendigheter fikk vi ikke muligheten til å teste begge kortene sammen.

5.3 Utfordringer

Vi valgte å bruke gyroskopet på MPU6050, og det var et utfordring å kunne kompensere drifting fra MPU6050. Ved hjelp av en timer klarte vi å bruke et komplementær filter for å finne mest riktig data fra MPU6050.

PID konstantene er ikke lett å finne, det ble testet mange forskjellige konstanter for å tune PID kontrolleren. Til slutt merket vi at det er en sammenheng mellom de forskjellige konstantene og det ble estimert noen konstanter som ga fornuftige og kjapp reaksjon av PID kontrolleren.

5.4 Videre arbeid

Videre arbeid i dette prosjektet ville naturligvis bestått i å designe en 3D-printed drone-chassis og å erstatte våre simulerte motorer med ekte motorer. Dette vil kreve en del arbeid, da vi ser for oss en del utfordringer knyttet til dette i kontrast til simulering.

Referanser

- [1] Ken Henry Andersen. *Timer implementasjon, Modul 4 - Prosjekt*. 05.02.2020. URL: <https://uia.instructure.com/courses/5066/pages/modul-4-prosjekt>.
- [2] AVAGO. *HLMP-4700, HLMP-4719, HLMP-4740*. 06.04.2020. URL: <https://www.datasheetq.com/datasheet-download/142874/1/Avagotech/HLMP1700>.
- [3] Ian Beavers. *The Case of the Misguided Gyro*. 05.02.2020. URL: <https://www.analog.com/en/analog-dialogue/raqs/raq-issue-139.html#>.
- [4] Joop Brokking. *Brokking.net*. 05.02.2020. URL: <http://www.brokking.net/>.
- [5] invensense. *MPU6050 spec*. 22.12.2019. URL: <https://43zrtwysvxb2gf29r5o0athu-wpengine.netdna-ssl.com/wp-content%20/uploads/2015/02/MPU-6000-Datasheet1.pdf>.
- [6] Rolf Ingebrightsen og Knut Harald Nygåard. *Analog Elektronikk*. lasertrykk.no, 2015. ISBN: 978-82-303-2812-5.
- [7] Mauri Honkanen Antti Lappeteläinen og Kalle Kivekäs. *Low End Extension for Bluetooth*. 2004. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1389107>.
- [8] ARM Ltd. *PID Motor Control*. 10.07.2019. URL: https://www.keil.com/pack/doc/CMSIS/DSP/html/group__PID.html.
- [9] Nordic. *nRF5-SDK*. 5.2.2020. URL: <https://www.nordicsemi.com/Software-and-tools/Software/nRF5-SDK>.
- [10] Nordic. *nRF52832*. 22.12.2019. URL: <https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52832-product-brief.pdf?la=en&hash=2F9D995F754BA2F2EA944A2C4351E682AB7CB0B9>.
- [11] Nordic. *Successive approximation analog-to-digital converter*. 11.03.2020. URL: <https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%5C%2Fsaadc.html>.
- [12] PCB. *MEMS*. 22.12.2019. URL: <https://www.pcb.com/resources/technical-information/mems-accelerometers>.
- [13] Mark Pedley. *Tilt sensing using a three-axis accelerometer*. 05.02.2020. URL: https://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf.
- [14] Mohammed Al-Rawi. *PWM-ELE217-project*. 14.04.2020. URL: <https://www.youtube.com/watch?v=twmgusGE564>.
- [15] Nordic semiconductor. *PWM Pulse width modulation*. 11.10.2017. URL: <https://infocenter.nordicsemi.com/index.jsp>.
- [16] STMicroelectronics. *Datablad*. 01.11.2017. URL: https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/06/a3/c1/ae/7d/27/4c/e0/DM00384353/files/DM00384353.pdf/jcr:content/translations/en.DM00384353.pdf.
- [17] Kevin Townsend. *Introduction to Bluetooth Low Energy*. 1.3.2019. URL: <https://cdn-learn.adafruit.com/downloads/pdf/introduction-to-bluetooth-low-energy.pdf?timestamp=1580883892>.
- [18] Wikipedia. *PWM Pulse width modulation*. 19.12.2019. URL: https://en.wikipedia.org/wiki/Pulse-width_modulation#cite_note-8.

A Timeplan og Tidsestimat

ELE217 Prosjektoppgave – DRONEKONTROLL

V.2

Flight control board – Får info fra GY-512/UM2197, Lager PWN til fire «motorer» samt kommunikasjon med en smarttelefon. «Motorene» emuleres vha. LED lys.

Start dato: 07.01.2020

Frist innlevering: 24.04.2020

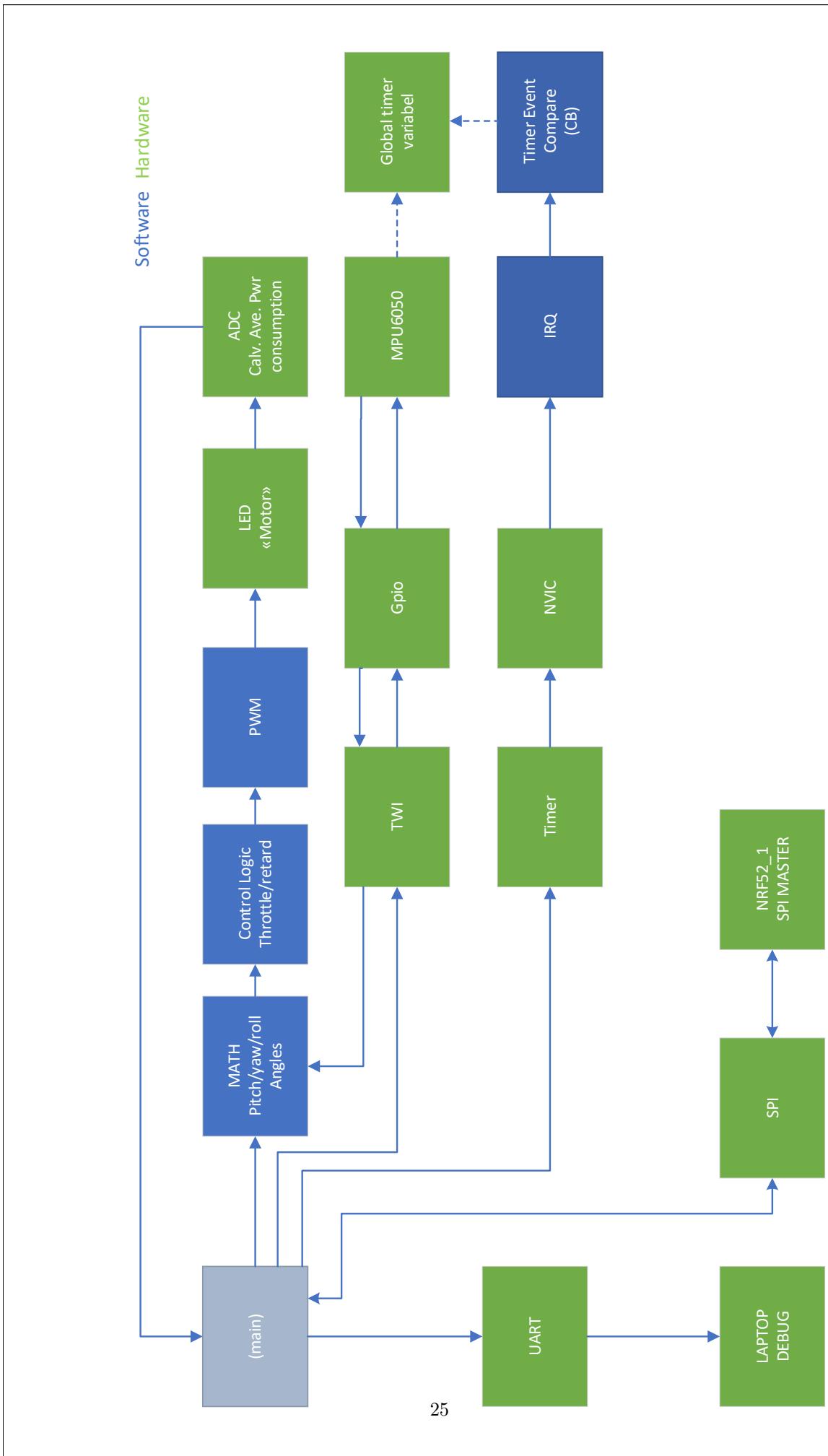
Tid: 108 dager.

Grovt underestimert tidsforbruk pr. 20.12.2019 (WORST CASE) : ~110 timer

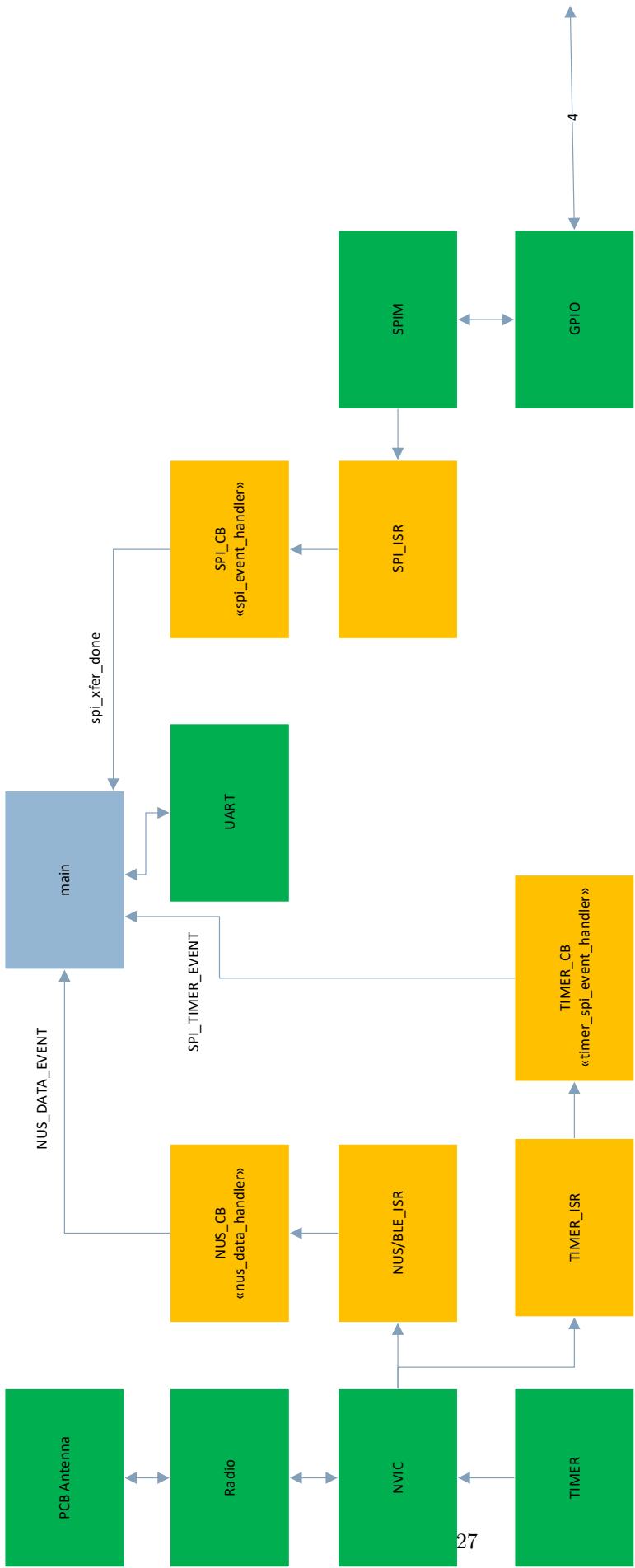
Grovt underestimert tidsforbruk pr. 20.12.2019 (BEST CASE) : ~70 timer

- **NRF52-1 (TWI-Master / SPI-slave)**
 - Kjøre RTOS og være hovedkomponent i kontrolleren.
 - Tilkoblet akseleromteret
 - Regne ut posisjon til «drone»:
 - Pitch – X – Nose/Tail UP
 - Yaw – Y – Nose SIDE TO SIDE
 - Roll - Z – CIRCULAR – Clock-/Anti-Clockwise
 - Gi PWM styresignal til «motorer» (LED)
 - «Motorer» under midtpkt. gis ekstra pådrag.
 - «Motorer» over midtpkt. gis redusert pådrag.
 - Måle gj. Snitt strømforbruk til motorer (FITTER)
 - ADC?
- **NRF52-2 (SPI-Master / BLE Server)**
 - Kommunikasjon med smarttelefon over BLE
 - Bruke ble_app_uart eksempel – tolke kode – Tegne SDL med fokus på endringer
 - Legg til SPI for kommunikasjon med NRF52-1 (MASTER)
 - Sende/Motta data vha. UART til PC
 - **Smarttelefon med BLE mot NRF52-2 (SLAVE)**
 - Be om akselerometermåling fra MPU6050
 - Be om vinkelen til drone ift. horisontal vinkel
 - Be om pådrag for alle 4 motorer i %
 - Be om gjennomsnittsstrømforbruk til alle 4 motorer i %
 - Sende ønsket vinkel drone skal ha ift. Horisontal vinkel
 - **PC over UART mot NRF52-2 (SLAVE) (VIRTUELL UART OVER USB)**
 - Be om akselerometer måling fra MPU6050
 - Be om vinkel til drone ift. Horisontal vinkel
 - Be om pådrag for alle 4 motorer i %
 - Be om gjennomsnittsstrømforbruk til alle 4 motorer i %
 - Sende ønsket vinkel drone skal ha ift. Horisontal vinkel.

B SW og HW SDL av kontrollenhet

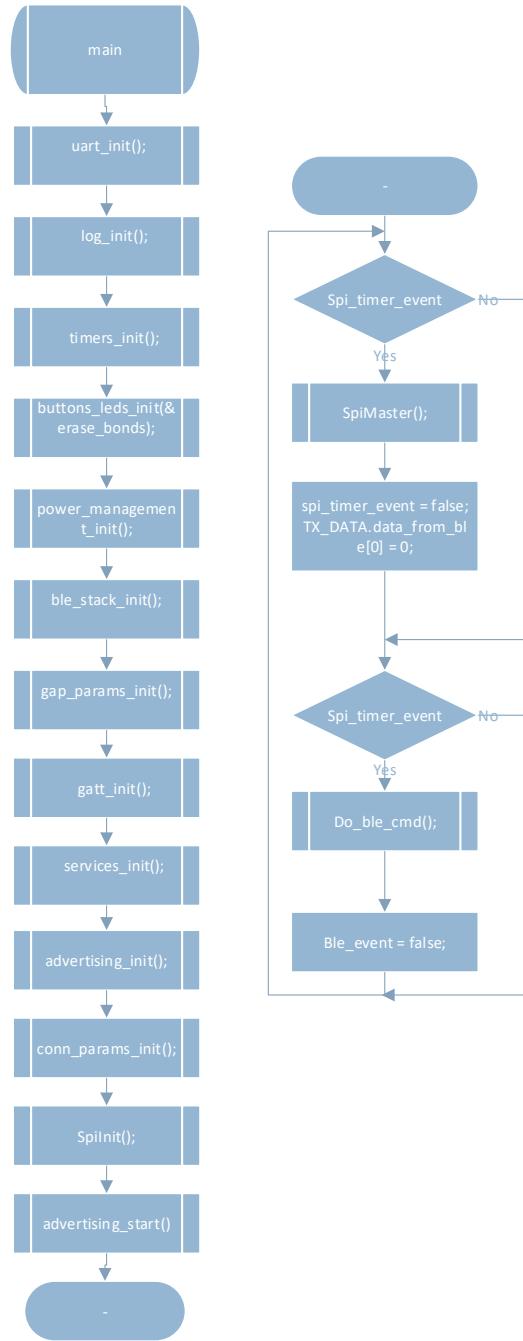


C SW og HW SDL av styreenhet



HW and SW SDL of nRF52_2 Control unit

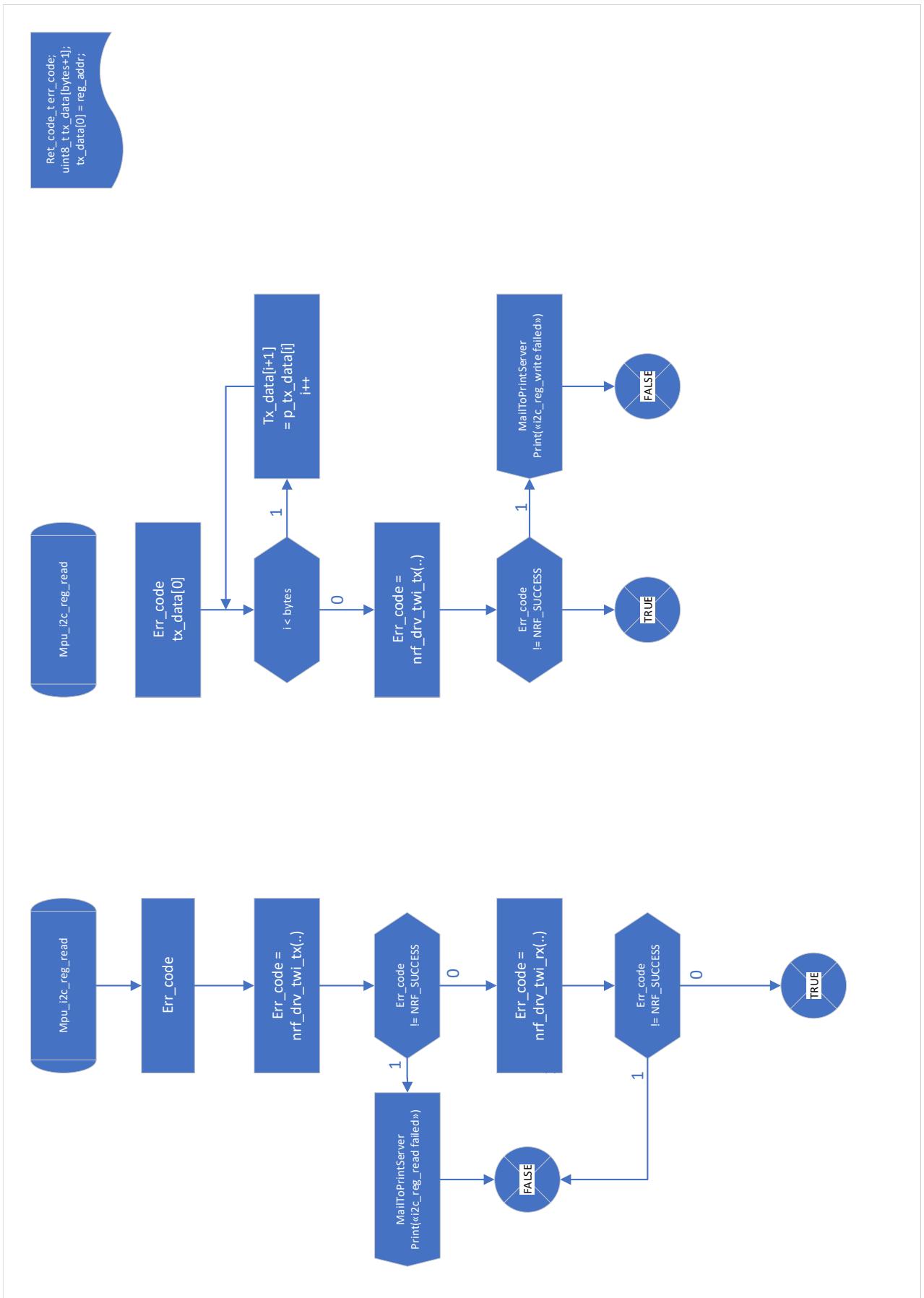
D SDL for nRF52 1 styreenheten



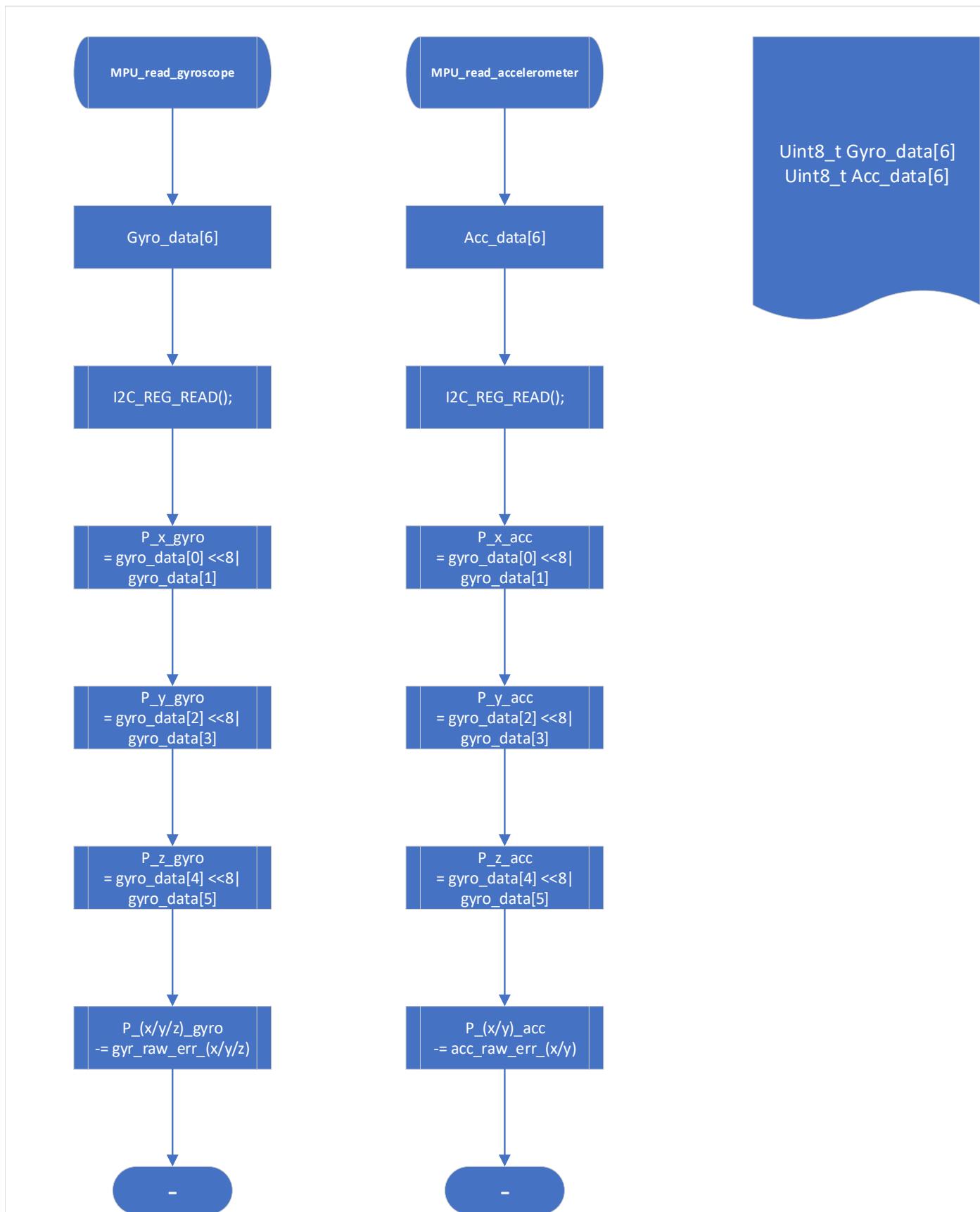
E SDL for Main i nRF52 Kontrollenhet



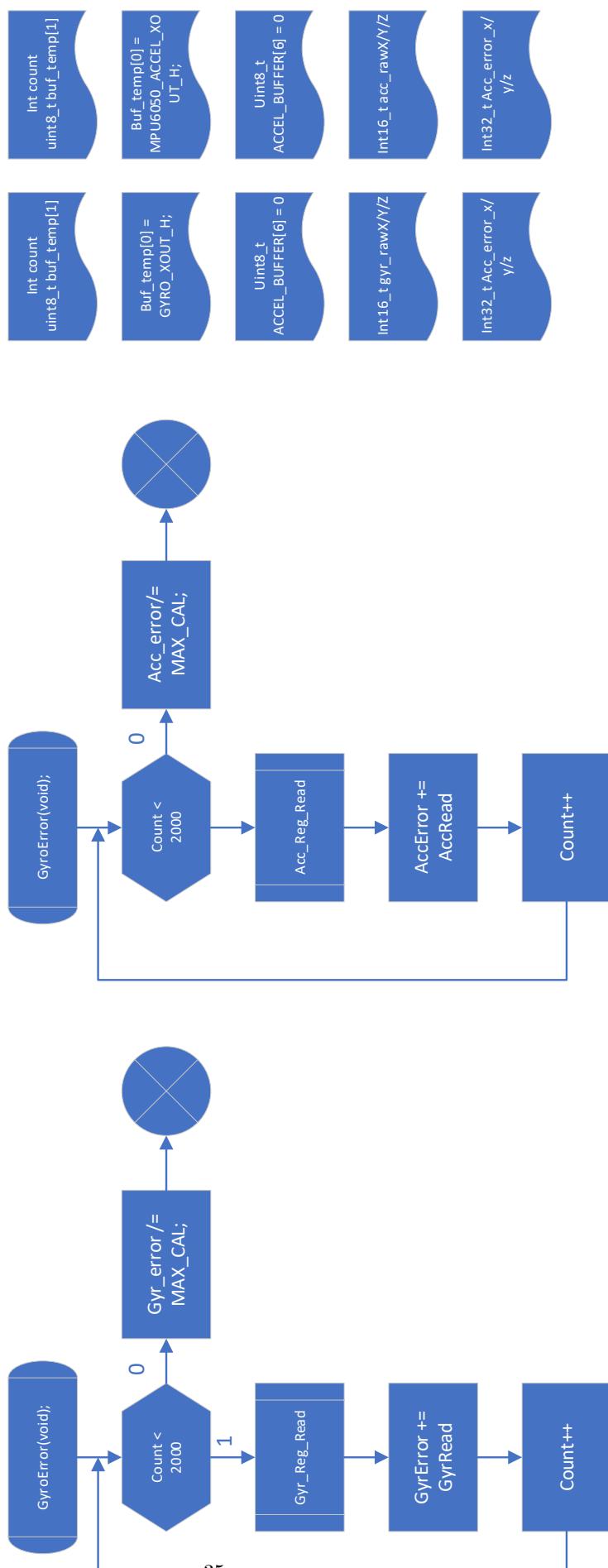
F SDL for i2c register read og write i nRF52 Kontrollenhet



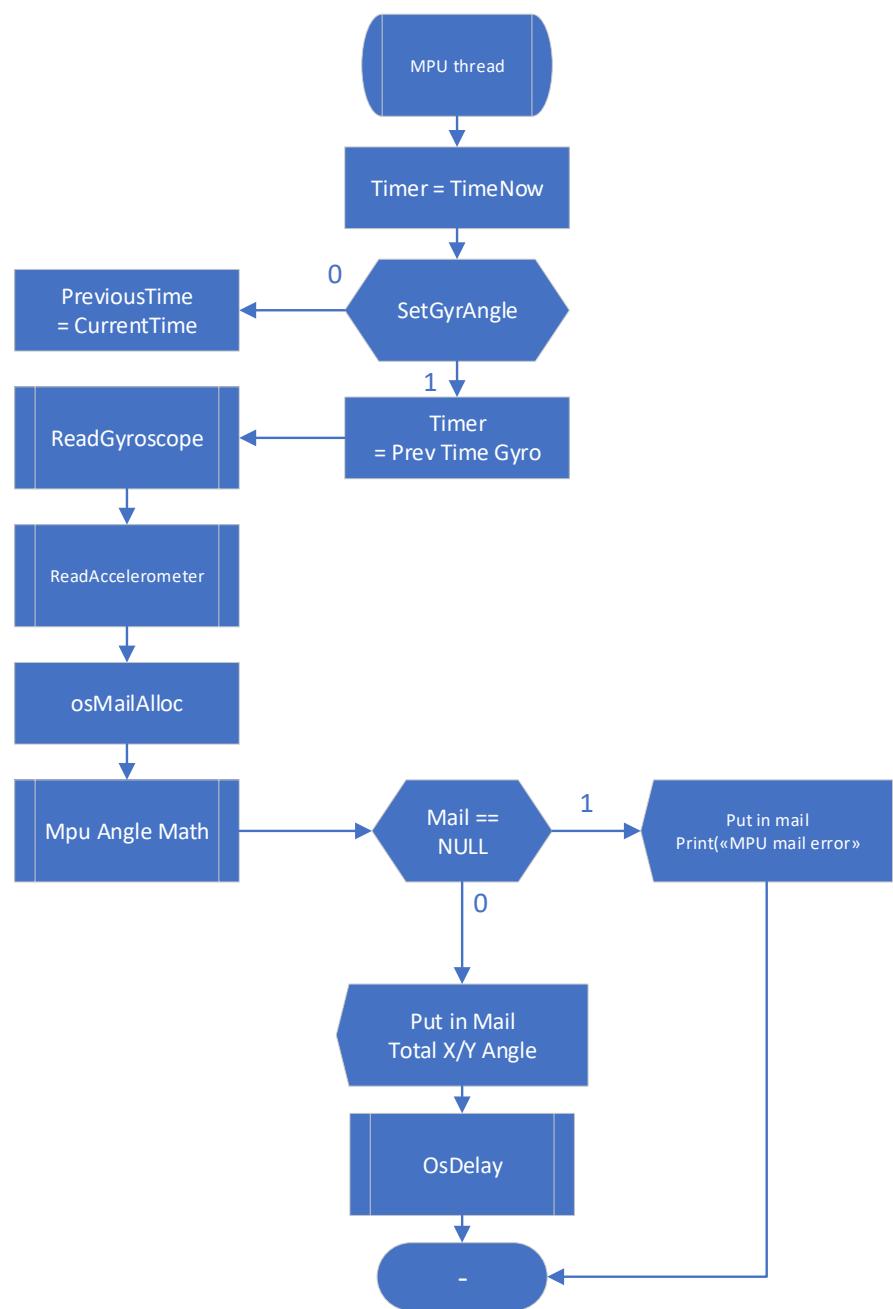
G SDL for accelerometer og gyroskope read nRF52 Kontrollehet



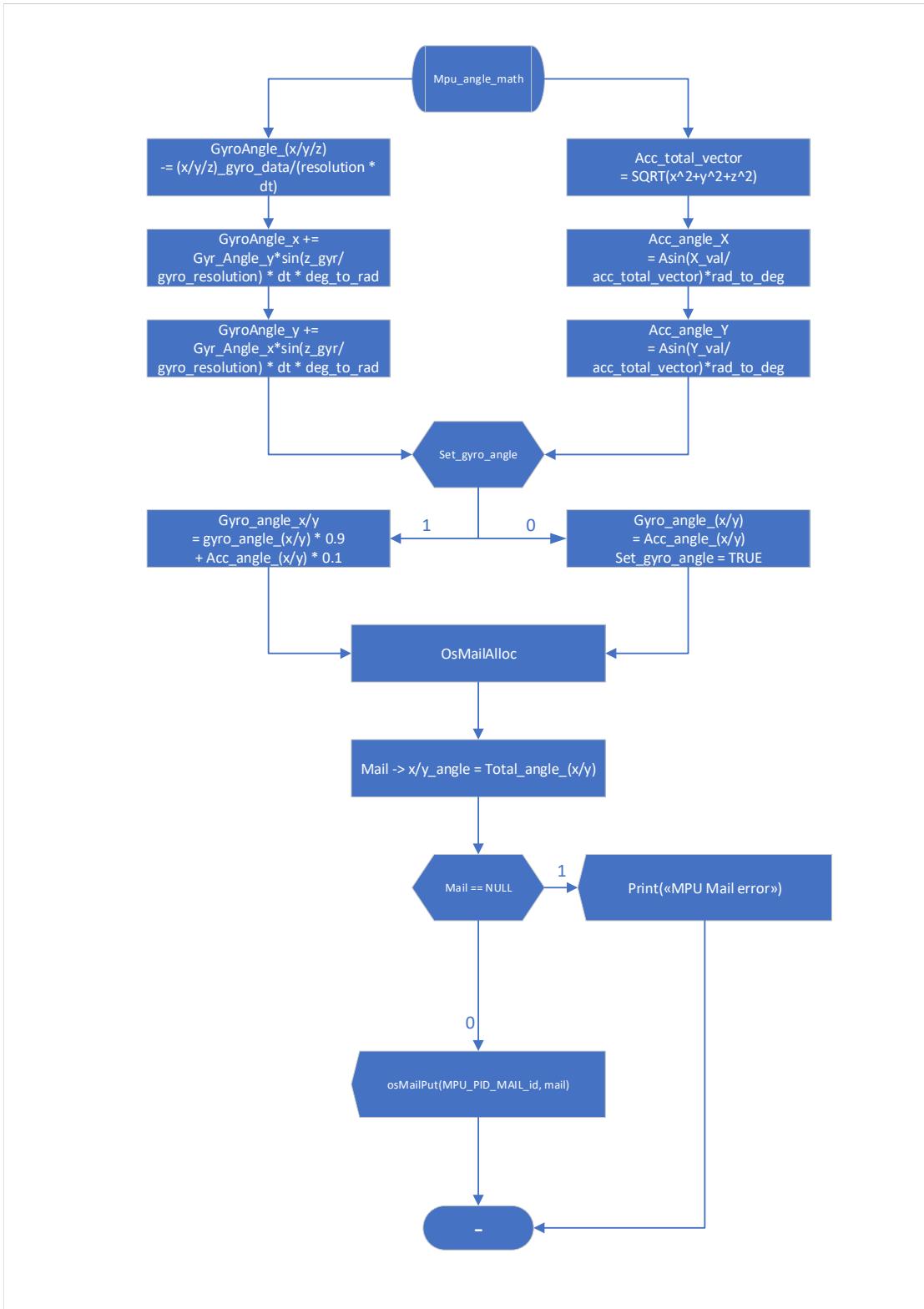
H SDL for MPU6050 Calibration nRF52 Kontrollenhet



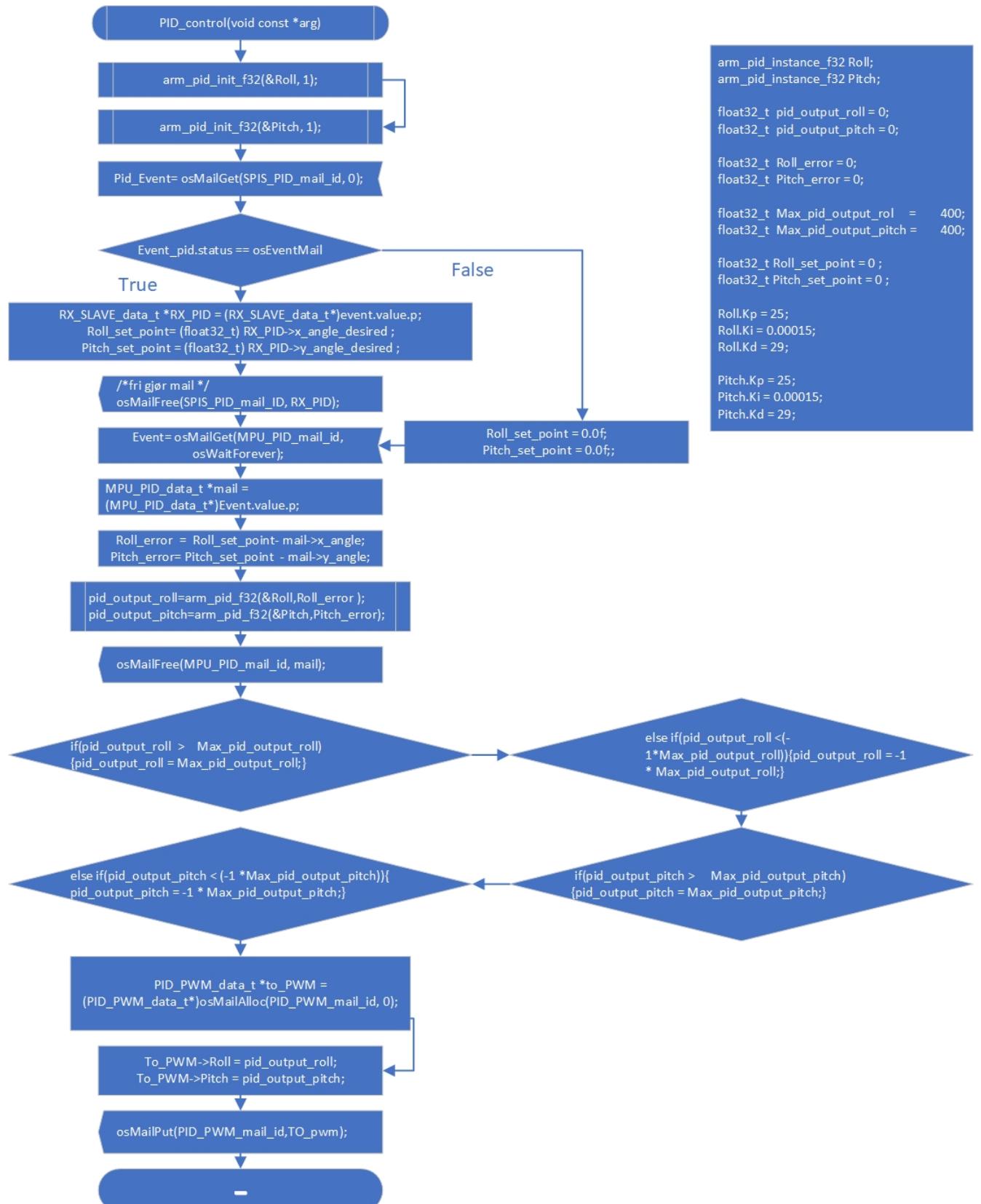
I SDL for MPU6050 tråd i nRF52 Kontrollenhet



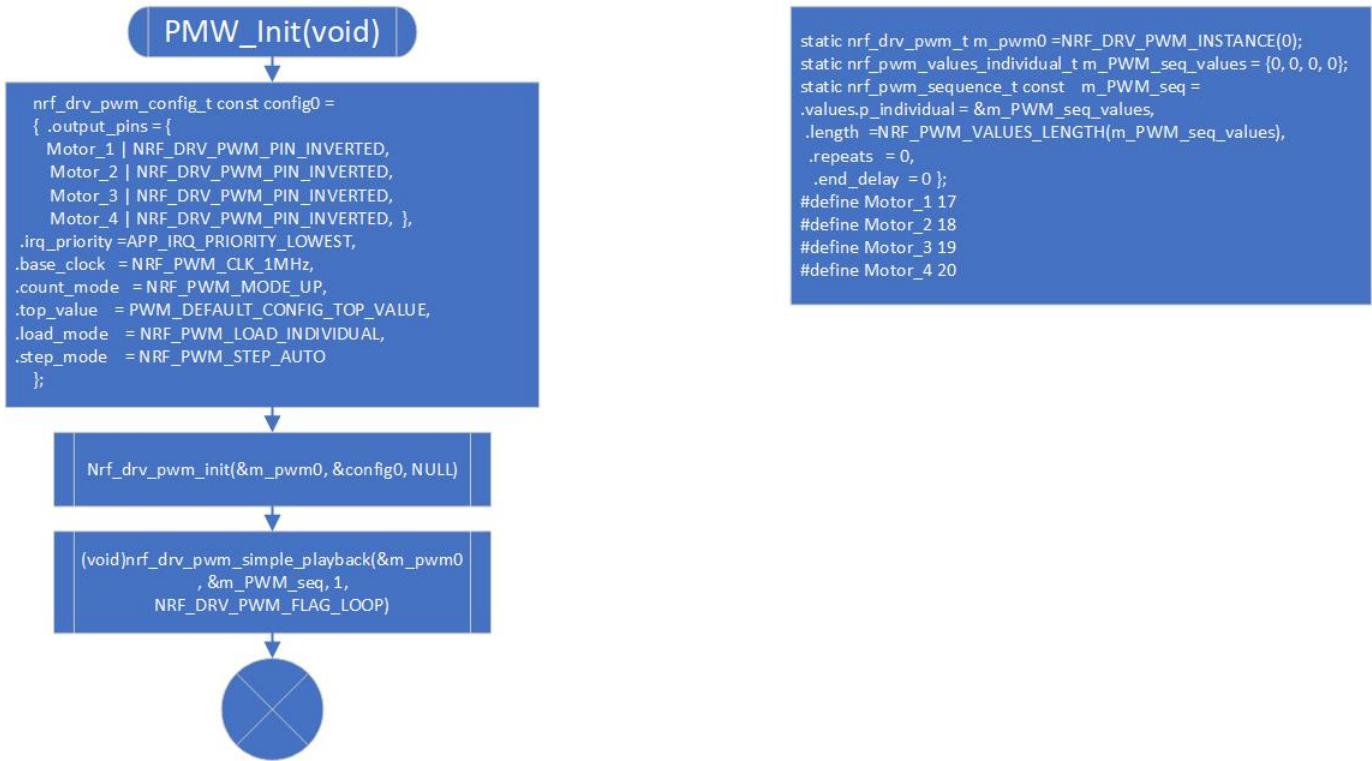
J SDL for MPU Angle Math



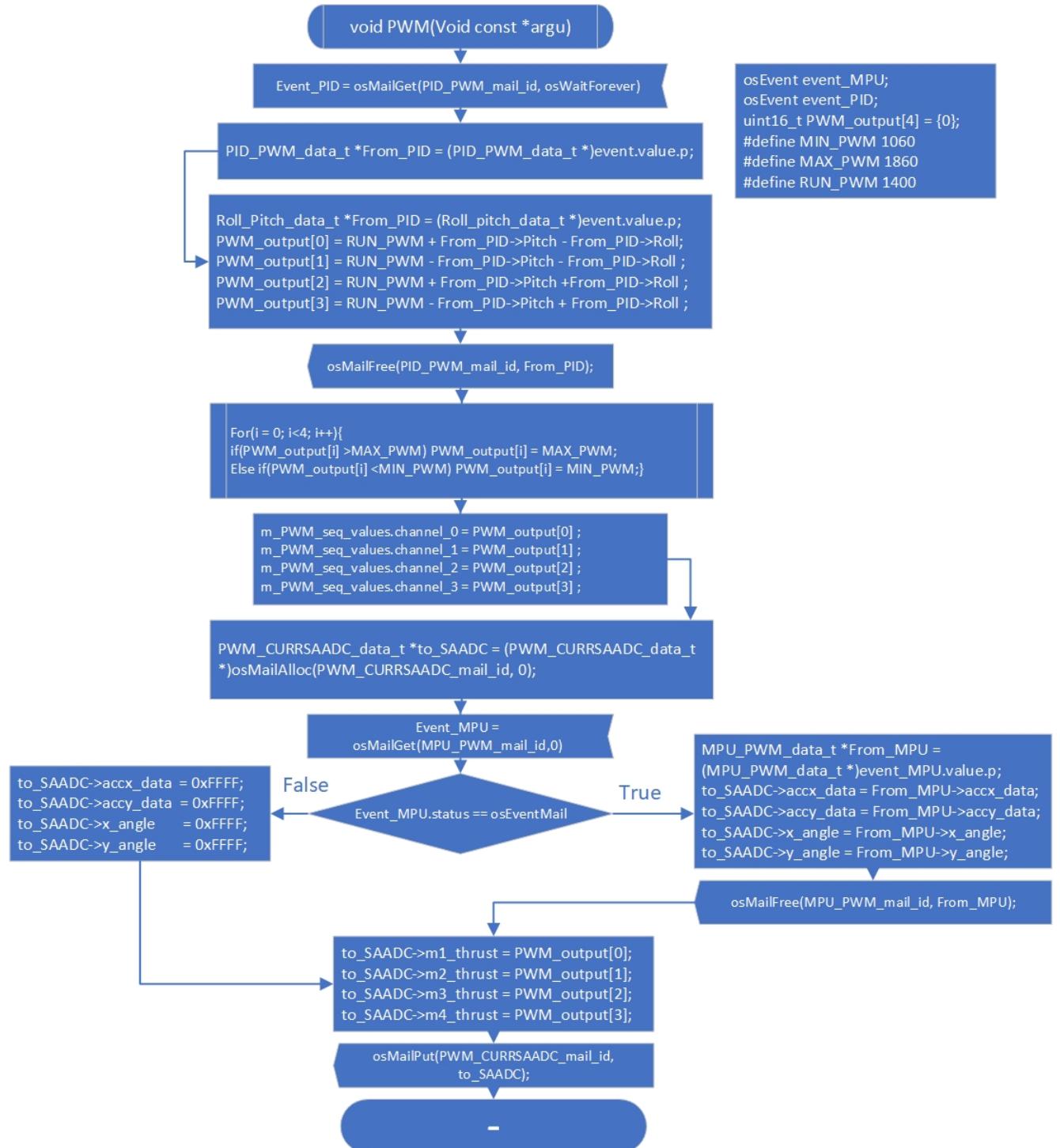
K SDL for PID i nRF52 2 kontrollenhet



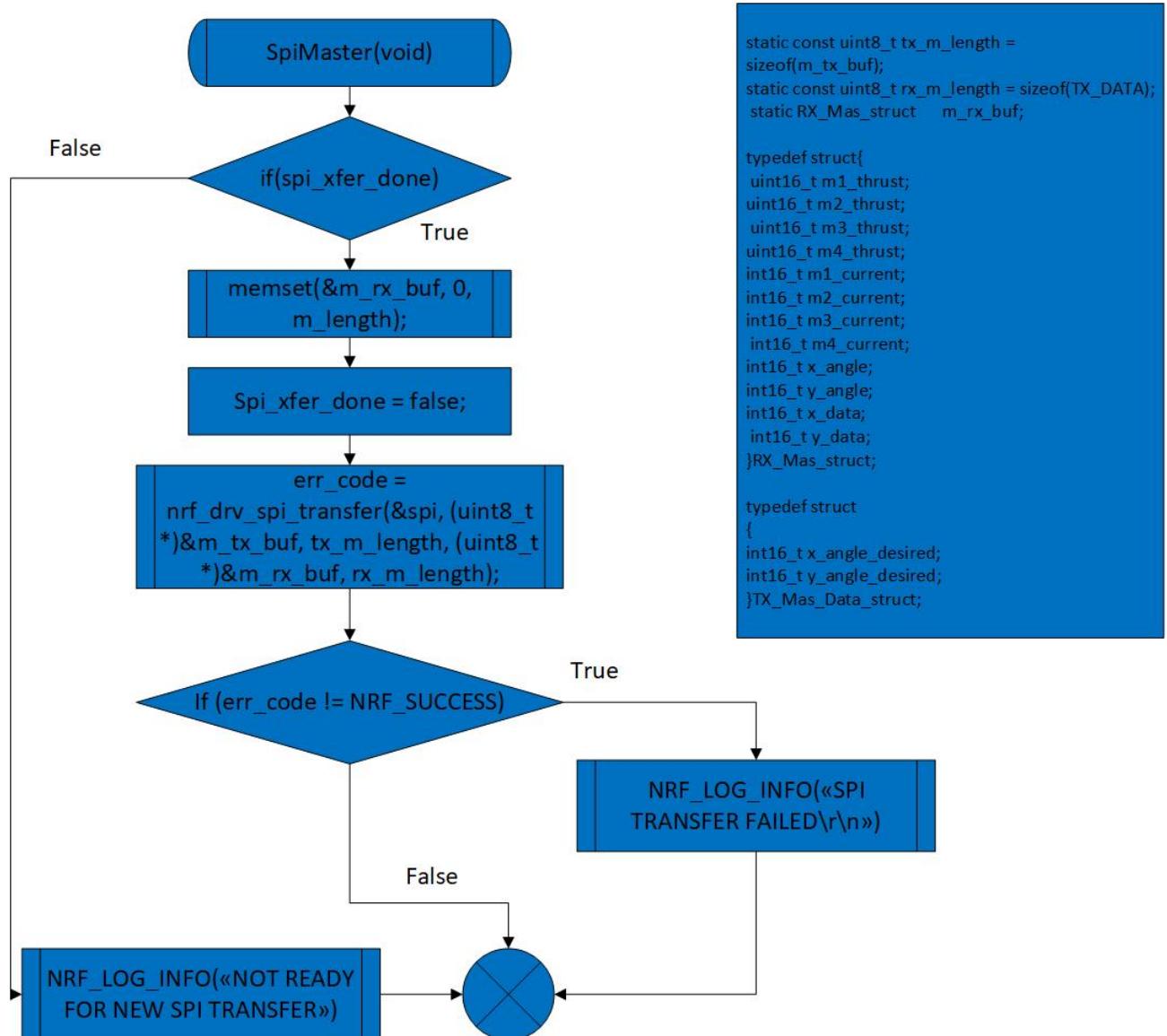
L SDL for PWM init nRF52 Kontrollenhet



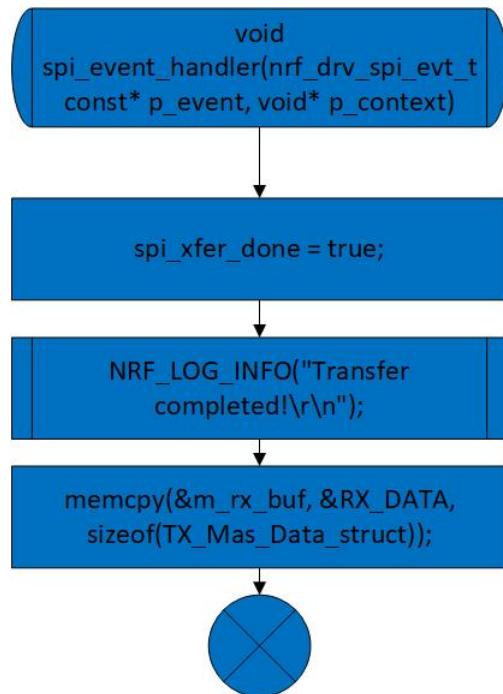
M SDL for PWM i nRF52 2 kontrollenhet



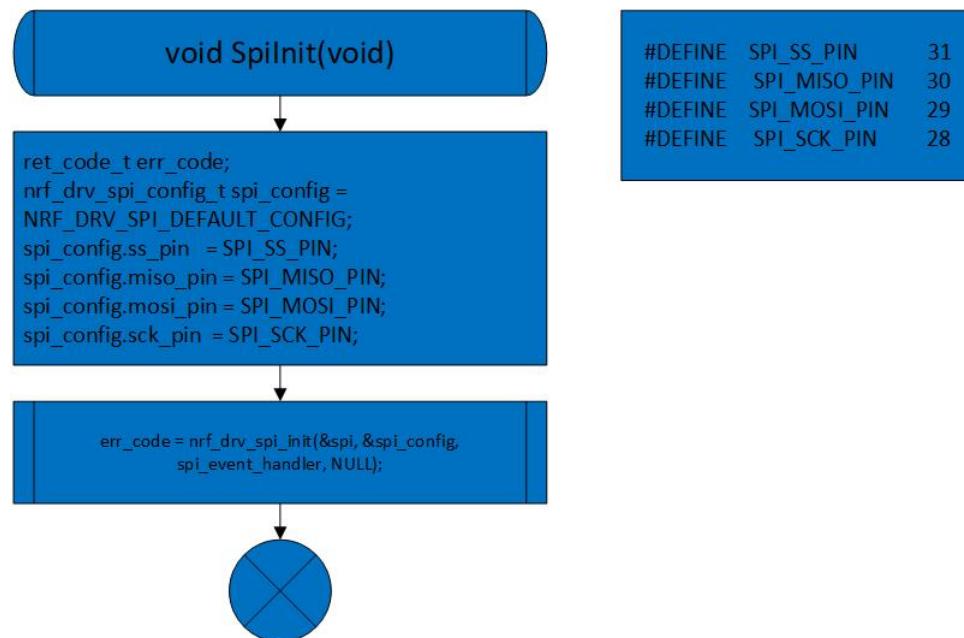
N SDL for SPI master funksjon



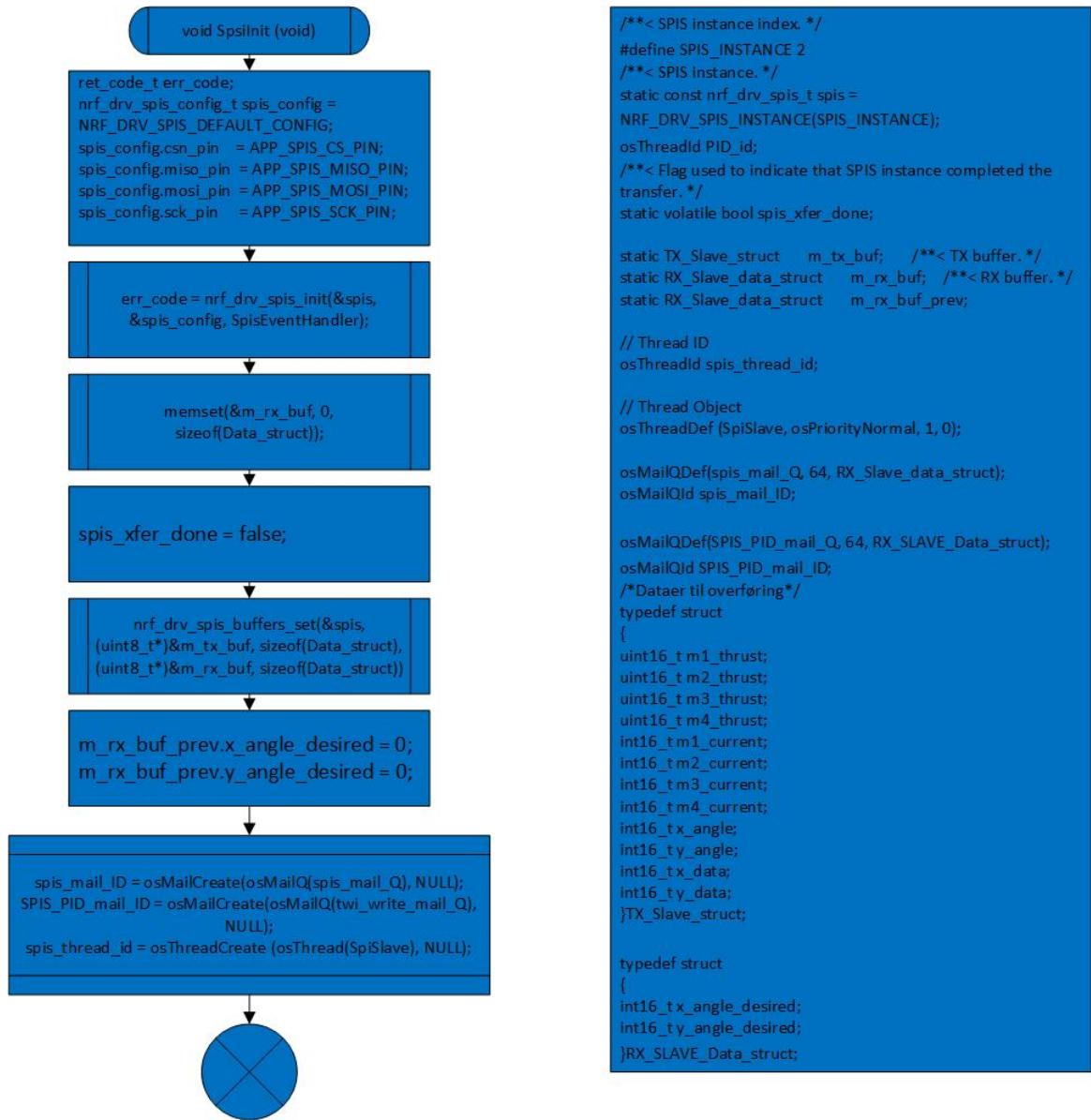
O SDL for SPI master event handler



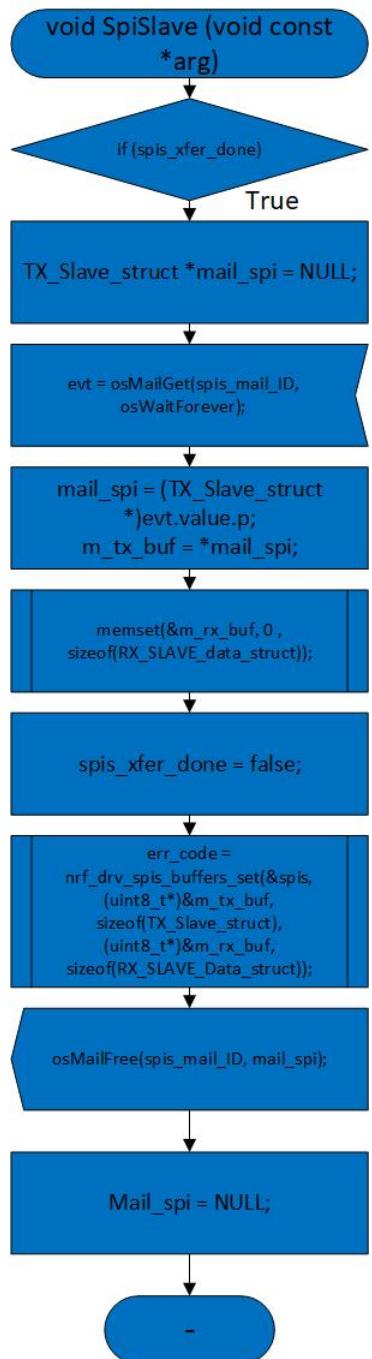
P SDL for SPI master init



Q SDL for SPI slave Init



R SDL for SPI slave tråd



S SDL for SPI slave event handler

