NTNU
Det skapende universitet

Presisjonslanding av telt i kriseområder

Ole Jørgen Karstensen

Master i kybernetikk og robotikk
Innlevert: juni 2017
Hovedveileder: Tor Engebret Onshus, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Abstract

In this thesis a prototype for automatic control of a parachute has been developed. The aim is to guide the parachute towards predefined GPS coordinates. The control system is implemented on the single-board microcontroller Arduino Uno. The prototype includes a GPS sensor for position tracking, an inertial measurement unit for attitude estimation and four servo motors to manipulate steering lines. Each servo motor is connected to one of the parachute's four steering lines. Code is written to sample relevant data from the sensors, process it to usable data and calculate servo motor control input. Filtering schemes has been derived and implemented. The IMU data is filtered using the discrete complementary filter in order to suppress noise from the sensors.

The tests have not been entirely positive. One can not determine based on the results that the prototype actively controls the parachute towards the target point. Results shows that the estimation of roll, pitch and yaw is not satisfactory and do not to yield reliable data. The IMU sensor suffers from too much noise due to rough test environment conditions and further consideration should be taken when filtering the signals. The GPS do deliver reliable data.

Recommended adjustments and improvements are provided. Among other things, a Kalman filter is suggested for better attitude estimation and a change of test environment due to major environmental disturbances.

Sammendrag

I denne masteroppgaven er det bygget en prototype og utviklet et kontrollsystem for automatisk styring av en fallskjerm. Prototypen tar sikte på å styre en fallskjerm mot et forhåndsdefinert punkt. Kontrollsystemet er implementert på mikrokontrollerbrettet Arduino Uno og inkluderer en GPS-sensor, en IMU og fire stykk servomotorer. GPS-sensoren bestemmer hvor droppet befinner seg relativt til det forhåndsdefinerte punktet, IMU-en bestemmer enhetens orientering i lufta og servomotorene manipulerer styrelinene. Hver servomotor er koblet til hver sin styreline på fallskjermen. Kode er skrevet for å hente inn relevante sensordata, prosessere dette og beregne inngangssignal til servomotorene. For å dempe sensorstøy er et diskret filter, kjent som komplementærfilteret, utledet og implementert.

Testene har ikke vært gjennomgående positive og man kan ikke fastslå basert på resultatene at prototypen styrer aktivt mot målpunktet. Resultater viser at estimatet av roll, pitch og yaw ikke er tilfredstillende med tanke på forutsigbar styring av fallskjermen. Testmiljøet er en tøff utfordring for sensorene og det er vanskelig å filtrere målingene tilstrekkelig. GPS-sensoren leverer pålitelig informasjon under testing, men det har ikke lyktes med å hente god informasjon fra IMU under dropp.

Anbefalte justeringer og forbedringer er gitt. Blant annet er et Kalmanfilter er foreslått for bedre estimater av orienteringsvinkler og bytte av testmiljø på grunn av store forstyrrelser fra omgivelser.

Forord

Denne rapporten er skrevet under vårsemesteret 2017 og er et resultat av det avsluttende arbeidet av min 5-årige mastergrad i Teknisk kybernetikk ved Norges Teknisk-Naturvitenskapelige Universitet. Den vil være til vurdering i emnet *TTK4900 Teknisk kybernetikk, masteroppgave - våren 2017*, som utgjør 30 studiepoeng.

Takk til min veileder Professor Tor Onshus for å legge til rette for og invitere til dialog for å sikre fremgang i prosjektet.

Takk til Tor Harald Jensen i Paraweather AS for oppgaven og for utstyret jeg fikk låne underveis.

Takk også til mekanisk verksted på ITK for lån av loddestasjon, deler og verktøy.

Problembeskrivelse

Et firma har utviklet et telt som tenkes brukt i nødhjelp og kriseområder der utslipp fra fly kan være eneste mulige transportmetode. Vanlig dropping fra fly vil spre teltene over store områder og en ser på metoder for å styre dem aktivt så de lander samlet. Prisen per telt bør være så liten som mulig, og de bør lande nærmest mulig et forhåndsbestemt punkt der de enkelt kan settes opp. En ser for seg et GPS-basert system der hver enhet har informasjon om hvor de skal lande og kan styre seg inn mot landingspunktet. Hvis en får fram et kostnadseffektivt konsept kan det antakelig også benyttes for å få fram annen nødhjelp der veier og annen infrastruktur er enten ødelagt eller overbelastet.

Oppgaven vil bestå i å komme opp med ulike konsepter og vurdere disse. Utvikling av prototyper for styring og uttesting i småskala av de mest lovende løsningene er også ønskelig.

Motivasjon

I krisesituasjoner, krig og katastrofer kan behovet for et tak over hodet bli akutt. Telt kan dekke dette behovet og avhengig av situasjonen kan flydropp være eneste mulighet til distribusjon på grunn av ødelagt eller manglende infrastruktur. Situasjon, geografi og vær kan kreve at dropp blir utført fra en slik høyde at stor spredning på dropp er sannsynlig. Større nøyaktighet og økt presisjon på dropp er i høyeste grad ønskelig for gjøre arbeidet med humanitær hjelp så effektivt som mulig.

Problemstillinger

Det er ønskelig å lande flydroppet nærmest mulig et gitt punkt. Nøyaktighet i dette tilfellet kan for eksempel være å lande innenfor et område på størrelse av en fotballbane ved et dropp fra normal høyde. I denne oppgaven skal vi

1. vurdere tidligere arbeid for økt presisjon av flydropp.
2. utvikle prototype for styring av en fallskjerm basert på Arduino Uno.
3. vurdere kostnadseffektiviteten ved en mulig løsning.

Innhold

Figurer

Tabeller

Nomenklatur

GPIO	=	General Purpose Input/Output
IMU	=	Inertial Measurement Unit
GPS	=	Global Positioning System
Roll	=	Rull eller kregning
Pitch	=	Stamp eller angrepsvinkel
Yaw	=	Gir eller dreining
EEPROM	=	Electrically erasable programmable read-only memory
SRAM	=	Static Random Access Memory
RISC	=	Reduced Instruction Set Computer
ϕ	=	Roll-vinkel
θ	=	Pitch-vinkel
ψ	=	Yaw-vinkel
UAV	=	Unmanned Aerial Vehicle

Kapittel 1

Introduksjon

1.1 Bakgrunn

Paraweather AS [1] ved Tor Harald Jensen har utviklet et telt som er tenkt brukt i forbindelse med nødhjelp i katastrofeområder. I forbindelse med innsalg til FN og muligens andre hjelpeorganisasjoner er det ønskelig å kunne spille på flere strenger - et mer presist flydropp er en av de.

Manglende eller ødelagt infrastruktur kan gjøre det utfordrende eller umulig å frakte inn nødhjelp annet enn med fly. Flydropp fra store høyder kan være aktuelt dersom situasjonen krever det. Avhengig av vær- og vindforhold, ferdighetene til pilot og annet personell kan droppet spres over store områder - noe som ikke er ønskelig da det gjør arbeidet med humanitærhjelp mindre effektivt. Et eksempel fra 2016: FNs første flydropp med nødhjelp i Syria ble sluppet over byen Deir al-Zor - "Av de 21 pallene som ble sluppet over byen Deir al-Zor, ble fire ødelagt, sju landet i ingenmannsland, og ti er ikke gjort rede for" [2].

Det innebærer i tillegg en risiko for personskade ved flydropp over områder med mennesker i nød, hjelpepersonell og frivillige. I tillegg er også nødhjelpen som skal droppes ekstra utsatt om man ikke treffer der man skal.

Følgelig ønsker man å lande droppet med minimal spredning og nærmest mulig et gitt referansepunkt.

1.2 Eksisterende løsninger

Fallskjermsystemer med automatisk styring har i det militære vært under utvikling lenge. Det amerikanske forsvaret tok et slikt system i bruk for å levere forsyninger til soldater i Afghanistan i 2006 [3].

JPADS, eller *Joint Precision Airdrop System*, er det amerikanske militærets program for presisjonslanding av flydropp [3]. I starten var metoden å hente inn tilgjengelig vær- og vinddata for deretter å kalkulere et optimalt utslippspunkt. Dette for å minimere avstanden mellom landingspunkt og målpunkt for en ikke-styrt fallskjerm [4].

Nå baserer man seg på GPS-, vær- og vinddata, lufttrykk og temperatur for å styre en firkantfallskjerm mot et forhåndsdefinert punkt. Ombord finnes det en datamaskin for prosessering av sensordata og sende styreinput til motorer som manipulerer styrelinene. Dette systemet har vært lite brukt i humanitær sammenheng. Det amerikanske forsvaret ønsker ikke å risikere at teknologien havner i feil hender der utstyret ikke kan bli hentet tilbake av amerikanske tropper. Med prislappen 60 000$ (ca. én halv million norske kroner) vil dette systemet også bidra vesentlig til regnskapet.

I senere tid har det også blitt utviklet styring som kun baserer seg på bildebehandling, slik at styring er mulig også der man ikke kan basere seg på GPS-signaler [5; 6]. Dette systemet etterligner måten fallskjermhoppere styrer sine skjermer - basert på visuell tilbakemelding. Under flyturen finner enheten sin egen posisjon ved å sammenligne data fra optiske sensorer med satelittbilder av samme område.

Artikkelen *Precision Airdrop* fra 2005 [7] beskriver flere systemer for presisjonslanding av flydropp. MMIST Sherpa, Strong Enterprises SCREAMER, DRAGONFLY og Capewell and Vertigo AGAS (The Affordable Guided Airdrop System) er alle eksempler på systemer som basert på blant annet GPS-informasjon forsøker å øke nøyaktigheten på flydropp. Sistnevnte er en løsning som inkluderer en rundfallskjerm med fire styreliner manipulert av to aktuatorer. Linene kan manipuleres individuelt eller i par slik at den kan styre i åtte forskjellige retninger.

1.3 Valg av løsning til implementering

Løsningen som skal bli forsøkt implementert ligner AGAS. Løsningen vil i likhet med systemene over inneholde en GPS-sensor for å vite hvor droppet er i relasjon til et forhåndsdefinert punkt. I tillegg inkluderes en IMU (Inertial Measurement Unit) med et magnetometer. IMU-en skal bestemme enhetens orientering og kompassretning. Basert på denne informasjonen skal enheten vite hvor langt unna den er målpunktet og i hvilken retning den må fly for å komme dit.

1.4 Begrensning av oppgaven

Flydropp i denne sammenheng består som regel av mange enheter, men denne oppgaven er begrenset til styring av én enhet.

1.5 Rapportens oppbygning

Kapittel 1 tar sikte på å gi noe bakgrunnsinformasjon og motivasjon for denne oppgaven. I tillegg vil noen eksisterende løsninger omtales kort og noe om valg av løsning til implementering. Kapittel 2 lister opp komponentene som er brukt i prosjektet og litt generell teori omkring disse. Kapittel 3 er en dokumentasjon av det som er implementert - fysisk oppsett og software. En utgreiing om nøkkeldata og beregninger vil også bli gitt her. Resultater fra tester blir presentert i kapittel 4, og kapittel 5 vil være en diskusjon av resultater og videre arbeid.

Kapittel 2

Hardware og software

Dette kapittelet gir bakgrunnsinformasjon for hardware og software brukt i forbindelse med oppgaven.

2.1 Hardware

Informasjon om de fysiske komponentene brukt i arbeidet følger under.

2.1.1 Arduino Uno

Arduino-brettet som ble valgt for å implementere styresystemet var Arduino Uno [8] (se figur 2.1) , som bruker mikrokontrolleren ATmega328P [9]. ATmega328P er en 8-bit AVR RISC-basert mikrokontroller med en klokkehastighet på 16MHz. Arduino Uno har 32kB flash-minne, 1024B EEPROM, 2kB SRAM og 23 GPIO-porter.

Figur 2.1: Mikrokontrollerbrettet *Arduino Uno* brukt i dette arbeidet. Bildet er hentet fra: [10]

Dette brettets spesifikasjoner ble vurdert som tilfredsstillende for dette prosjektet. Arduino Uno er det mest brukte og best dokumenterte [8] i Arduino-serien. Det finnes mange offentlige biblioteker for forenkling av grensesnitt mellom sensorer og mikrokontroller.

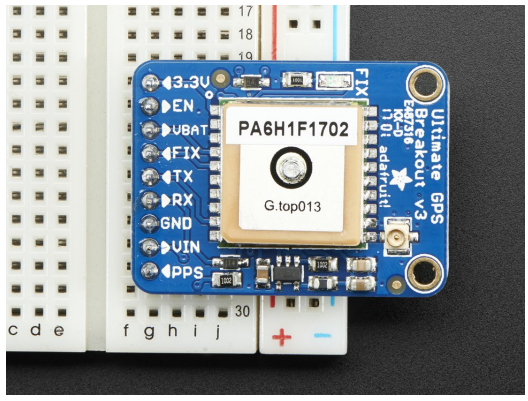
Et ankepunkt kan være Unos presisjon av flyttall. Arduino lagrer datatypen *float* i 4 bytes, og oppnår kun en presisjon på 6-7 sifre. Man kan ikke øke presisjonen med datatypen *double*, da denne datatypen er den samme som float på Arduino [11].

Eksempel: Arduino Uno vil beregne minste avstand mellom to GPS-målinger til å være 1.5190m ved endring av siste signifikante siffer i breddegrad, og 0.5098m ved samme tilfelle i lengdegrad (Haversine-funksjonen er brukt for utregning av distanse [12]). Dette kan gi utfordringer i tilfeller der droppet flyr rett over landingspunkt. Til sammenligning vil samme utregning i MATLAB (med dobbel presisjon) gi 1.2182m og 0.4976m, henholdvis. I tillegg til begrenset presisjon kommer også et avvik som følge av at avrundingsfeil akkumuleres i utregningene. Derfor vil minste avstand mellom GPS-punkter vil være begrenset av dette.

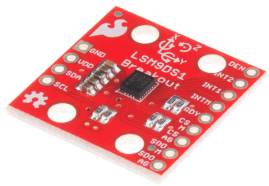
I en reell situasjon vil avstandene vil være store nok til at denne presisjonen blir ubetydelig. I testsammenheng med den prototypen som er utviklet vil avstandene være mye mindre, men det er likevel vurdert tilstrekkelig for å undersøke prototypens virkemåte i denne fasen av utviklingen.

2.1.2 Global Positioning System

For å få informasjon om enhetens posisjon inkluderes det er GPS-sensor. Sensoren som er brukt i dette prosjektet er *Adafruit Ultimate GPS Breakout - Version 3* [13] (se figur 2.2). Sensoren har en høy-sensistiv mottaker (-165dBW, utendørs er signalene typisk -155dBW [14]), drar lite strøm (20mA) og har en oppdateringsfrekvens på 10Hz.



Figur 2.2: GPS-sensoren brukt i dette arbeidet. Bildet er hentet fra: [13]



Figur 2.3: IMU-en brukt i dette arbeidet. Bildet er hentet fra: [15]

2.1.3 Inertial Measurement Unit (IMU)

For å kunne avgjøre hvilken retning fallskjermen skal styres og enhetens orientering, er en IMU inkludert. Til dette prosjektet ble *Sparkfun 9DOF IMU Breakout - LSM9DS1* [15] valgt, se figur 2.3. LSM9DS1 inneholder et 3-akset akselerometer og gyroskop i tillegg til et 3-akset magnetometer. Dette gir ni typer informasjon: akselerasjon/g-krefter i x-,y- og z-retning, vinkelakselerasjon i i x-,y- og z-retning og styrken av det magnetiske feltet i x-,y- og z-retning.

2.1.4 Servomotorer

Servomotorene som er brukt i dette prosjektet er av typen *Tower Pro MG90S* [16] (se figur 2.4). Deres jobb vil være å trekke i fallskjermens styreliner.

Utvalgte spesifikasjoner:

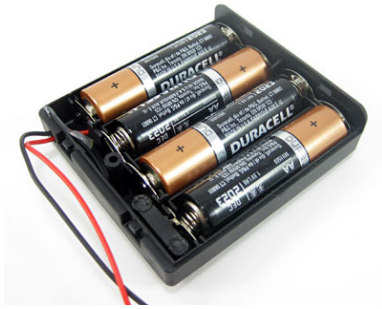
- Vekt: 13.4g
- Dimensjon: 22.8x12.2x28.5mm
- Dreiemoment: 1.8kg/cm (4.8V); 2.2kg/cm (6.6V)
- Hastighet: 0.10sek/60°(4.8V); 0.08sek/60°(6.0V)
- Driftspenning: 4.8V $\sim$ 6.6V

2.1.5 Batteri

Anbefalt inngangsspenning på Arduino-brettet er 7-12V [8], så et 9V alkalisk batteri ble valgt til å mate brettet med strøm (se figur 2.5a). Servomotorene tar ikke strøm fra Uno-brettet, men har sin egen strømforsyning. Dette er fordi motorene periodevis kan dra så mye strøm at Uno-brettet resettes på grunn av lav spenning. Den eksterne strømkilden består av en batteriholder med fire stykk seriekoblede 1.5V AA-batterier med en total spenning på 6V (se figur 2.5b).

Figur 2.4: Servomotortypen brukt i dette arbeidet. Bildet er hentet fra: [16]

(a) 9V-batteri.



(b) 4xAA-batteriholder

Figur 2.5: Strømkilder.

2.1.6 Annet

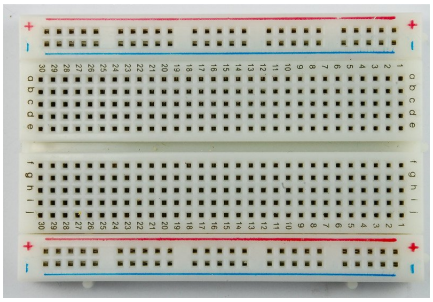
LCD

En 16x2 LCD (se figur 2.6) er også inkludert for å kunne lese noe sensorinformasjon og utvalgte kalkuleringer direkte fra enheten.

Koblingsbrett

Et koblingsbrett av typen på figur 2.7 er brukt for enkel oppkobling av kretsen uten bruk av loddebolt.

Figur 2.6: Typen LCD-skjerm brukt i dette arbeidet. Bildet er hentet fra: [17]



Figur 2.7: Koblingsbrett. Bildet er hentet fra: [18]

Kondensator

$470\mu F$ elektrolytiske kondensatorer er inkludert i kretsen. En kondensator er topolet elektrisk komponent som lagrer elektrisk energi [19]. De er koblet i parallell med servomotorene - én til hver motor. Motorene drar tidvis mye strøm og kondensatorene bidrar til å jevne strømpulsene fra strømkilden ved å levere strøm etter behov. Kondensatorene må ha tilstrekkelig kapasitans for å kunne levere tilfredsstillende mengder energi. $470\mu F$ eller større er vanlig i forbindelse med servomotorer av denne størrelsen [20].

2.2 Software

2.2.1 Arduino IDE

Arduino-program kan skrives i hvilket som helst språk med kompilator som produserer binær maskinkode for den aktuelle prosessoren [21]. Arduino har også sitt eget IDE [22] (Integrated Development Environment), som vil bli brukt her, som stammer fra *Wiring* [23] (språk og utviklingsmiljø myntet på programmering av mikrokontrollere).

Programmeringsspråket Arduino er et sett av C/C++-funksjoner og kompileres av en C/C++-kompilator (avr-gcc) [24], slik at alle kodesnutter i C/C++ støttet av avr-gcc fungerer på Arduino.

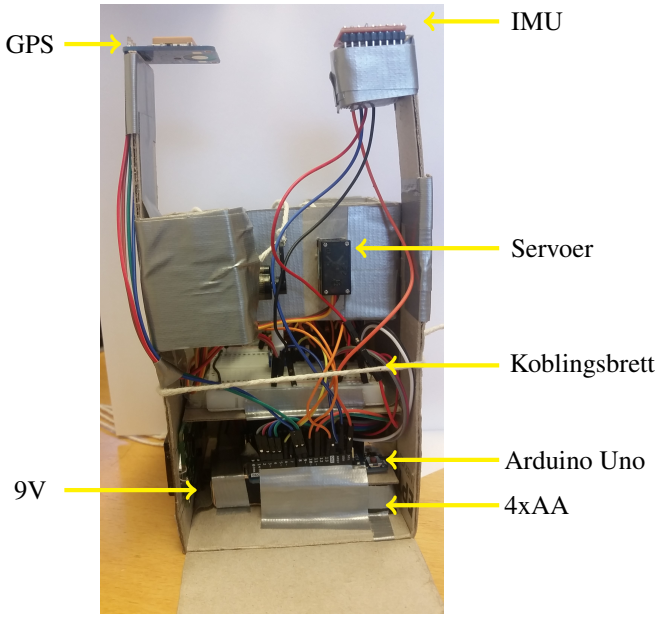
2.2.2 MATLAB

MATLAB er et høy-nivå programmeringsspråk og utviklingsmiljø med et enkelt brukergrensesnitt for numerisk analyse [25].

I dette arbeidet har MATLAB blitt brukt til postprosessering av magnetometerdata i forbindelse med kalibrering og til analyse av innsamlet data fra dropp.

Kapittel 3

Implementasjon



Figur 3.1: Bilde av prototypen brukt i dette arbeidet.

- Dimensjon: 10cm x 10cm x 20cm
- Vekt: 466 gram. Total vekt inkludert fallskjerm og styreliner: 515 gram

Batteriholderen og 9V-batteriet er plassert i bunnen. Disse er de definitivt tyngste komponentene og dette vil holde tyngdepunktet langt nede. Arduino-brettet er plassert rett over

batteriene, deretter koblingsbrett, servomotorer og til slutt sensorene.

3.1 Koblingsskjema

Se figur 3.2 for koblingsskjema. Dokumentasjonsverktøyet *Fritzing* [26] er brukt for å tegne disse oppkoblingene.

3.2 Fallskjerm

Fallskjermen er av typen rundskjerm [27], har en diameter på 102cm og er laget av plast (se figur 3.3). En teorisk landingshastighet for denne diameteren kan estimeres ved hjelp av følgende formel [28]:

$$V = \sqrt{\frac{W}{\rho C_D A}} = \sqrt{\frac{0.515 \cdot 9.81}{1.225 \cdot 0.75 \cdot \pi \cdot 0.51^2}} = 2.6\,\mathrm{m\,s^{-1}}, \tag{3.1}$$

der W [N] er total vekt inkludert fallskjerm, ρ [$\mathrm{kg\,m^{-3}}$] er lufttetthet ved havnivå, C_D er dragkoeffisienten og A [$\mathrm{m^2}$] er arealet av skjermen. Landingshastigheten er innenfor det som karakterises som myk landing [29] og dette sørger for at prototypen tåler påkjenningen fra landing.

Det er klippet et hull i midten av fallskjermen med diameter 7.5cm for at skjermen skal falle mer stabilt og forutsigbart. Styrelinene har samme lengde som diameteren av fallskjermen. Vekt, inkludert styrelinene, er 49 gram.

3.3 Hvordan fungerer den?

Fallskjermen har fire styreliner, og den vil styre i retning av linene som er kortest. Fra de fire styrelinene er det definert åtte retninger som vist i figur 3.3. Merk at retningene her kun er relativt til boksen og ikke kompassretninger. Noen større oppløsning enn åtte retninger er ikke nødvendig da systemet ikke vil være så følsomt for styring. Idéen er å forkorte den/de styrelinen(e) som vil styre fallskjermen i retning punktet man ønsker å treffe basert på dens orienteringen.

Servomotorene veksler mellom to posisjoner som vist i figur 3.4. Maksimalt pådrag vil tilsvare å trekke inn en styreline $2 \cdot 1.75\,\mathrm{cm} = 3\,\mathrm{cm}$. Eksempelvis setter man servoen merket "N" (North) i posisjonen 180° hvis man ønsker å fly rett frem. Ønsker man å styre mot "N-E" settes både servoen merket med "N" og servoen merket med "E" (East) til posisjonen 180°.

3.4 Forklaring av kode

En enkel gjennomgang av koden er vist i figur 3.6 i flytdiagramform. Ellers se vedlegg for koden i sin helhet.

Variabelen *dir* i flytdiagrammet styrer hvilke servomotorer som skal ”på” eller ”av”. Hver retning er definert av et intervall i størrelsesorden 40°. Eksempelvis skal boksen styre rett frem (”N”) om *dir* ligger i intervallet 340°-20° (det vil si 340°-360° og 0°-20°). Det er satt av et dødsoneintervall på 5° mellom hver definerte retning for at servoene ikke skal hoppe for hyppig mellom de definerte retningene. Det vil si at ”N-E” ligger i intervallet 25°-65°, ”E” i 70°-110° etc.

Grensesnittet mot GPS-sensoren er basert på Adafruits Arduino-biblioteker for denne brikken. Samme gjelder for IMU-en og Sparkfun. Disse bibliotekene bidrar til enkel avlesning av sensorene.

Under følger utfyllende informasjon til kode og flytdiagram.

3.4.1 Roll og pitch

Se figur 3.8 for beskrivelse av vinklene.

Akselerometer

Rådataene fra akselerometeret må konverteres og skaleres til akselerasjon. Sparkfun har laget et bibliotek for LSM9DS1 som inkluderer en funksjon som konverterer rådataene til g-krefter/akselerasjon. Når sensoren står plant skal sensoren gi 0 g i x-og y-retning, og 1 g i z-retning. Deretter kalibreres akselerometeret ved å notere avvikene fra disse verdiene og trekke disse fra målingene. Roll og pitch kan deretter estimeres med formlene

$$\tan \phi_a = \frac{G_y}{G_z}, \tag{3.2}$$

og

$$\tan \theta_a = \frac{-G_x}{\sqrt{G_y^2 + G_z^2}}, \tag{3.3}$$

hvor G_x, G_y og G_z er x-, y- og z-komponentene av gravitasjonskraften, henholdsvis [30].

Gyroskop

Gyroskopet gir oss vinkelhastigheten rundt x-, y- og z-aksene i $[deg/s]$. Det betyr at for å bestemme roll og pitch må disse vinkelratene integreres:

$$\phi_g(t) = \int_0^t \dot{\phi_g} dt, \tag{3.4}$$

$$\theta_g(t) = \int_0^t \dot{\theta_g} dt, \tag{3.5}$$

For å implementere dette i Arduino-koden kan man diskretisere ved hjelp av Eulers metode.

$$\phi_g[n] = \phi_g[n-1] + h\phi'_g[n], \tag{3.6}$$

$$\theta_g[n] = \theta_g[n-1] + h\theta'_g[n], \tag{3.7}$$

hvor h er tida mellom hver telling. Denne approksimasjonen kan naturligvis introdusere noe feil i posisjonsmålingene når gyroskopdataene forandrer seg hurtigere enn tellefrekvensen.

3.4.2 Komplementærfilter

Problemet med å bruke akselerometeret til å estimere roll og pitch er at akselerometeret måler lineær akselerasjon, det vil si alle kreftene som virker på sensoren, og ikke bare gravitasjonsfeltet. Vibrasjoner, kjappe retningsforandringer, støt og fall vil alle gi kraftige utslag på målingene slik at roll- og pitch-estimatene blir ubrukelige. Målingene av gravitasjonsfeltet er et lavfrekvent signal og kan i teorien isoleres fra høyfrekvente signaler (vibrasjoner, støt, etc.) ved å lavpassfiltere målingene.

Et lavpassfilter er et filter som slipper gjennom signaler under en viss frekvens (knekkfrekvens) og blokkerer signaler over denne frekvensen. Merk - dette er i det ideelle tilfellet. I praksis så dempes amplituden på signalene over denne knekkfrekvensen slik at vi fortsatt vil ha en viss "forstyrrelse" fra disse i målingene.

Gyroskop fungerer veldig godt ved dynamiske bevegelser, men vinkelmålingene vil drifte over tid. Dette er fordi gyroskopet er utsatt for lavfrekvent støy som akkumuleres i utregningene når vinkelratene integreres. For å kompensere for dette kan signalene fra gyroskopet høypassfiltreres. Et høypassfilter slipper gjennom signaler over en viss frevens og demper signaler under denne grensen. Det fungerer derfor godt til å filtrere gyroskopdataene.

Ved å kombinere informasjonen fra akselerometer og gyroskop oppnår man en bedre representasjon av roll og pitch. På kort sikt vil gyroskopet gi et godt estimat av vinklene. Akselerometeret prioriteres på lengre sikt og vil kunne kompensere for gyroskopdrift. Med et komplementærfilter kan man kombinere høypassfiltrert gyroskopdata med lavpassfiltrert

akselerometerdata, se figur 3.7. Diskretisert vil komplementærfilteret ta følgende form (se appendiks B):

$$y_{kf}[n] = \alpha y_{kf}[n-1] + (\alpha - 1)y_a[n] + \alpha(y_g[n] - y_g[n-1]), \tag{3.8}$$

hvor y_a og y_g henholdvis er dataene som skal lavpassfiltreres (akselerometer) og høypassfiltreres (gyroskop) og $0 \leq \alpha \leq 1$ er en utjevningsfaktor gitt av knekkfrekvensen [31]. Denne diskrete versjonen lar seg enkelt implementere i Arduino-kode.

3.4.3 Yaw

Se figur 3.8 for beskrivelse av yaw.

Magnetometer

Det lokale magnetfeltet avhenger av jordas magnetfelt og alle magnetfelt fra omkringliggende objekter. For å få fornuftige verdier fra magnetometeret må målingene kompenseres for hard- og mykjernseffekter. Se appendiks A for metode og resultat av kalibrering av magnetometeret.

Tiltkompensering

Når kompasset holdes plant finner man kompassretning enkelt ved formelen

$$\psi = \operatorname{atan2}\left(\frac{m_y}{m_x}\right), \tag{3.9}$$

der ψ er yaw/heading, m_x og m_y er x- og y-komponentene av magnetfeltet. Når kompasset ikke lenger står plant, det vil si roll- og/eller pitch-vinkelen er ulik null, så endres m_x og m_y og dermed også kompassretningen. For å kompensere for vinkelendring i roll- og pitch må man finne horisontalkomponentene m_x^h og m_y^h av m_x og m_y.

Ved å utnytte egenskapene ved rotasjonsmatriser [32] kan man representere horisontalkomponentene slik: (Rekkefølgen av rotasjonsmatrisene er vilkårlig, men det omvendte produktet vil gi andre horisontalkomponenter. De tjener derimot samme hensikt.)

$$\begin{aligned}
\begin{bmatrix} m_x^h \\ m_y^h \\ m_z^h \end{bmatrix} &= R_{y,\theta} R_{x,\phi} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \\
&= \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \\
&= \begin{bmatrix} cos(\theta) & sin(\theta)sin(\phi) & sin(\theta)cos(\phi) \\ 0 & cos(\phi) & -sin(\phi) \\ -sin(\theta) & cos(\theta)sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}
\end{aligned} \tag{3.10}$$

Trekker man ut m_x^h og m_y^h får man

$$m_x^h = m_x cos(\theta) + m_y sin(\theta) sin(\phi) + m_z sin(\theta) cos(\phi), \tag{3.11}$$

$$m_y^h = m_y cos(\phi) - m_z sin(\phi). \tag{3.12}$$

(3.11) og (3.12) kan man sette inn i (3.9) for å få roll- og pitch-kompensert kompassmåling. I tillegg må effektene av magnetisk misvisning kompenseres for. Magnetisk misvisning (magnetisk deklinasjon) er avviket mellom retningen til den geografiske nordpolen og magnetisk nord [33]. I Trondheim er denne vinkelen +3.4°, og denne må trekkes fra utregningen av yaw.

Gyroskopestimat og sensorfusjon

Et yaw-estimat kan også genereres ved hjelp av gyroskopet på lik linje med gyroskopet roll- og pitch-estimat. Denne målingen kan fusjoneres med den tiltkompenserte kompassmålingen i et komplementærfilter, på samme måte som for roll og pitch, for å gjøre yaw-målingen mer robust mot forstyrrelser (se område med grå bakgrunn i flytdiagram 3.6). De første testene fra denne implementasjonen har ikke gitt gode resultater. Dette filteret har blitt implementert sent i prosessen og det har derfor ikke vært tid til å forfølge dette videre. Resultater fra dette vil inkluderes i vedlegget, men ikke i rapporten.

3.4.4 GPS fix

For at en GPS skal kunne fastsette sin posisjon i 3D må den ha kontakt med fire eller flere satellitter. ”GPS fix” i flytdiagrammet 3.6 er en boolsk variabel som er sann når sensoren er låst til signalet fra fire eller flere satellitter og usann ellers.

3.4.5 Bearing

Bearing er en essensiell del av denne implementasjonen, da det er forskjellen i yaw og bearing som definerer inngangssignalet til servomotorene. Avhengig av kontekst kan bearing bety flere ting, men her brukes det i betydningen vinkelen fra retning nord til et gitt punkt som observert fra punktet man befinner seg [35], se figur 3.9.

Formelen for å regne ut bearing fra ett GPS-punkt til et annet [36] er

$$\mathrm{atan2}(a, b), \tag{3.13}$$

hvor

$$a = \sin(\Delta\lambda)\cos(\gamma_2), \tag{3.14}$$

$$b = \cos(\gamma_1)\sin(\gamma_2) - \sin(\gamma_1)\cos(\gamma_2)\cos(\Delta\lambda). \tag{3.15}$$

γ_1 og λ_1 er breddegrad og lengdegrad i startpunktet, γ_2 og λ_2 er breddegrad og lengdegrad i sluttpunktet og $\Delta\lambda$ er forskjellen i lengdegrad mellom de to punktene.

3.5 Kostnader

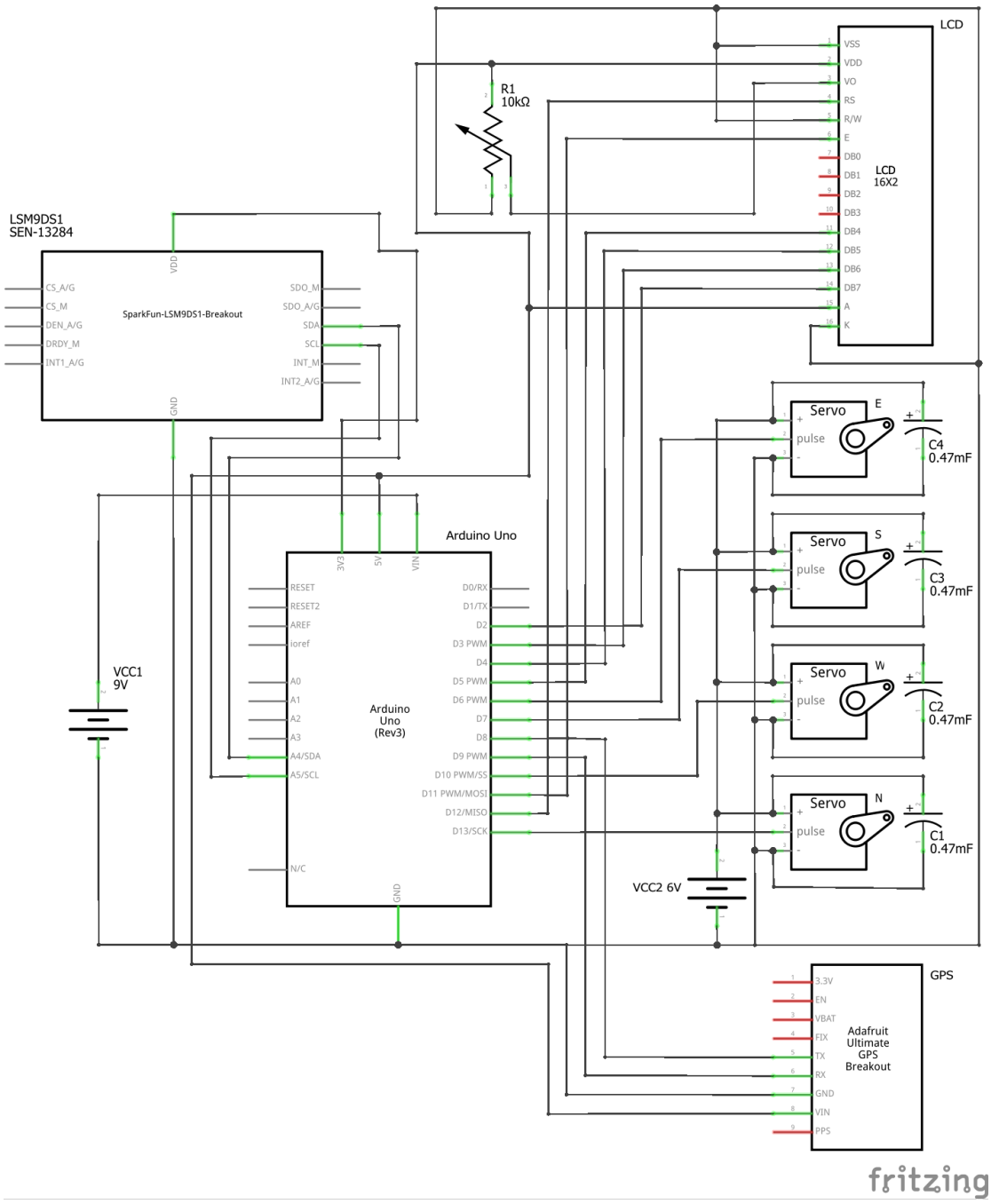
Prisene er hentet rett fra leverandør der det var mulig. Kurs er hentet fra Norges Bank (oppdatert 18.05.2017):

- Euro (EUR): 9.4113
- Amerikanske dollar (USD): 8.4566

Komponent:	Pris	Pris (nok)
Arduino Uno [37]:	€20	= 188.23 nok
Adafruit Ultimate GPS [13]:	$39.95	= 337.84 nok
SparkFun 9DoF IMU Breakout [15]:	$24.95	= 210.99 nok
Tower Pro MG90S [38]:	$3.99x4 = $15.96	= 134.97 nok
	SUM	= 872.0 nok

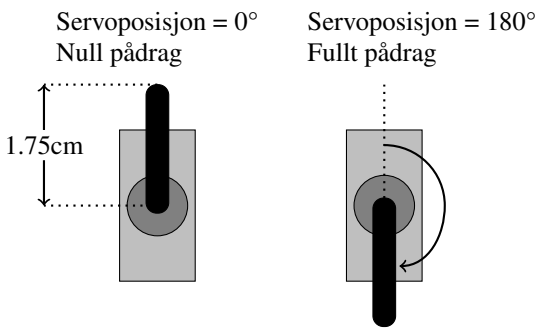
Tabell 3.1: Utstyrskostnader

Annet utstyr som koblingsbrett, batterier og LCD er utelatt fra regnskapet. LCD er kun inkludert i prosjektet for enkle tilbakemeldinger fra sensorer under utviklingsarbeidet og representerer derfor ikke noen kostnad for selve kontrollsystemet. Kostnadene ved resten av utstyret er vurdert til å være neglisjerbare.

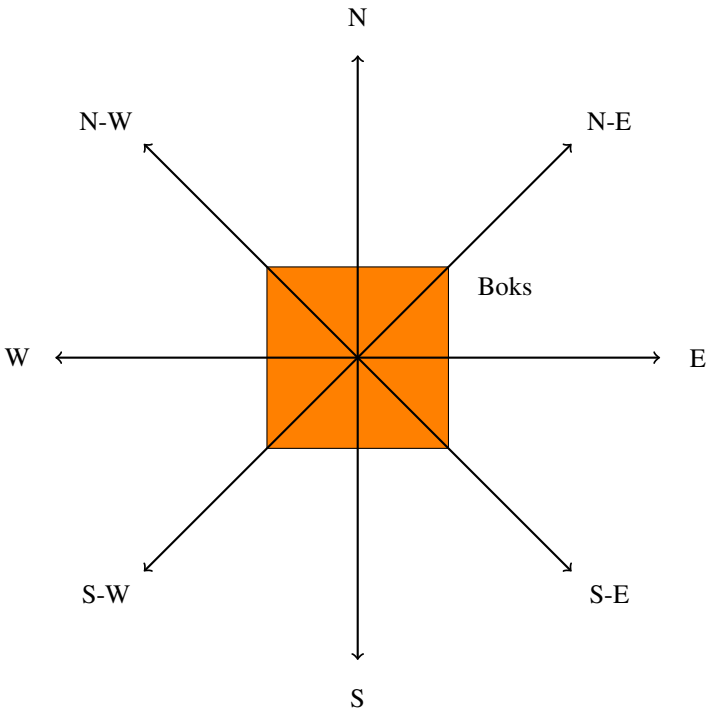


Figur 3.2: Koblingsskjema.

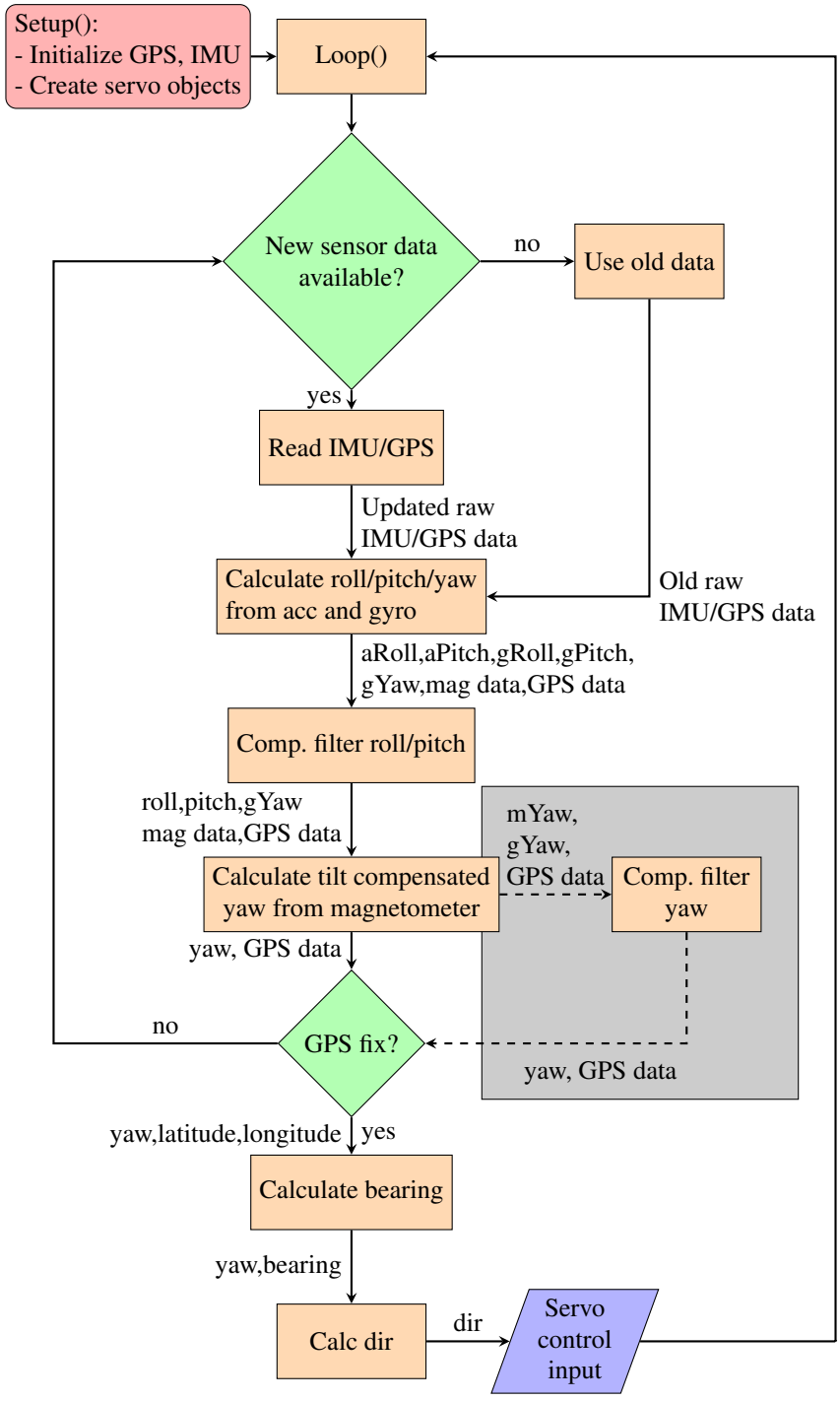
Figur 3.3: Fallskjerm med 4 styreliner.



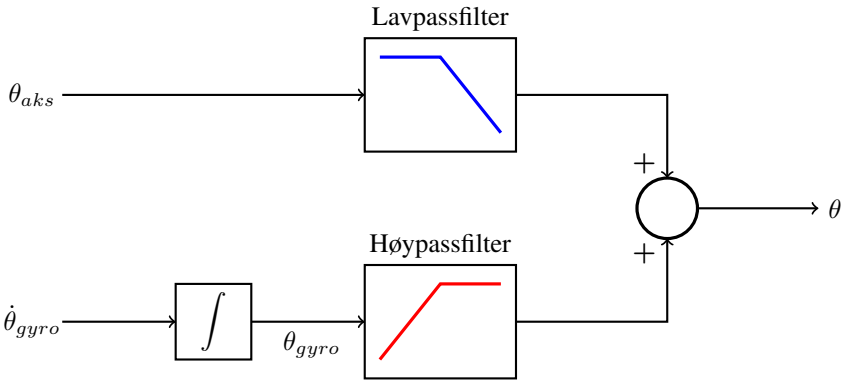
Figur 3.4: Servoposisjoner for null og fullt pådrag.



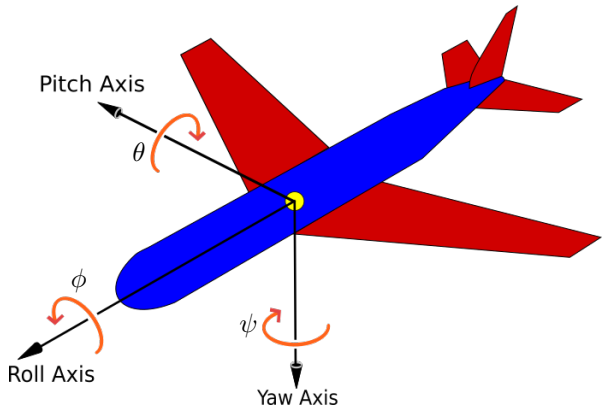
Figur 3.5: De åtte definerte retningene som enheten kan styre mot.



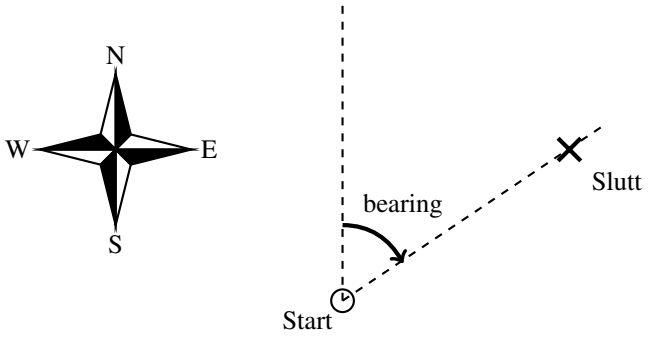
Figur 3.6: Flytdiagram av Arduino-kode.



Figur 3.7: Komplementærfilter brukt på akselerometer- og gyroskopdata



Figur 3.8: Beskrivelse av rotasjonene roll, pitch og yaw. Bildet er hentet fra: [34]



Figur 3.9: Beskrivelse av bearing.

Kapittel 4

Testing

4.1 Testmiljø

Testingen er gjort fra taket på bygget Elektro B på Gløshaugen. Dropphøyden her er omtrent 25 meter og dette ga en dropptid på rundt 9 sekunder.

4.2 Testdesign og prosedyre

Boksen slippes med fullt utslått fallskjerm når GPS-sensoren har fått kontakt med tilstrekkelig antall satelitter og funnet sin posisjon.

Kode er skrevet for å detektere når fallskjermen blir sluppet og den skriver utvalgte data til Arduino Unos EEPROM under droppet. Dataene som skrives til minnet er breddegrad, lengdegrad, yaw, roll og pitch. På grunn av begrenset minne er avstand til mål og bearing beregnet fra breddegrad og lengdegrad i etterkant.

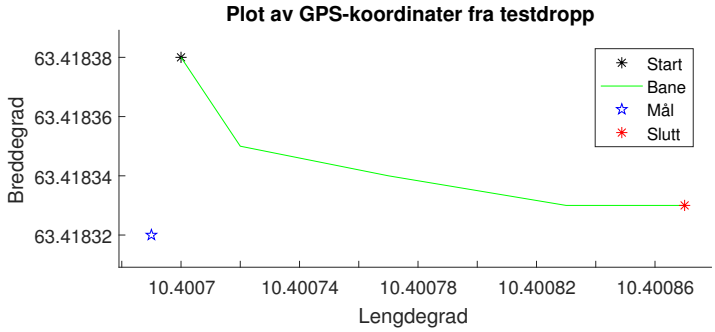
4.3 Egenskaper som skal testes

Sensorstabilitet og pålitelighet under dropp skal undersøkes. Altså i hvilken grad man kan stole på sensorverdiene som reelle representasjoner av det de skal måle. Posisjonsmåling fra GPS-sensoren samt roll, pitch og yaw fra IMU-en er de aktuelle sensordataene.

4.4 Resultater

4.4.1 Test #1

Se figur 4.1-4.5 for testresultater. Se også figur 4.11 for alternativ presentasjon av GPS-data.



Figur 4.1: GPS-data test #1. Banen har retning vekk fra målpunktet.

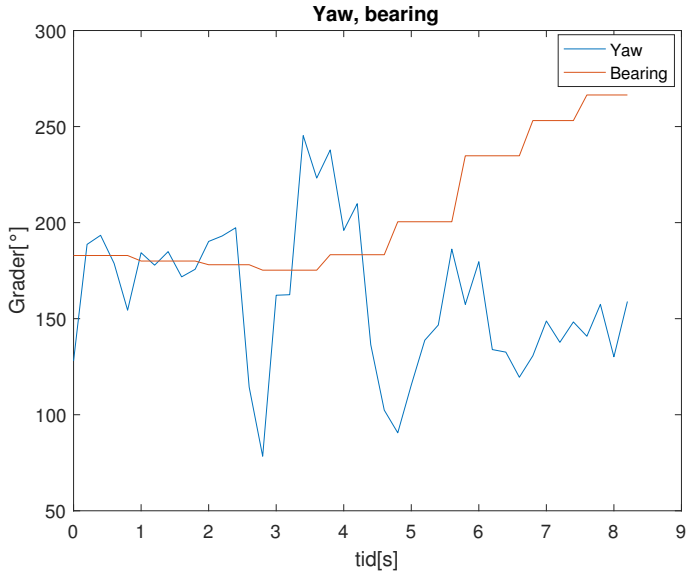
GPS-sensoren leverer pålitelig informasjon under dropp og klarer fint å bestemme sin posisjon i forhold til det forhåndsdefinerte punktet. Som følge av det gir kalkuleringen av avstand og bearing forventede verdier. Sensorinformasjon sammenfaller med det som er observert under testing.

Roll og pitch holder seg innenfor rimelige verdier i test #1, men signalet inneholder mer støy enn det som er observert under testing. Yaw opplever et stort dropp etter ca. 2.5 sekunder (figur 4.2) som sammenfaller med en stor ending i pitch (figur 4.5) (samme etter ca. 4 sekunder som sammenfaller med stor endring i roll (figur 4.4)). Dette indikerer at tiltkompenseringen ikke fungerer optimalt. Signalene er generelt støyete.

Fallskjermen klarer ikke å styre seg mot målpunktet.

4.4.2 Test #2

Se figur 4.6-4.10 for testresultater. Se også figur 4.12 for alternativ presentasjon av GPS-data.

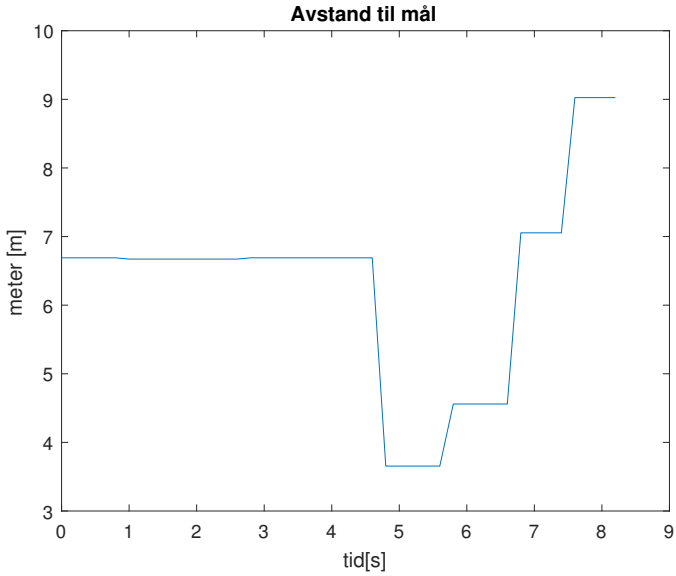


Figur 4.2: Yaw og bearing test #1. Mye støy i yaw-signalet i tillegg til to store dropp (etter ca. 2.5 og 4 sekunder) som sammenfaller med store endringer i roll- og pitch-målingene. Dette tyder på dårlig tiltkompensering.

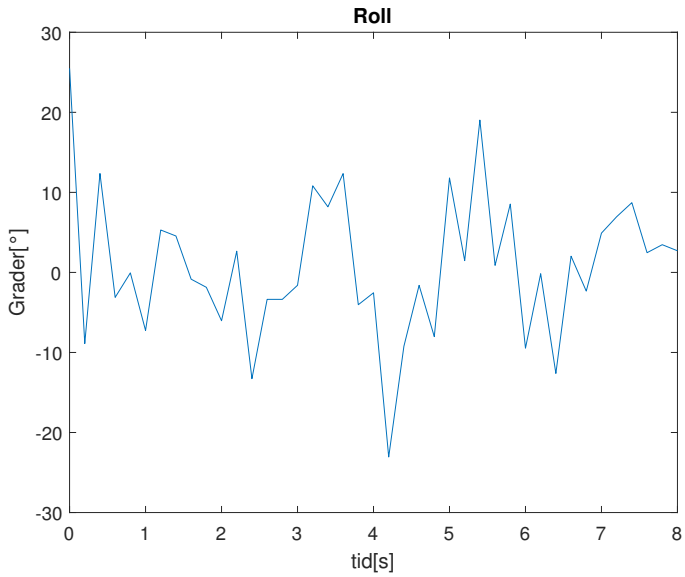
GPS-sensoren leverer igjen pålitelig informasjon under dropp og klarer fint å bestemme sin posisjon i forhold til det forhåndsdefinerte punktet. Som følge av det gir kalkuleringen av avstand og bearing forventede verdier. Sensorinformasjon sammenfaller med det som er observert under testing.

Test #2 viser kjempesprang i yaw på ca. 300° (figur 4.7) etter ca. ett sekund og etter tre sekunder. Sistnevnte sammenfaller med med stor endring av roll (figur 4.9). Store utslag i både roll- og pitch-målinger som ikke er observert under testing. Målingene har generelt mye støy.

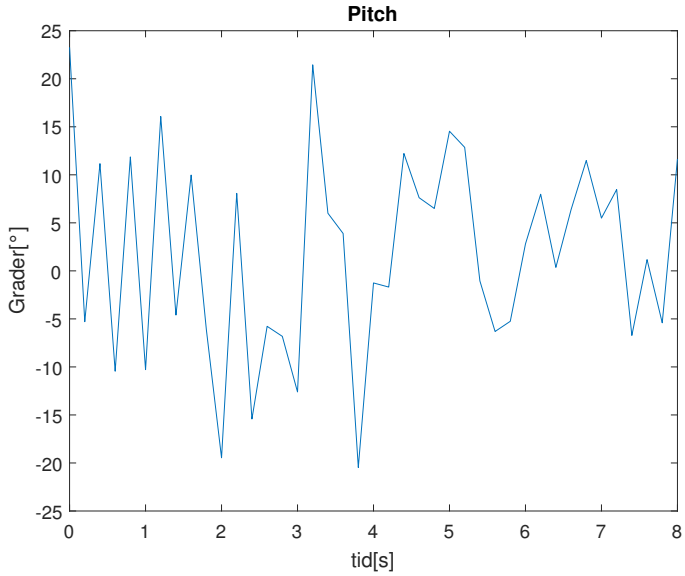
Fallskjermen har retning mot målpunktet, men IMU-dataene er for lite pålitelige til å avgjøre om dette er aktiv styring eller ikke.



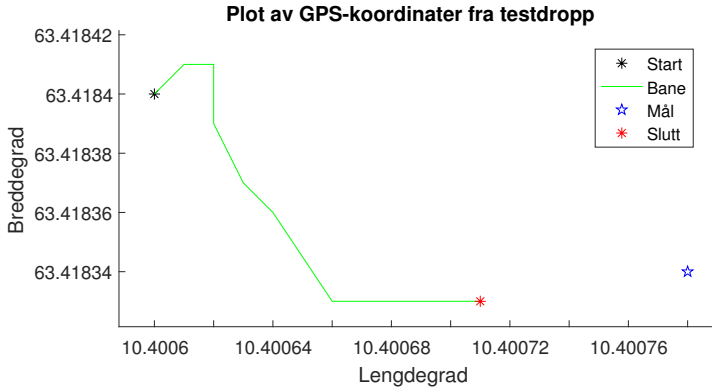
Figur 4.3: Avstand til mål test #1.



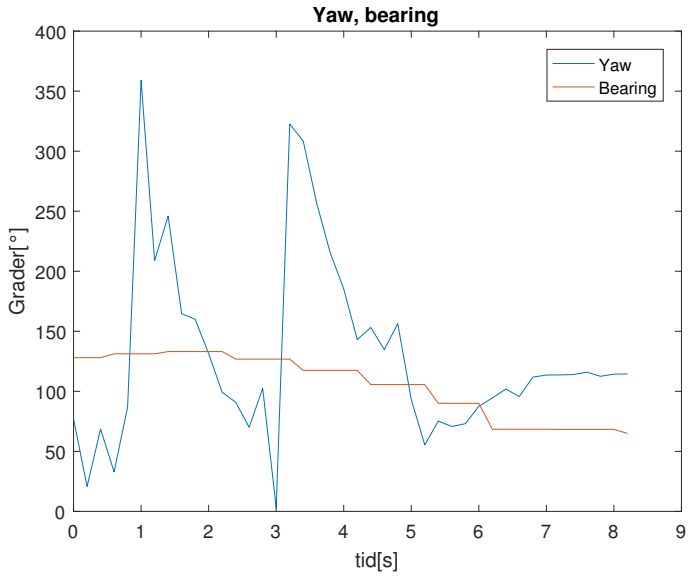
Figur 4.4: Roll test #1. Målingen er støyete og har et markant dropp etter ca. 4 sekunder som påvirker yaw-målingen i stor grad.



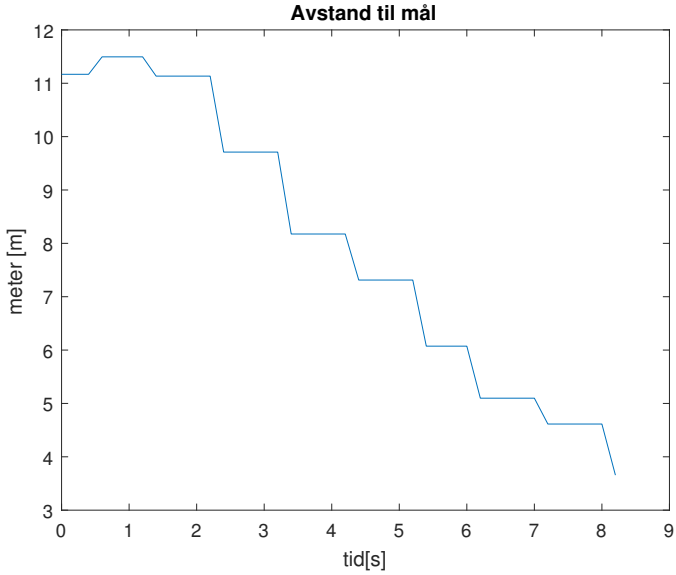
Figur 4.5: Pitch test #1. Målingen er støyete og har et par markante endringer som setter spor også i yaw-målingen.



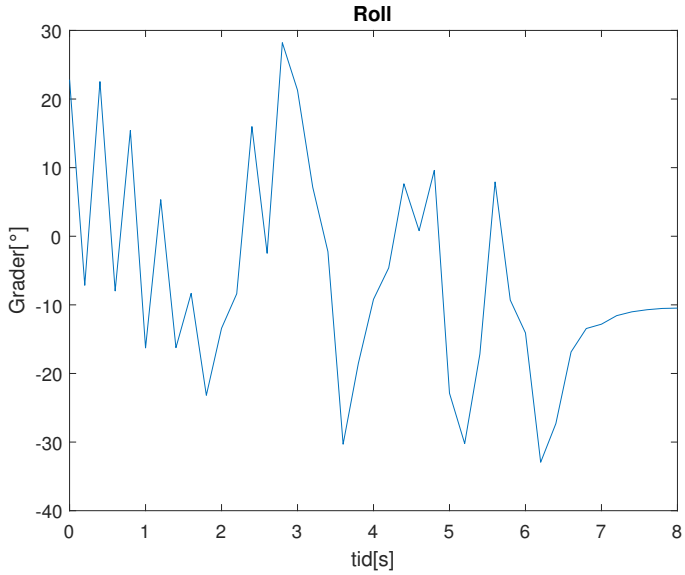
Figur 4.6: GPS-data test #2. Banen har retning mot målpunktet.



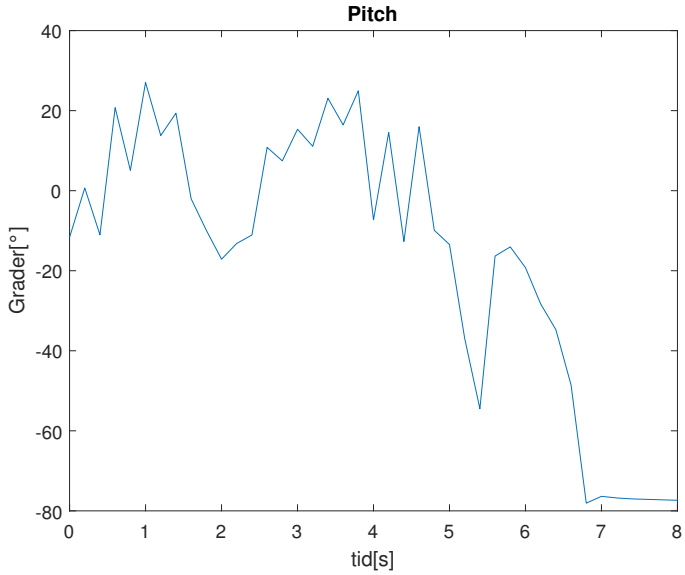
Figur 4.7: Yaw og bearing test #2. Store utslag etter 1 sekund og 3 sekunder der sistnevnte sammenfaller med stor endring i roll-vinkel. Dette tyder på dårlig tiltkompensering. Målingen er generelt støyete.



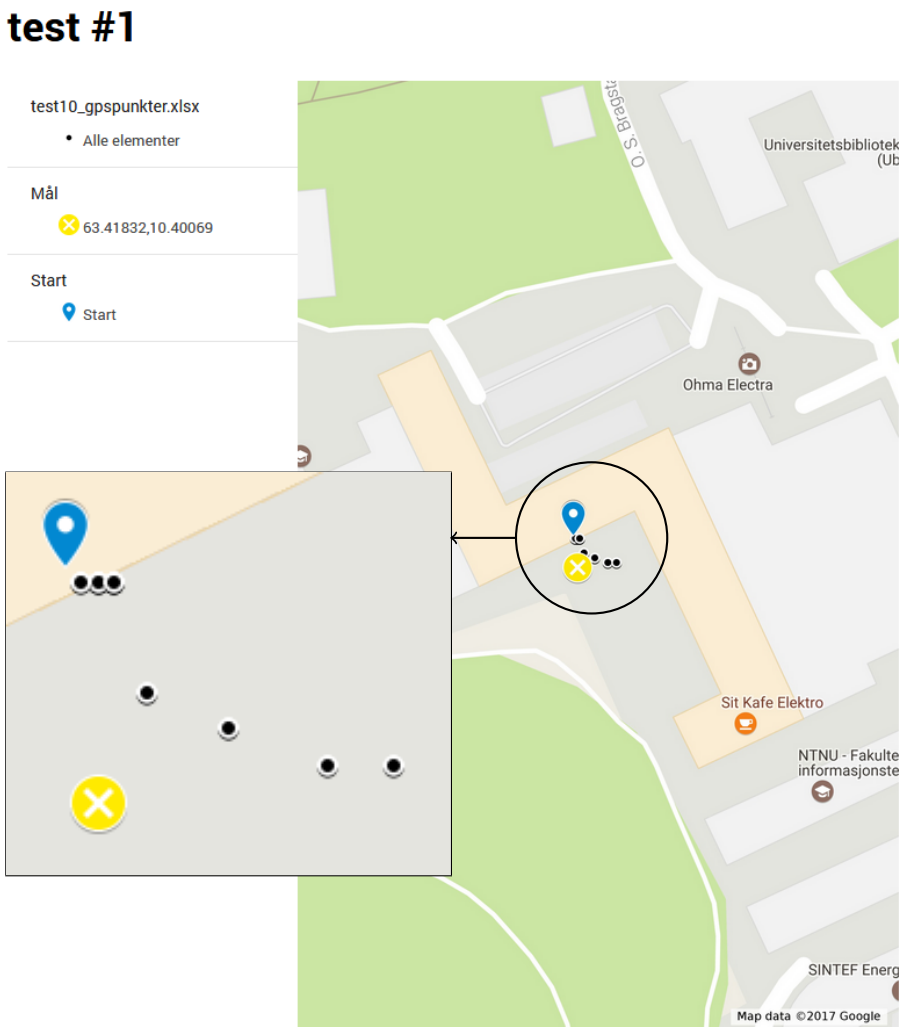
Figur 4.8: Avstand til mål test #2. Avstanden synker rimelig monotont som stemmer godt overens med plottet av GPS-koordinatene.



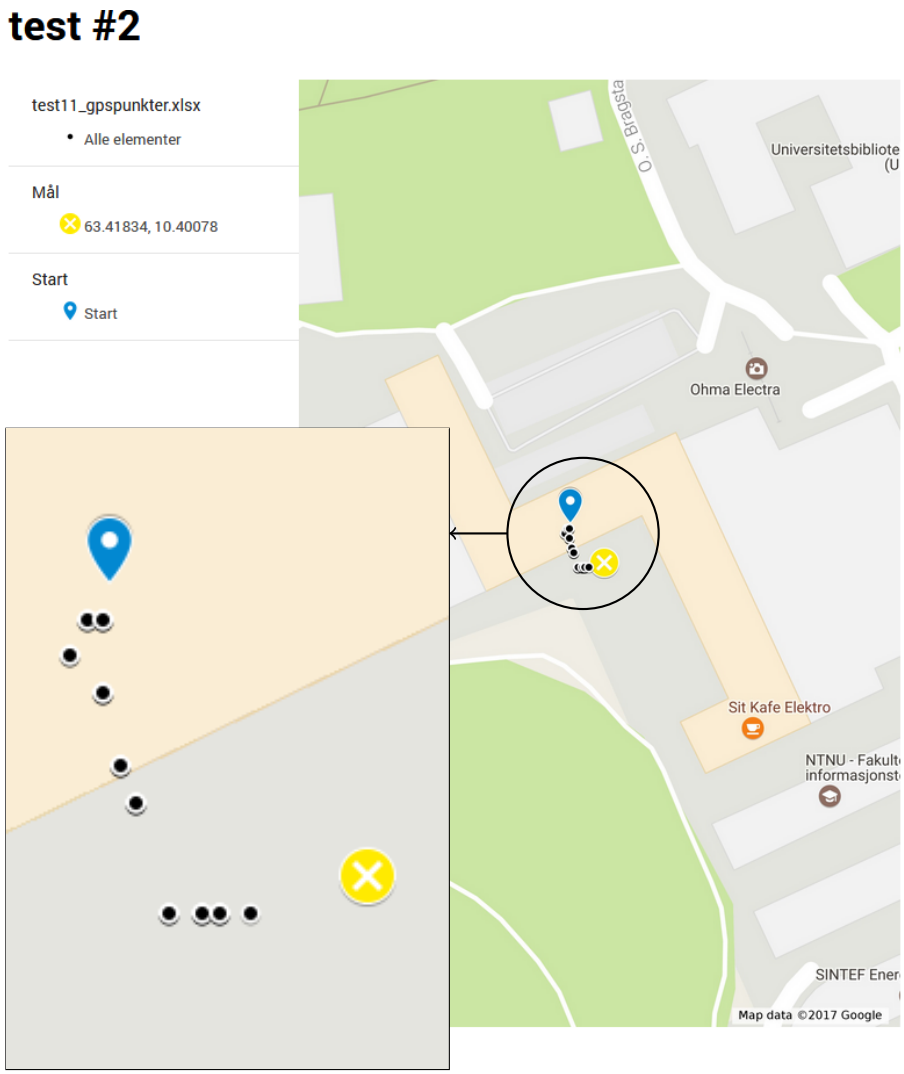
Figur 4.9: Roll test #2. Generelt støyete måling og flere store utslag som ikke er observert under testing. Dette vil igjen påvirke yaw-estimatet.



Figur 4.10: Pitch test #2. Generelt støyete måling. Stor endring i yaw-estimat sammenfaller med økning av pitch rundt 1 sekund. Tyder igjen på dårlig tiltkompensering.



Figur 4.11: GPS-data fra andre test #1. Her presentert med applikasjonen Google My Maps [39].



Figur 4.12: GPS-data fra andre test #2. Her presentert med applikasjonen Google My Maps [39].

Kapittel 5

Diskusjon

Kostnadene forbundet med utvikling av denne prototypen er lave, men det er lett å se for seg store sprang i kostnader ved oppskalering. Sensor- og mikrokontrollerkostnad vil holdes relativt uendret. I hovedsak vil kostnadsspranget være knyttet til større motorer og større batterier.

Testresultatene viser at GPS-sensoren leverer god informasjon under testing, men støyete IMU-data gjør derimot at styringen blir uforutsigbar.

Det er fortsatt mye arbeid som gjenstår for å utvikle et system som kan gjøre gode beslutninger basert på pålitelig sensordata under påvirkning av et utfordrende testmiljø.

5.1 Anbefalinger og videre arbeid

5.1.1 Fallskjerm

Den fallskjermen som er laget i forbindelse med dette prosjektet innfører naturligvis en del usikkerhet. Man kan ikke forvente at denne enkle konstruksjonen skal ha veldig gode egenskaper med tanke på styrbarhet. Skjermdesignet bør også vurderes videre. Rundskjermer har begrensede manøvreringsmuligheter som følge av at de ikke genererer løft [27], men ble foretrukket til dette prosjektet av praktiske hensyn. Den mer manøvrerbare firkantskjermen kan være et design å se videre på.

5.1.2 Motorer

Servomotorene brukt i dette prosjektet kan ha vært for små. Pådraget gitt til styrelinene var maks av servomotorenes kapasitet. Muligheten for å kunne sette et større pådrag på

styrelinene kan være nødvendig for å se bedre resultater. Et naturlig steg vil være å bruke større og kraftigere servomotorer, men også se på andre metoder for å trekke i styrelinene. Steppermotorer og "continuous rotation"-servomotor kan brukes til å implementere en vinsj-type virkemåte. Pådragsmessig vil man da ha større fleksibilitet.

5.1.3 Testmiljø

For å kunne se i større grad hvordan systemet presterer bør dropphøyden økes. Her er dropphøyden og falltiden såpass begrenset at det blir vanskelig å se systemets respons. Responsen ved overskyt ville for eksempel vært interessant å observere ved større høyder. Store lokale variasjoner i magnetfelt er også registert i testmiljøet - noe som naturligvis kan påvirke magnetometeret og beregningen av yaw.

5.1.4 Filterdesign

Som man ser av resultatene så er sensormålingene ganske støyete. I dette tilfellet er driftsmiljøet ganske tøffe for sensorene. De blir utsatt for fritt fall, kjappe retningsforandringer, vibrasjoner og elektrisk støy fra motorer og lokale variasjoner i magnetfelt.

Ved videre arbeid bør fysisk skjerming og filterdesign vurderes. Komplementærfilter ble implementert i dette tilfellet fordi det er ansett som et effektivt filter og blir ofte brukt til estimering av orientering av UAV [40; 41]. Det er enkelt å implementere og tar lite datakraft. Dette gjør filteret godt egnet til små, innebygde datasystemer.

Det kan være verdt å undersøke muligheten for å implementere et Kalmanfilter. Dette filteret er også et hyppig brukt filter i denne sammenhengen [42; 43]. Det er tyngre å implementere og matematisk mye mer komplekst. I korte trekk vil Kalmanfilteret bruke en serie målinger observert over tid, som kan inneholde støy og andre unøyaktigheter, og beregner et statistisk optimalt estimat av verdien. Dette setter større krav til regnekraft slik at Arduino Unos 8-bit mikroprosessor blir for ueffektiv.

5.1.5 Mikrokontroller

Som nevnt i kap. 5.1.4 bør det oppgraderes fra en 8-bit mikrokontroller for å effektivt kunne prosessere sensordata gjennom et Kalmanfilter. Oppgraderingen vil også gjøre at man kan operere med større presisjon av flyttall. Man kan da representere breddegrad og lengdegrad fra GPS-sensoren enda mer presist og større presisjon gir også mindre feil som følge av avrunding. Presisjonen var ikke kritisk i dette prosjektet, men det kan utgjøre en forskjell i de situasjoner hvor fallskjermen flyter rett over målpunkt.

Arduino Unos ATmega328P med sine 32kB flash-minne holdt mer eller mindre akkurat til denne implementasjonen. Dette ble ikke et problem i dette prosjektet, men ustabilitetsproblemer kan forekomme når man nærmer seg minnekapasitet. Skal for eksempel et Kalmanfilter implementeres så må flash-minnet økes.

5.1.6 Hastighetsregulering

I de tilfellene der fallskjermen kommer inn over målpunktet med relativt stor hastighet så vil man få et stort overskyt. Slik denne prototypen fungerer så vil den i teorien styre med fullt pådrag mot målpunktet helt til den oppdager av den er forbi. Det vil si enheten vil fly et stykke forbi målpunktet før den igjen snur. Dette kan reguleres slik at pådraget blir mindre og mindre jo nærmere man er målet, men også ved å bremse mer aktivt ved å reversere pådraget. Fra GPS-sensoren kan man hente ut både bakkefart og høydemeter. Vet man høyden til destinasjonspunktet kan både høydemeter og bakkefart fra GPS-sensoren brukes som inngangssignal til en slik reguleringssløyfe for å begrense overskyt.

Bibliografi

[1] "Paraweather AS," 2017. URL: https://www.paraweather.no, [Online; urldato 24.04.2017].

[2] S. Eisenträger, "Slik får FN nødhjelp inn i Syria," 2016. URL: http://www.vg.no/nyheter/utenriks/syria/slik-faar-fn-noedhjelp-inn-i-syria/a/23631814/, [Online; urldato 04.04.2017].

[3] Wikipedia, "Joint Precision Airdrop System," 2017. URL: https://en.wikipedia.org/wiki/Joint_Precision_Airdrop_System, [Online; urldato 02.05.2017].

[4] A. B. Hall, "Conceptual and preliminary design of a low-cost precision aerial delivery system," Master's thesis, Naval Postgraduate School, 2016.

[5] Draper, "Aerial Delivery Without GPS," 2016. URL: http://www.draper.com/news/aerial-delivery-without-gps, [Online; urldato 02.05.2017].

[6] Dr. Gareth Evans, "JPADS: circumventing GPS for next-gen precision airdrops," 2016. URL: http://www.airforce-technology.com/features/featurejpads-circumventing-gps-for-next-gen-precision-airdrops-4872436/, [Online; urldato 28.05.2017].

[7] M. R. Wuest and R. J. Benney, *Precision Airdrop (Largage de précision)*, vol. 24 of *Flight Test Techniques Series*. RTO/NATO, 2005. Tilgjengelig fra: http://www.dtic.mil/dtic/tr/fulltext/u2/a455121.pdf".

[8] Arduino, "Arduino Uno," 2017. URL: https://www.arduino.cc/en/Main/ArduinoBoardUno, [Online; urldato 20.04.2017].

[9] Atmel, "Atmega328p datasheet," 2015. URL: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf, [Online; urldato 25.04.2017].

[10] Arduino.org, "Arduino Uno," 2017. URL: http://www.arduino.org/products/boards/arduino-uno, [Online; urldato 26.04.2017].

[11] Arduino, "Float," 2017. URL: https://www.arduino.cc/en/Reference/Float, [Online; urldato 30.04.2017].

[12] Wikipedia, "Haversine formula," 2017. URL: https://en.wikipedia.org/wiki/Haversine_formula, [Online; urldato 18.05.2017].

[13] Adafruit, "Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3," 2017. URL: https://www.adafruit.com/product/746, [Online; urldato 20.04.2017].

[14] Wikipedia, "GPS navigation device," 2017. URL: https://en.wikipedia.org/wiki/GPS_navigation_device#Sensitivity, [Online; urldato 29.04.2017].

[15] Sparkfun, "SparkFun 9DoF IMU Breakout - LSM9DS1," 2017. URL: https://www.sparkfun.com/products/13284, [Online; urldato 15.05.2017].

[16] Tower Pro Pte Ltd, "Tower Pro MG90S," 2017. URL: http://www.towerpro.com.tw/product/mg90s-3/, [Online; urldato 20.04.2017].

[17] Simon Monk, "Hello World!," 2017.

[18] Simon Monk, "Breadboard," 2015. URL: https://learn.adafruit.com/lesson-0-getting-started/breadboard, [Online; urldato 02.04.2017].

[19] Wikipedia, "Kondensator (elektrisk)," 2017. URL: https://no.wikipedia.org/wiki/Kondensator_(elektrisk), [Online; urldato 26.05.2017].

[20] Simon Monk, "If the Servo Misbehaves," 2015. URL: https://learn.adafruit.com/adafruit-arduino-lesson-14-servo-motors/if-the-servo-misbehaves, [Online; urldato 27.04.2017].

[21] Wikipedia, "Arduino # Software development," 2017. URL: https://en.wikipedia.org/wiki/Arduino#Software_development, [Online; urldato 26.05.2017].

[22] Arduino, "Download the Arduino IDE," 2017. [Software]. Tilgjengelig på: https://www.arduino.cc/en/Main/Software.

[23] Wiring, "Wiring," 2017. URL: http://wiring.org.co/, [Online; urldato 26.05.2017].

[24] Arduino, "Arduino Build Process," 2016. URL: https://www.arduino.cc/en/Hacking/BuildProcess, [Online; urldato 28.04.2017].

[25] MathWorks, "MATLAB," 2017. [Software]. Tilgjengelig på: https://se.mathworks.com/products/matlab.html.

[26] University of Applied Sciences Potsdam, "Fritzing (Versjon 0.9.3b)," 2016. [Software]. Tilgjengelig på: https://www.fritzing.org.

[27] Wikipedia, "Fallskjerm," 2017. URL: https://no.wikipedia.org/wiki/Fallskjerm, [Online; urldato 24.05.2017].

[28] Dr. Jean Potvin, "Calculating the descent rate of a round parachute." URL: http://www.pcprg.com/rounddes.htm, [Online; urldato 02.06.2017].

[29] J. Whitting, J. Steele, M. Jaffrey, and B. Munro, "Parachute landing fall characteristics at three realistic vertical descent velocities," *Aviat Space Environ Med.*, 2007. PubMed PMID: 18064918. Tilgjengelig fra: https://www.ncbi.nlm.nih.gov/pubmed/18064918.

[30] Freescale Semiconductor, "Tilt Sensing Using a Three-Axis Accelerometer," 2013. URL: http://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf, [Online; urldato 24.05.2017].

[31] Wikipedia, "Low-pass filter," 2017. URL: https://en.wikipedia.org/wiki/Low-pass_filter, [Online; urldato 08.05.2017].

[32] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons Inc, 2005.

[33] Kartverket, "Magnetisk misvisning," 2016. URL: http://www.kartverket.no/Kunnskap/Posisjon-og-navigasjon/Slik-finner-du-kompasskursen/Magnetisk-misvisning/, [Online; urldato 26.05.2017].

[34] Wikipedia, "Aircraft principal axes," 2017. URL: https://en.wikipedia.org/wiki/Aircraft_principal_axes, [Online; urldato 02.05.2017].

[35] Wikipedia, "Bearing (Navigation)," 2017. URL: https://en.wikipedia.org/wiki/Bearing_(navigation), [Online; urldato 28.04.2017].

[36] Chris Veness, "Calculate distance, bearing and more between Latitude/Longitude points," 2017. URL: http://www.movable-type.co.uk/scripts/latlong.html, [Online; urldato 25.05.2017].

[37] Arduino, "Arduino Uno REV3," 2017. URL: https://store.arduino.cc/arduino-uno-rev3, [Online; urldato 19.05.2017].

[38] Value Hobby, "Towerpro MG90S Metal Gear Micro Servo," 2017. URL: http://www.valuehobby.com/mg90s-mini-servo.html, [Online; urldato 19.05.2017].

[39] Google, "Google My Maps," 2017. URL: https://www.google.com/mymaps, [Online; urldato 12.05.2017].

[40] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing uav," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 340–345, Sept 2008.

[41] A. J. Baerveldt and R. Klang, "A low-cost and low-weight attitude estimation system for an autonomous helicopter," in *Proceedings of IEEE International Conference on Intelligent Engineering Systems*, pp. 391–395, Sep 1997.

[42] H. G. de Marina, F. Espinosa, and C. Santos, "Adaptive UAV Attitude Estimation Employing Unscented Kalman Filter, FOAM and Low-Cost MEMS Sensors," 2012. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3444117/`, [Online; urldato 25.05.2017].

[43] H. G. de Marina, F. J. Pereda, J. M. Giron-Sierra, and F. Espinosa, "Uav attitude estimation using unscented kalman filter and triad," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 4465–4474, Nov 2012.

[44] Freescale Semiconductor, *Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference*, 11 2015. Rev. 4.0.

Appendiks

A Kalibrering av magnetometer

For å få fornuftige verdier fra magnetometeret må effektene av hard- og mykjern fjernes fra målingene. Optimalt vil målingene fra magnetometeret gi en sirkel med sentrum i origo, når x-, y- og z-komponentene av magnetfeltet plottes mot hverandre (se rød sirkel i for eksempel figur A.1a. Effekten av hardjern på målingene er et konstant additivt ledd som vil skyve sentrum av sirkelen vekk fra origo. Mykjernseffekten er noe mer kompleks, men i plottet vises effekten ved at plottet får en elliptisk form.

Første skritt er å lagre all sensordata man får ved å rotere sensoren i alle mulige retninger. Videre behandles dataene i MATLAB. Metoden for å kompensere for hardjernseffekter er rimelig rett fram. Man regner ut gjennomsnittet av minimal- og maksimalverdien for hver komponent

$$v_x = \frac{\min(m_x) + \max(m_x)}{2}, \tag{i}$$

$$v_y = \frac{\min(m_y) + \max(m_y)}{2}, \tag{ii}$$

$$v_z = \frac{\min(m_z) + \max(m_z)}{2}, \tag{iii}$$

og trekker denne fra magnetometermålingen.

En forenklet mykjernskorreksjon er utviklet fra [44] der forvrengningsmatrisa W er approksimert som er diagonal matrise med elementer gitt av (viii)

$$w_x = \frac{\min(m_x) - \max(m_x)}{2}, \tag{iv}$$

$$w_y = \frac{\min(m_y) - \max(m_y)}{2}, \tag{v}$$

$$w_z = \frac{\min(m_z) - \max(m_z)}{2}, \tag{vi}$$

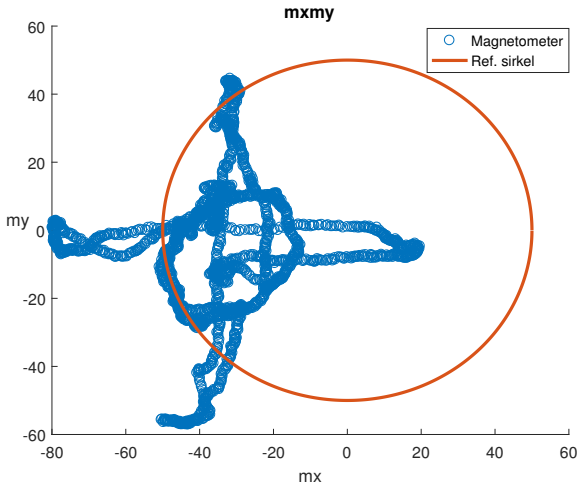
$$\bar{w} = \frac{w_x + w_y + w_z}{3}, \tag{vii}$$

$$W = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_3 \end{bmatrix} = \begin{bmatrix} \frac{\bar{w}}{w_x} & 0 & 0 \\ 0 & \frac{\bar{w}}{w_y} & 0 \\ 0 & 0 & \frac{\bar{w}}{w_z} \end{bmatrix}. \tag{viii}$$

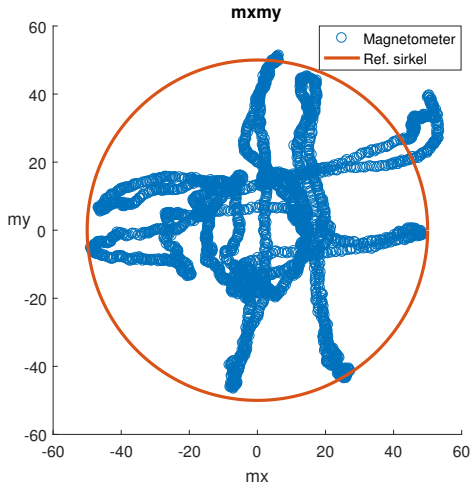
Målingene korrigert for hard- og mykjernseffekter blir da

$$m_x^{korr} = w_1(m_x - v_x), \tag{ix}$$
$$m_y^{korr} = w_2(m_y - v_y), \tag{x}$$
$$m_z^{korr} = w_3(m_z - v_z). \tag{xi}$$

Se figur A.1a-A.3b for plott av data før og etter kalibrering.

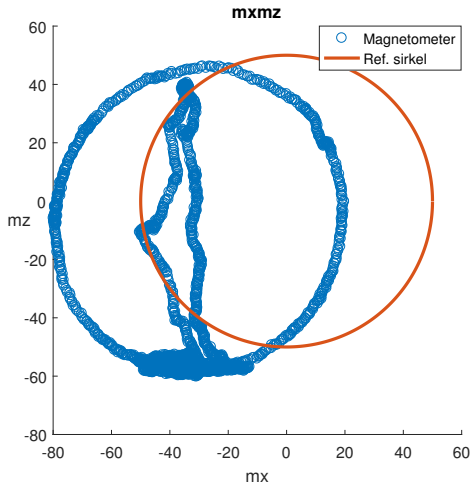


(a) Før kalibrering

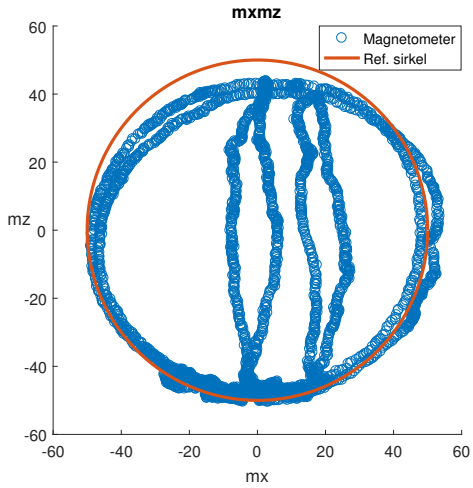


(b) Etter kalibrering

Figur A.1: x- og y-komponenter av kompassmåling før og etter kalibrering.

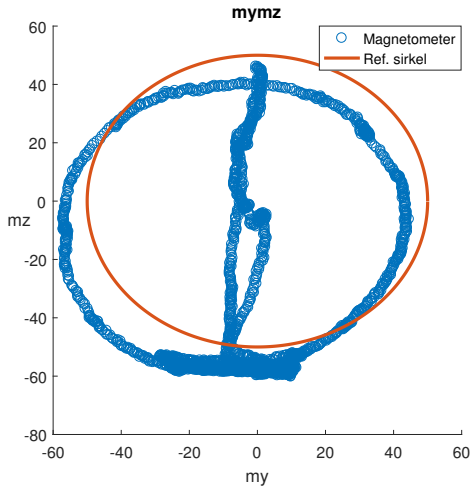


(a) Før kalibrering

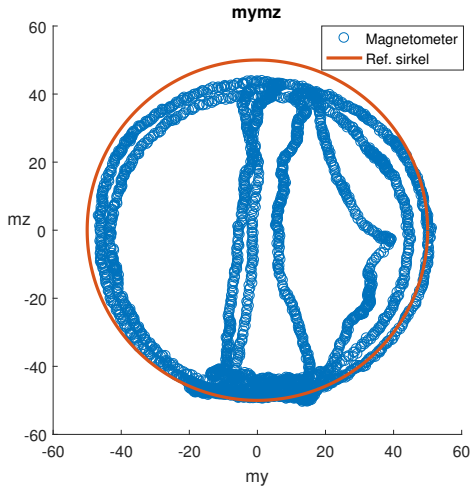


(b) Etter kalibrering

Figur A.2: x- og z-komponenter av kompassmåling før og etter kalibrering.



(a) Før kalibrering



(b) Etter kalibrering

Figur A.3: y- og z-komponenter av kompassmåling før og etter kalibrering.

B Diskretisering av komplementærfilter

Transferfunksjonen for et lavpassfilter:

$$H_{lp} = \frac{1}{\tau s + 1}. \tag{xii}$$

Transferfunksjonen for et høypassfilter:

$$H_{hp} = 1 - H_{lp} = \frac{\tau s}{\tau s + 1}. \tag{xiii}$$

Komplementærfilteret er en kombinasjon av lavpass- og høypassfilter, hvor signal 1 filtreres av det ene mens signal 2 filtreres av det andre. I.e.

$$Y_{kf} = H_{lp}Y_1 + H_{hp}Y_2, \tag{xiv}$$

$$Y_{kf} = \frac{1}{\tau s + 1}Y_1 + \frac{\tau s}{\tau s + 1}Y_2, \tag{xv}$$

$$Y_{kf}(\tau s + 1) = Y_1 + \tau s Y_2, \tag{xvi}$$

$$\Downarrow \tag{xvii}$$

$$\tau \dot{y}_{kf} + y_{kf} = y_1 + \tau \dot{y}_2. \tag{xviii}$$

Diskretisering av (xviii) kan gjøres ved hjelp av Eulers metode, det vil si

$$\dot{y} \approx \frac{y[n] - y[n-1]}{h}, \tag{xix}$$

hvor $y[n]$ er telling nummer n av y og h er tiden mellom hver telling. Dette fører til

$$\frac{y_{kf}[n] - y_{kf}[n-1]}{h}\tau + y_{kf}[n] = y_1[n] + \frac{y_2[n] - y_2[n-1]}{h}\tau, \tag{xx}$$

$$y_{kf}[n]\frac{\tau + h}{h} - \frac{\tau}{h}y_{kf}[n-1] = y_1[n] + \frac{\tau}{h}(y_2[n] - y_2[n-1]), \tag{xxi}$$

$$y_{kf}[n] - \frac{\tau + h}{\tau}y_{kf}[n-1] = \frac{h}{\tau + h}y_1[n] + \frac{\tau + h}{\tau}(y_2[n] - y_2[n-1]). \tag{xxii}$$

Ved å definere

$$\alpha = \frac{\tau}{\tau + h}, \tag{xxiii}$$

kan man forenkle (xxii) til

$$y_{kf}[n] = \alpha y_{kf}[n-1] + (1 - \alpha)y_1[n] + \alpha(y_2[n] - y_2[n-1]). \tag{xxiv}$$

C DVD-vedlegg

Vedlegget inneholder

- Rapport
- Resultater som ikke ble med i rapporten
- Koblingsskjema
- Arduino_Uno
  - Arduino Uno Rev3-skjema
  - ATMega328P Datablad
- GPS
  - Skjematisk tegning GPS
  - Adafruit GPS-bibliotek for Arduino
  - Datablad
- LSM9DS1_IMU
  - Datablad
  - Sparkfun LSM9DS1 Arduino-bibliotek
  - Skjematisk tegning LSM9DS1
  - Freescale Semiconductor application note: Tilt sensing
  - Freescale Semiconductor application note: Calibrating an eCompass
- Arduinokode
  - parachute_control.ino
  - eepromRead.ino (Lese ut testdata)
  - LSM9DS1_Basic_I2C.ino (Grensesnittet mot IMU-en er basert på denne koden)
  - parsing.ino (Grensesnittet mot GPS-en er basert på denne koden)
  - libraries (Alle relevante Arduino-bibliotek)
- Prototype
  - demo (video)
  - bilder
- MATLAB-filer