Sensur av hovedoppgaver
Høgskolen i Buskerud
Fakultet for Teknologi

Prosjektnummer: 2013-12
For studieåret: 2012/2013
Emnekode: SFHO-3200

Prosjektnavn: System for Tactical Aerial Reconnaissance

Utført i samarbeid med: Kongsberg Integrated Defence Systems.

Ekstern veileder: Per Wollebæk.

Sammendrag: S.T.A.R. har laget en flygende plattform som samler inn etterretningsdata, herunder bilder og lokasjonsdata.

Stikkord:
- UAS
- Sanntidssystem
- Geodata

Tilgjengelig: JA

Prosjektdeltakere og karakter:

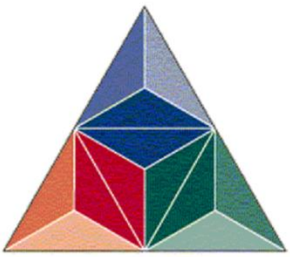
Navn	Karakter
Liam Jensrud	
Jørgen Markussen	
Espen Nilsen	
Carl Sandaker	
Gunnar Sandaker	

Dato: 12. Juni 2013

Karoline Moholth Intern Veileder	Hallstein Asheim Hansen Intern Sensor	Per Wollebæk Ekstern Sensor

1	Etteranalyse
2	Implementasjon
3	Utvidelser
4	Prosjektplan
5	Risikoanalyse
6	Systemdesign
7	Programvaredesign
8	Fysisk design
9	Kravspesifikasjon
10	Testspesifikasjon
11	Testrapport

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Etteranalyse
Dokumenttype:	Scrum dokument
Dokument ID:	SCRUM011
Versjonsnummer:	V1.0
Versjonsdato:	24.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	24.05.13	Første versjon

3 Introduksjon

Dette dokumentet vil fungere som en oppsummering av prosjektet i sin helhet. Vi tar her et tilbakeblikk på prosjektet på samme måte som vi tidligere har hatt «sprint review» for hver sprint.

4 Innledning

Fire personer fikk høsten 2012 en veldig åpen prosjektoppgave med stort potensial fra Kongsberg Integrated Defence Systems. Det var i utgangspunktet Carl Sandaker som hadde kontakt med bedriften og som videre satte sammen en gruppe med motiverte studenter. Snart knyttet også gruppa til seg et femte medlem slik at totalen ble tre datastudenter og to elektrostudenter.

Oppgaveformuleringen var på dette tidspunktet i sin spede begynnelse og den har blitt raffinert og utviklet mye siden den gang. Vi skulle i utgangspunktet, med et fly eller helikopter kjøpt inn av oppdragsgiver, utvikle funksjonalitet for å kunne følge og rekognosere for en kolonne med kjøretøy. Dette ville kunne gi militære bakkestyrker en taktisk fordel. Navnet STAR, System for Tactical Aerial Reconnaissance, stammer fra dette bruksområdet.

Etter en modningsprosess kom vi, i samråd med oppdragsgiver, frem til at vi selv skulle anskaffe nødvendig utstyr og utvikle på dette. På dette tidspunktet begynte det å bli klart at prosjektet også kunne dratt nytte av én eller flere maskinstudenter, noe som da var for sent å ta tak i.

Bruksområdet for det endelige produktet har gått gjennom flere iterasjoner. Intensjonen er nå at produktet skal kunne fungere som en lavkost erstatning for større, mer kostbare droner, slik at oppdragsgiver kan bruke «flyfoto» fra STAR i en demo- eller salgssituasjon.

5 Sammendrag

Da prosjektet startet hadde vi i prosjektgruppa ikke kunnskapsgrunnlaget for å forstå omfanget av oppgaven vi hadde foran oss. Vi visste ikke hvor stort dette kunne være, eller hvorvidt det i det hele tatt var mulig. Dette var noe av det som gjorde oppgaven så spennende; vi så at det var veldig stort potensiale. Vi hadde, og har fortsatt, mange spennende ideer for utvidelser og fordypningsområder. Selv om det har vært en fantastisk oppgave med store muligheter, ser vi i retrospekt at det kanskje var for stort og for ambisiøst. Siden hverken prosjektgruppa eller oppdragsgiver satt med kunnskaper innenfor RC-flygning var det ingen som satte på bremsene når ambisjonene ble for høye. Dette har ført til at vi ikke har nådd alle målene vi satte ut for å nå.

Vi synes på den andre siden at prosjektets konsept er så bra at vi håper våre erfaringer vil bygges videre på av fremtidige avgangsstudenter. Det er nemlig ikke bare vi som har jobbet med oppgaven som har latt oss rive med av interesse og spenning. Vi har trukket mye oppmerksomhet fra medstudenter på alle klassetrinn, ansatte ved høgskolen, venner, familie og ikke minst ildsjeler i RC-samfunnet verden over. Med STAR har vi faktisk, i løpet av et kort skoleår, bygget et lett gjenkjennelig varemerke, og vår visjon er at STAR skal bli et gjengående hovedprosjekt i mange år fremover.

6 Prosjektevaluering

I oppstartfasen merket vi spesielt nytten av å ha en intern veileder som er engasjert og brenner for prosjektstyring. Det at vi har hadde en utrolig dyktig veileder som kunne guide oss gjennom prosjektets oppstart og videre utover var utrolig nyttig og ikke minst lærerikt. Spesielt om man ser på hvor omfattende og uoversiktlig prosjektet var tidlig i løpet, var det nyttig med en veileder som pekte oss i riktig retning. Veiledningen gjorde så vi kom oss på beina tidlig i prosessen og kunne angripe utfordringer som møtte oss på en konstruktiv og fornuftig måte.

Oppgaven ble gradvis mer definert i løpet av prosjektets gang. Det at oppgaven har blitt så bra som den har blitt kan i stor grad tilskrives oppdragsgivers fleksibilitet og tilpasningsevne. Oppdragsgiver har siden dag én vært åpen for å diskutere målsetning og utforming av prosjektet, noe som har vært kritisk i et så dynamisk og uforutsigbart prosjekt. Selv langt ut i prosjektet var oppdragsgiver åpen for å diskutere krav til systemet. Det har på møter med oppdragsgiver vært en god tone, og som gruppe mener vi at vi har hatt et godt forhold til både oppdragsgiver og veileder/sensor. I etterpåklokskapens lys mener vi nok allikevel at terskelen for å be om hjelp og veiledning ville vært lavere om ekstern veileder og sensor ikke hadde vært én og samme person.

Veien fra prosjektet startet til det sluttet har vært lang, utfordrende, spennende og ikke minst utrolig lærerik. Det har vært i dette prosjektet vi har funnet frem alt vi har lært gjennom 3 års skolegang og jobbet sammen mot et felles mål. Det har gitt oss en forsmak på hvordan det er å jobbe som ingeniør i et prosjekt og å sette kunnskapene ut i livet.

Mer enn noe annet, har det vært et tidssluk å drive med RC-kjøretøy. Det har vært en bratt læringskurve, men det har også vært fantastisk morsomt. I utgangspunktet skulle dette være nesten en ren programvareoppgave. Slik ble det ikke, faktisk har egenprodusert programvare heller blitt en av prosjektets minste komponenter. Det har gått utrolig mye tid til bygging og tuning av RC-plattformen samt det å forske på muligheter innen hardware. Prosjektet var såpass stort at vi kun fikk mulighet til så vidt å være innom hver enkelt del.

Den delen av prosjektet som omhandler hardware og alle komponentene vi har brukt ble mye større enn vi tidligere antok. Beslutninger som ble tatt rundt bestillinger måtte tas hurtig men allikevel velbegrunnet. Bestillinger måtte bli gjort på et tidlig tidspunkt for at prosjektet ikke skulle stagnere. I de aller fleste tilfeller har vi endt opp med komponenter som gjør jobben på en god måte. Med det antallet og omfanget vi hadde å forholde oss til ble vi nødt til å gjøre noen antakelser underveis. Man kunne kanskje tatt selvkritikk for noen av valgene som ble tatt, men vi føler ikke at dette er på sin plass med tanke på; mengden forskning gjort på forhånd, budsjett og tidspress. I retrospekt ser vi også at valgene vi tok stor sett var riktige. Uansett var det viktigere å få alt i tide enn å forske for lenge.

Vi føler at vi har lagt en solid grunnmur for videre arbeid på denne plattformen. Det gleder oss veldig å se kunne se dette prosjektet bli ført videre. Valgene vi har tatt har gitt dette prosjektet uendelig mange muligheter og veier å gå fra der vi leverer det fra oss. Alle enheter og programvare er åpent, «open source» / hardware, og har et sterkt nettsamfunn i ryggen.

Videre er det bare fantasien som setter grenser.

6.1 Prosjektmodell

Prosjektet har vært utført med Scrum som prosjektmodell. Dette var ikke den første prosjektmodellen gruppen hadde i tankene. I utgangspunktet valgte vi RUP, men etter noe mer omfattende forskning og dokumentasjon av flere modeller falt til slutt valget på Scrum.

Gruppen opplevde at modellen og verktøyene som underbygger bruken av Scrum ga oss en arbeidsflyt og organisatorisk struktur som vi trivdes i. Det var likevel utfordrende at prosjektet inneholder mange administrative oppgaver som kan være vanskeligere å håndtere i Scrum enn i en prosjektmodell med et sterkere hierarki og ansvarsområder.

Scrum har til tider hvert utfordrende og vi har erfart at man må være godt sammensveiset, ha et godt miljø og gruppemedlemmer som er selvgående og selvstendige. Hvis disse kriteriene blir møtt vil Scrum gi en veldig god gruppedynamikk som er veldig tilpasningsdyktig og har stort potensiale.

Når vi kom i gang fokuserte vi på å kjøre så ren Scrum som mulig. Dette innebar da også at tidligere fordelte prosjektroller og ansvarsområder ble tilpasset Scrum. Selv om modellen som sagt har vært utfordrende til tider, har den vært veldig spennende å jobbe etter. Dag til dag jobbing kan kanskje være lettere å forholde seg til hvis man har en veldig god plan, for eksempel et Ganttdiagram med tidsfrister og perioder. Ved bruk av Daily Scrum-møter (DS) og korte, godt planlagte sprinter, vil også fokuset bli lagt der det burde ligge og knytte alt sammen, samt gi god oversikt på en dag til dag basis.

Scrum er en veldig smidig modell, dette er noe vi merket spesielt godt i forhold til krisehåndtering. Krisesituasjoner som inntraff fikk derfor ikke de største konsekvensene for progresjonen i prosjektet. Som sagt tidligere, har Scrum vist seg å være veldig tilpasningsdyktig når det er nødvendig og har fungert veldig godt for gruppa. Modellen passer absolutt best til rene programvareprosjekter og kan være litt vrien å integrere med andre deler av et prosjekt, derfor føler vi at vi ikke fikk utnyttet prosjektmodellen og administrasjonsverktøyet fullt ut.

Alle sprinter i løpet av prosjektet har resultert i iterasjonsdokumenter som har blitt levert til ekstern og intern veileder. Ett dokument som omhandler planlegging for sprinten og ett som tar for seg avslutningen. I hvert planleggingsdokument er det en figur som viser til et estimat over hvor vi kommer til å ende opp ved leveranse. Dette er en figur over alle kravene fra oppdragsgiver, hvor hvert enkelt krav har fått ett tidsestimat og hele figuren har en rød strek som har beveget seg opp og ned gjennom hele prosjektet. Når vi startet med dette var ikke estimatene veldig presise. Etter vi begynte å avlede hovedkravene fra oppdragsgiver til mindre aktiviteter ble disse estimatene ikke bare mer realistiske, men også arbeidet mot på en mer direkte måte. Vi føler dette har gitt veiledere og sensorer en viss innsikt i hvor det er vi kommer til å ende opp til slutt.

6.2 Prosjektarbeidet

Hovedoppgaven har gitt oss som studenter muligheten til å praktisere mye av det vi har lært gjennom bachelorutdanningen. Det å kunne se at teorien en har lært seg faktisk virker og man kan bruke den til noe nyttig er en veldig viktig milepæl i en hver utdanning. Samtidig føler vi at vi har lært mye av å jobbe med selve prosjekt.

Som en gruppe er alle enige i at vi har fungert veldig godt sammen. Konstruktiv kritikk og tilbakemeldinger har alltid blitt godt mottatt, og det har ikke blitt noen sure miner av den grunn. Gruppemedlemmer har fått ytret seg fritt og alt har foregått på et profesjonelt nivå.

Det at vi ikke har hatt noen «teknisk» veileder har til tider vært en utfordring. Denne utfordringen har vi løst ved å være aktive på diverse forum på Internett, som *Google Groups*, *GitHub* og *SourceForge*. Vi har også hatt kontakt med utviklere i andre land via epost. Det har igjen åpnet en ny verden for oss der vi ser at den som spør faktisk får svar. I «open source»-verden sitter det ildsjeler som engasjerer seg i andres utfordringer og som bidrar med sin kunnskap. Det å stå så fritt på egne bein har gjort at vi har fått gjøre oss våre egne erfaringer. Prosjektet har nok ikke kommet så langt som det kunne ha kommet med et sterkt nettverk av ressurspersoner, men kanskje har vi som studenter lært mer av det.

Som tidligere nevnt mener vi det kunne vært en bedre ordning å ha ekstern sensor og veileder som to personer. Dette er noe både Høgskolen og oppdragsgiver kan vurdere å ta til etterretning for senere hovedprosjekt. Vi har fra første stund fokusert på å ha et profesjonelt forhold til oppdragsgiver og sensor. Samtidig som dette er veldig fornuftig, har det kanskje hindret oss i å benytte oss fullt ut av personens veilederrolle. Rollen som sensor og veileder blir gjerne mer sensor enn veileder. Vi mener at forholdet man får til en sensor og en veileder er veldig forskjellige. Det er vanskelig for en person å inneha begge rollene samtidig.

Vi mener også at skolen kunne forberedt ingeniørstudentene bedre på enkelte aspekter ved prosjektarbeid. Spesielt når det gjelder kjennskap til prosjektmodeller og prosjektstyring, dokumentering av kode (*Javadoc*, *Doxygen*) og versjonskontroll (*GitHub*, *Subversion*). Høgskolen kunne også bedre informert studentene om de laboratorier og det utstyr som er tilgjengelig. Vi mener også det kunne vært veldig nyttig for prosjektgruppene om vi hadde disponert et felles tekjøkken/pauserom som ville være tilgjengelig utenfor kantinens åpningstider. Det burde også vært tilrettelagt for at skolen kan ha spesielle åpningstider i spesielt intensive perioder (les: at studentene ikke må ut klokka 23). Disse tiltakene kunne gjort prosjektarbeidet enda bedre enn det allerede er.

6.3 Produktet

Vi har sammenliknet de krav, ønsker og oppfatninger vi hadde i starten av prosjektet med hva vi har utviklet. Det er mye vi ikke har oppnådd, samtidig som vi er meget fornøyde med det endelige resultatet. Når man løfter blikket opp fra detaljene i hvert krav og ser på helheten i systemet, så føler vi at vi har satt opp et robust system som vil fungere meget godt for videre utvikling. Den hardware som vi har anskaffet vil ikke virke begrensende på systemets muligheter. Det faktum at programvaren til autopiloten er i stadig utvikling av et nettsamfunn gjør at ytterligere funksjonalitet vil tilføres systemet over tid.

6.3.1 Bruker- og utviklerveiledning

For å gjøre systemets dokumentasjon så bra og brukervennlig som mulig, har vi opprettet en wiki-side som inneholder brukermanualen og utviklermanualen. På den måten kan vi guide brukere og andre utviklere ved hjelp av tekst, bilder, video og direkte linker til eksterne ressurser.

Dette er en løsning som er godkjent av ekstern veileder/sensor og vi er meget fornøyd med resultatet. Her vil både brukere av systemet og de som skal videreutvikle det enkelt kunne finne frem til den informasjonen de leter etter.

6.4 Presentasjoner

I prosjektsammenheng er presentasjoner det eneste stedet hvor vi kan påvirke inntrykket utenforstående personer har om prosjektet vårt. Dette gjør de tre presentasjonene vi holder enormt viktige for vårt bilde utad.

Vi føler at vi har klart å holde underholdende presentasjoner med god flyt og relevant innhold. Dette har vært et veldig stort fokus for oss. Det har krevd mye arbeid, men gjort oss alle vanvittig mye bedre på å holde presentasjoner og langt mer komfortable foran større tilskuermengder. Vi er godt fornøyd med det grunnlaget vi har fått gjennom presentasjoner i andre fag. Utviklingen for hovedprosjektet kontra det vi har gjort tidligere har vært at det nå har vært et langt større fokus på å selge, både oss selv og et produkt. Presentasjoner i andre fag har vært mer tørre og fagtunge.

Presentasjonsverktøyet Prezi var også noe vi oppdaget i sammenheng med presentasjoner holdt i andre fag på skolen. Alt i alt har vi fått utrolig mye ut av presentasjonene holdt i sammenheng med hovedprosjektet og lært mye nytt om oss selv.

6.5 Risikohåndtering

I alle prosjekter finnes det en del risikoer, og som oftest vil noen av disse inntreffe i løpet av prosjektets levetid. Det å ha oversikt over disse, samt en plan for hvordan de bør håndteres dersom de inntreffer er veldig viktig.

Det er flere risikoer som har inntruffet i løpet av prosjektets levetid. Disse har blitt håndtert på en god måte og vært omtalt som en stor risiko i risikodokumentet. Den første risikoen som inntraff var at et gruppemedlem ble sykemeldt. Videre hadde vi en avhengighet mot en tredjepart som gikk på «open source» programvare, hvor vi var avhengig av at mye ferdig funksjonalitet skulle være på plass. Funksjonaliteten har ikke kommet på plass. Dette ble håndtert ved å fokusere på å bygge opp rammeverket rundt systemet, slik at når tiden kommer da denne funksjonaliteten implementeres vil vårt system være godt forberedt på å bli integrert i systemet.

Siste store hendelse var at hexakopteret gikk i bakken og knakk en arm, senterplatene og noen propeller. Dette var en hendelse vi ikke bare hadde forutsett, men som vi forventet. Derfor hadde vi sørget for å ha nok reservedeler til å håndtere situasjonen. Løsningen var å bygge om plattformen fra en hexa-rotor til en quad-rotor, altså med 4 armer.

6.6 Ansvarsområder

På de fem gruppemedlemmene i gruppen, har vi blant oss dekket følgende fagfelt:

- Data, virtual systems.
- Data, embedded systems.
- Elektro, kybernetikk.
- Elektro, audioteknologi.

Uavhengig av linje, har alle i gruppen måtte dykke dypt ned i stoff utenfor deres spesifikke fagfelt og har gjort seg flere nyttige erfaringer med nye systemer og fagområder underveis. Alle har måttet tørre å hoppe i det med begge beina først, og utviklingen har faktisk ligget på et langt høyere nivå enn vi i utgangspunktet forventet.

Siden Scrum baserer seg på en relativt flat administrativ struktur og ingen egentlig sitter noe høyere enn andre, så har ikke gruppa hatt noen prosjektleder eller noe som helst form for hierarki. Scrum sier likevel at man har forskjellige roller i et *Scrum team*, tre stykk for å være eksakt. Disse er:

- Scrum master.
- Product owner.
- Utviklere / Team-medlemmer.

Innad i gruppen har vi alle vært utviklere og medlemmer av teamet, men to av oss har hatt andre roller i tillegg. Scrum master rollen har gått på å holde Scrum relaterte møter og forfatte tilhørende dokumenter, samt passe på at alle medlemmer har nok å henge fingrene i. *Product owner* har hatt ansvar over *sprint backlog* og sørget for at oppdragsgivers interesser har blitt ivaretatt i planlegging og utviklingsfasen.

Når vi delte ut disse rollene kunne vi ikke like mye om Scrum eller kjente hverandre like godt som vi gjør ved prosjektets slutt. Vi visste dermed ikke hvem som passet best eller var mest skikket til å inneha diverse roller. I retrospekt skulle kanskje disse rollene vært delt ut annerledes. Ikke nødvendigvis fordi de ble utført dårlig, men fordi vi nå har et bedre inntrykk av hvilke personligheter som passer hvilken rolle best.

7 Prosjektoversikt

7.1 Økonomi

Prosjektet hadde i utgangspunktet et budsjett på NOK 10.000,-. Etter forstudien var gjennomført fremmet vi tre prisoverslag for oppdragsgiver. Beslutningen ble da tatt om å utvide budsjettet til NOK 15.000,-. Siste revurdering av budsjettet ble tatt i april måned. Vi informerte oppdragsgiver om at en del utstyr som brukes er på utlån fra gruppemedlemmene. Oppdragsgiver ytret da et ønske om å overskride budsjettet for å kjøpe inn disse komponentene.

Budsjettet og de fallgruvene vi har gått i er nok en refleksjon av hvor lite vi visste om RC-bygging da vi startet prosjektet. Det viste seg å være svært mange kostnader som vi i utgangspunktet ikke forutså. Måten vi så håndterte dette på var at vi gjorde det vi kunne for å holde kostnadene nede. Det betydde å bruke eget utstyr der vi hadde det, kjøpe de billigste komponentene som var gode nok, og ikke minst å produsere selv det vi hadde kompetanse og utstyr til.

I etterkant kan det diskuteres om det hadde vært en bedre løsning å gå tilbake til oppdragsgiver og be om mer penger, heller enn å forsøke å spare penger der det var mulig. Enkelte steder ville vi kunnet kutte utviklingstiden betydelig om vi hadde kjøpt dyrere komponenter, som for eksempel CMOS kamera som viste seg å ha dårlig driverstøtte.

Noe som har reddet budsjettet er at noen av gruppemedlemmene har hatt veldig mye utstyr liggende, og har villig gitt forbruksmateriell og lånt ut verktøy til prosjektet. Hadde dette ikke vært tilfelle ville ikke prosjektet vært mulig å gjennomføre.

Regnskap er grundig bokført og overleveres oppdragsgiver i en egen perm.

7.2 Timer

Timer brukt på prosjektet har blitt ført på to forskjellige steder, i OnTime prosjektstyringsverktøy og i et Google Docs regneark.

7.2.1 OnTime

I administreringsverktøyet OnTime, har vi ført effektivt arbeid på spesifikke og planlagte aktiviteter. Administrativt arbeid som møter og liknende blir sett bort ifra. Tidligere i prosjektet førte vi også administrativt arbeid i OnTime, men vi besluttet at verktøyet og prosjektmodellen ikke egnet seg for denne typen kontroll.

7.2.2 Google Docs

Reelle timer brukt på prosjektet har blitt ført på regneark i skytjenesten Google Docs. Her blir alt av timebruk ført sammen med en forklaring på hva timene ble brukt på og hvilken dag det gjaldt.

8 Avsluttende tanker

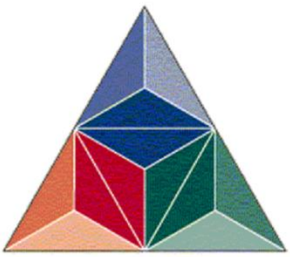
Vi har alle hatt et fantastisk skoleår med et spennende prosjekt! Resultatet er faktisk at vi fler av oss ser for seg å fortsette med RC som en hobby, og vi ønsker å fortsette å forske og bidra til de «open source» prosjektene vi nå har engasjert oss i.

I løpet av året har vi utviklet oss både faglig og personlig. Vi har fått uvurderlig erfaring innen samarbeid og presentasjonsteknikk. Sist, men ikke minst, har vi tatt et gigantisk steg fremover i utdanningen, og følger oss faktisk klare for å gå ut i arbeidslivet som ingeniører!

Til slutt vil vi takke følgende personer for deres bidrag til prosjektet. Uten disse menneskene ville prosjektet ikke kommet dit det er i dag:

- Karoline Moholth
- Arne Bjørnar Næss
- Per Wollebæk
- Åge Skaug
- Rolf Longva
- Barbro Guldbransen
- Anton Babushkin, utvikler på PX4
- Adam Sylvester, NITRO
- Lorenz Meier, initiativtaker og utvikler på PX4
- Robert C. Nelson, spesialist på BeagleBoard
- E.K. Nilsen Elekronikk Service & IT-Systemer, sponsing av materiell og diverse
- Traconet, sponsing av materiell

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Implementasjon
Dokumenttype:	Systemdokument
Dokument ID:	S009
Versjonsnummer:	V1.0
Versjonsdato:	24.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Figurliste

3 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	24.05.13	Første versjon

4 Introduksjon

Dette dokumentet gir en innføring i hvilken funksjonalitet som finnes i systemet og hvilke muligheter for utvidelser som er mulig. Det er utformet en online wiki[1] som inneholder utviklerguiden og brukermanualen. Ved å benytte en wiki er det lettere å oppdatere dokumentasjonen og kontrollere historikken. For tanker og planer om hvordan systemet kan videreutvikles og forbedres, se dokumentet *S010 – Utvidelser.*

Utviklerguiden gir en grundig innføring helt fra det å sette opp utviklingsmiljøer for de ulike programvareelementene og helt til hvordan å fortsette utviklingen av ny funksjonalitet utover hva som står beskrevet implementert i dette dokumentet.

Brukerguiden gir en god oversikt over hvordan systemet skal tas i bruk, hvilke hensyn man bør ta for flygning, billedtakning og oppdragsplanlegging.

Dokumentet forteller i hovedsak virkemåten til de implementerte løsningene, men går ikke ned i dybden på hvordan løsningene er blitt implementert i kode og lignende.

Figur 1 – Hexakopteret og Quadkopteret

5 Komponenter

5.1 BeagleBoard-xM

Hovedfunksjonen BeagleBoard har i vårt system er bildebehandling. BeagleBoard har tilkoblet kamera og har to-veis kommunikasjon til autopiloten.

5.1.1 Operativsystem

Originalt ble BeagleBoard levert med Linux distrubisjonen; *Ångström.* Dette er en distribusjon med et meget smart konsept, nemlig å skreddersy et filsystem for en embedded enhet. Vi har vært igjennom flere varianter av *Ångström* som vi har generert, men vi har imidlertid kommet fram til at *Ubuntu* ble et bedre valg. For å nevne noen av fordelene så var tilgangen på programvare og støtten fra nettsamfunn mye større.

Utfordringene rundt operativsystem til BeagleBoard har vært mange og lange. Dette skyldes i stor grad at vi var avhengige av å kompilere inn drivere for kameramodulen fra Leopard Imaging. I utgangspunktet kommer BeagleBoard med Ångström distribusjon. Denne var uaktuell å bruke da den ikke støttet kameramodulen. Vi satte så i gang forskning for å finne en fungerende løsning. Aptina, som leverer selve kamerasensoren til kameramodulen, har laget en driver for modulen, men denne er til en eldre kernel versjon, 2.6.X. Vi krysskompilerte en versjon av Ångström der vi brukte *menuconfig* verktøyet for å sette opp parametre for å kompilere inn, blant annet, driveren fra Aptina.

Videre har vi vært gjennom mange ulike oppsett før vi endte på Ubuntu 12.10:

- Ångström med kernel 2.6.32, både versjon med og uten grafisk brukergrensesnitt
- Ångström med kernel 3.6.X, 3.7.X
- Texas Instruments eget operativsystem for BB
- Android 2.1
- Ubuntu 12.10 console image

Med Ubuntu Linux fungerte det meste *out of the box*, som USB, FTDI og WiFi. I tillegg er Ubuntu i daglig bruk for flere av utviklerne på gruppa.

5.1.2 Billedtagningsmodul

Vi planla å benytte en bildemodul fra *Aptina Imaging Corporation* levert av *Leopard Imaging Inc*. BeagleBoard har en egen dedikert kontakt for kameramoduler fra. Modellen vi har leverer ut *raw-RGB-Bayer* som videre behandles i CPUens dedikerte ISP (image singal proccessor) som er en del av DSPen (TMS320C6000). Støtte for denne enheten kan se ut til å være veldig begrenset i versjon 3.x av Linux.

5.1.3 GEO tagging

BeagleBoard er også kraftig nok til å kunne skrive NSIF/NITF filer fra bildene som blir tatt. NSIF filer er en spesiell filtype som *wrapper* diverse bilder med standardiserte *headere*. For dette benytter vi biblioteket *NITRO* som er *OpenSource* og tilgjengelig fra sf.net. Det finnes flere biblioteker for dette som vi også har tatt en liten titt på som *GDAL*, *openmap* og *osgeo*.

NSIF1.0 og NITF2.1 er identiske formater. Biblioteket vi har jobbet med å bruke for å opprette NITF (.ntf) filer, NITRO, er svært dårlig dokumentert. Tilsynelatende er NITRO allikevel et svært kraftig

bibliotek som kan benyttes til det aller meste av det vi ønsker å gjøre. Det inneholder støtte for BLOCKA, en utvidelse som oppdragsgiver ønsker at benyttes for nøyaktig merking av koordinater i bilder.

NITRO støtter å dekomprimere JPEG (C3) bilder, men støtter ikke å komprimere til C3. Derfor må de resulterende NITF filene inneholde ukomprimerte bilder. Dette vil igjen føre til at et oppdrag med medfølgende bilder vil ta stor plass. Om en kunne skrive en C3 komprimering ville dette kunne bedres. For dekomprimering av C3 benytter NITRO jpeglib.h. Det kan være en utfordring å få alle dette til å fungere, og bruk av andre JPEG biblioteker kan vurderes.

I utgangspunktet er NITRO skrevet i C, men mye av funksjonaliteten støttes også ved hjelp av Python bindinger. Disse er ikke altomfattende, og om Python skal benyttes til dekomprimering av JPEG må bindinger for dette eventuelt også skrives.

Informasjon om NITF Image Subheader kan finnes i STANAG-4545 på side C-1-13. Informasjon om BLOCKA finnes i Appendix E – ASDE, side E-41.

5.1 Kamera

Fra et tidlig stadium der vi var på jakt etter hardware til prosjektet, valgte vi å gå for et CMOS kamera på grunn av vekt og pris. Mulighetene for *interfacing* med disse modulene er ofte mye bedre enn ferdigproduserte kameraer i den nedre prisklassen. Valget falt på kameramodulen fra Leopard Imaging ettersom BeagleBoard allerede var vurdert som en *embedded* maskin, hadde en kontakt som passet, samt riktig DSP.

Kamera ble også vurdert på grunnlag av «proof of consept» strategien og mangler derfor blant annet autofokus og andre fordyrende egenskaper.

Senere dukket det opp store utfordringer rundt støtte for denne modulen. Produsenten lover at modulen er i stand til å klare 720p@ 25fps. Ingen i BeagleBoard-samfunnet har til dags dato klart å få bilder med rette farger eller høyere enn 6 fps fra denne modulen.

I sluttfasen av prosjektet hadde vi et krasj med modellen vår som førte til at vi ble nødt til å bytte ramme. Kamerastativet vi hadde bygget for kamera gikk også i stykker i krasjen, og kamerakontakten på BeagleBoard ble bøyd. Ombyggingen til quadkopter medførte nye utfordringer når det gjaldt kontaktens tilgjengelighet. Vi valgte derfor å benytte oss av et USB-webkamera isteden.

Det nye kameraet er i stand til å ta 8mpx bilder og video opp til 1080p, samt at det er mer robust. Dette kameraet er ikke spesielt godt egnet til å ta gode avstandsbilder, men gir mer en god nok kvalitet for å illustrere funksjonaliteten til systemet.

5.1 PX4

PX4FMU og PX4IO er de to hardwarekomponentene som utgjør autopiloten. Programvaren som kjører på disse utvikles av et nettsamfunn og er tilgjengelig på GitHub[2]. Det foregår en kontinuerlig utvikling og forbedring av koden med oppdateringer daglig. PX4 har sitt utspring fra Pixhawk[3] prosjektet ved ETH Zürich[4].

Autopiloten har to hovedfunksjoner; Stabilisere den flygende plattformen og tilby mulighet for autonom flygning. Stabiliseringen er fullstendig implementert og bruker PID[5]-tuning for å holde seg

stabil i luften. Som en sikkerhetsmekanisme er det ikke lov å ha mer enn en viss prosent forskjell i kraft på motstående rotorer. Det er fordi den flygende plattformen ikke skal falle ned å krasje ved at enkelte motorer stopper opp i arbeidet med å kompensere for vær og vind. Man mister dermed en del manøvrerbarhet for presisjonsflygning. Dette er derimot akseptabelt fordi dette systemet ikke skal drive «stuntflygning», men rolig overvåkningsflygning.

I skrivende stund tilbyr PX4 en autopilot som stabiliserer Tait-Bryan[6] rotasjoner. Den har derimot ikke stabilisering av posisjon. Det fulle omfanget av funksjonalitet er for stort og komplisert til å utdype på få linjer. PX4 er nærmere beskrevet på vår wiki[1].

6 Programvareimplementasjon

De ferdige løsningene og implementasjonene i dette avsnittet reflekterer ikke nødvendigvis mengden løsninger som har vært prøvd og heller ikke forskningen som har måtte vært gjort på forhånd. I tillegg til å ha gått gjennom mangfoldige iterasjoner, har enkelte applikasjoner også vært gjennom flere vurderte programmeringsspråk. Riktig støtte og rammeverk har også naturligvis vært veldig viktig for programmet man skal utvikle og mye tid har gått med for å finne ut av hva som kunne møte våre krav. Samtidig må man tenke på at dette skal kjøres på en embedded maskin uten altfor mye prosesseringskraft eller ferdig støtte for alt av rammeverk.

Testing er heller ikke et lettvint tema når man utvikler på en plattform noe som skal kjøre på en helt annen plattform. F. Eks på PX4 autopilot måtte alt av egenprodusert kode integreres med det som allerede fantes av *firmware*, for så å krysskompileres til en ARM arkitektur og *flashes* inn på autopiloten. Med dette kan man se at testprosessen tok lengre tid enn det ellers ville tatt om koden ble testet på samme plattform som den ble utviklet til å kjøre på.

6.1 Utviklingsmiljø

Det å krysskompilere applikasjoner til å kjøre i et annet miljø byr på en del utfordringer om man ikke har god kjennskap til dette. Både PX4 og BeagleBoard har arkitektur forskjellig fra en vanlig pc. Det er dermed nødvendig å sette opp et spesialisert utviklingsmiljø, og utvikleren står mindre fritt til å velge IDE.

6.2 Versjonskontroll

For at flere personer skal kunne jobbe med samme kode i de samme prosjektene, er det absolutt nødvendig å ha en form for versjonskontroll. Dette er også veldig nyttig med tanke på backup. Gruppen har benyttet GitHub[7] til all kode som vi har produsert. Bruk av GitHub er ikke trivielt, og er heller ikke et tema som per nå er dekket av utdanningen ved HiBu. Derfor valgte vi å utnevne en GitHub-ekspert i gruppa. Denne personen satte seg grundig inn i GitHub slik at vi kunne holde en *workshop* for gruppa og alle andre som måtte ønske å delta.

Installasjon av Git er ulikt på ulike operativsystemer. I Linux er Git en naturlig del av terminalen og veldig enkelt å installere. I Windows er det mer omfattende, og man har også ulike grafiske alternativer, og programmer som integrerer med Windows Explorer. Vi har brukt Git i både Linux og Windows og har hatt god erfaring med dette.

6.3 Python program for billedtagning og kommunikasjon

På BeagleBoard kjører det et Python program som vi har utviklet. Programmet tar seg av klargjøring for billedtagning. Dette er en prosess som består av flere steg. I kontrollen sjekkes det om USB-flashminne er tilkoblet og tilgjengelig og prøver eventuelt å montere filsystemet om nødvendig. Dersom programmet ikke finner noe USB-flashminne å skrive til venter programmet på at dette blir tilgjengelig. Programmet setter opp *path* til folderen som bilder og video skal lagres til. Programmet sjekker om det finnes en spesiell mappe i filstrukturen som skal hete *misson* dersom denne banen ikke er tilgjenglig opprettes banen. Dersom den derimot eksisterer scannes folderen for subfoldere og finner den med høyest nummer, legger til 1 og benytter dette som folderen den lagrer filer for dette oppdraget i. Programmet har en autonummer-teller som leses inn fra en fil, slik at programmet fortsetter nummerserien i forhold til hvor mange bilder som totalt er tatt, uavhengig av nåværende oppdrag.

For hver fil lagres det også et tekstdokument med samme tittel, dette dokumentet inneholder opplysninger nødvendig for bildetagging. Disse opplysningene kommer fra autopiloten og er som følger:

- GPS-tid
- Timestamp for posisjonsmåling
- Timestamp for målt attitude
- GPS-feilmargin
- GPS latitude
- GPS longitude
- GPS altitude
- Attitude roll
- Attitude pitch
- Attitude yaw

Det går også an å spørre etter enkeltopplysninger.

Videre i oppstartssjekken skanner programmet etter FTDI-porter (/dev/ttyUSBx) i området fra 0-3. Dersom ingen devicer er tilgjenglig så fortsetterprogrammet videre med kun http-støtte.

6.3.1 Kommandoer

BeagleBoard lytter etter kommandoer fra autopiloten og kan etterspørre diverse informasjon fra autopiloten. Vi har skrevet et program som kjører på autopiloten, ved hjelp av dette programmet er BeagleBoard i stand til å etterspørre geo- og flydata fra PX4. Dette gjøres via seriekommunikasjon på en FTDI-kabel. Kommandoene som kan sendes fra autopiloten er kommandoer som **ta bilde, start video, start bildeserie, stop video/serie, bilde *burst*.** Disse kommandoene sendes fra programmet (bb_handler) på autopiloten i henhold til oppdraget som er programmert på bakkestasjonen. Det er dermed lagt til rette for dette i vår modul på PX4 og i QGC (bakkestasjonen), men ettersom PX4 enda ikke er i stand til å fly en multirotor etter *waypoints* vil denne funksjonaliteten først være tilgjengelig når PX4-sammfunnet har utviklet dette. Det er også mulig å benytte fjernkontrollen for å sende kommando om å ta enkeltbilder.

6.3.2 http-grensesnitt

BeagleBoard er utstyrt med en wifi-link slik at vi kan overføre data til og fra BeagleBoard. Vi har også laget et webgrensesnitt for billedtagningen. Dette er skrevet i php og kjører i apache2. Web-grensesnittet fungerer slik at det benytter «cURL» for å sende kommandoer over til Python deamonen som kontroller billedtagningen. Python programmet benytter ***werkzeug*** og ***flask*** for å kjøre en liten http server på port 5000. All kommunikasjon med deamon går via GET-forespørsler. Webgrensesnittet lager også noen forhåndsvisninger (nedskalerte varianter) av bildene som blir tatt ved hjelp av GD-biblioteket (ett av flere bildebehandlingsbibilioteker) til PHP. Selve webinterfacet er å betrakte som en «bonus» feature ettersom det på en måte er en midlertidig løsning inntil full autonom støtte er implementert i autopiloten. Fra webinterfacet kan man **Ta bilde, start video, start bildeserie, stop video/serie, bilde *burst*.**

6.1 BB_handler – en egen px4 app

For at autopiloten skal være i stand til å gjøre det vi ønsker, altså å ta bilder basert på et forhåndsbestemt oppdrag, fant vi det nødvendig å skrive et eget program til PX4. I dette miljøet omtaler vi vårt program som en «app».

PX4 kjører NuttX, et sanntidsoperativsystem som er POSIX (Portable Operating System Interface) kompatibelt. Dette medfører at vanlige systemkall som man finner i blant annet UNIX inspirerte systemer er tilgjengelig i denne plattformen. PX4 benytter ORB (Object Request Broker) internt for *inter-process* kommunikasjon.

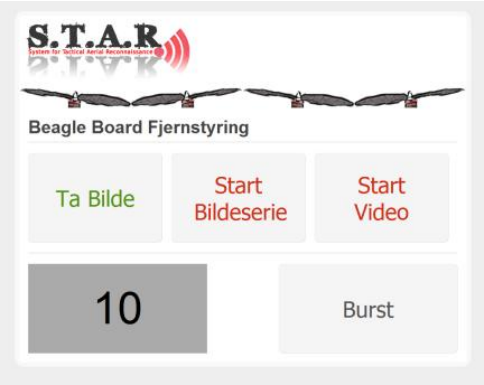
Vårt program lytter på flere *topics* som er publisert på ORB og det har blitt definert en egen *topic* som man sender data på. Dataene som sendes er en sammenstilling av informasjon om GSP, posisjon, orientering, klokker og annen nyttig informasjon relatert til jobben med merke bilder med geografiske data.

Denne meldingen sendes til bakkestasjonen via en egenutviklet utvidelse til MAVLink. Vi har nemlig definert vår egen MAVLink melding, men det er ikke implementert noe håndtering av denne meldingen i bakkestasjonen. Denne meldingen har ID 180 (se MAVLink definisjon for standardmeldinger, meldinger mellom 150 og 248er åpne). Visjonen er at bakkestasjonen skal kunne bruke disse dataene til noe fornuftig som for eksempel å markere i kartet de områder som ble fotografert.

BB_handler trenger en UART (Universal Asynchronous Receiver/Transmitter) for å kunne kommunisere med BeagleBoard og tar *device*-navnet som et kommandolinjedirektiv. Programmet starter en *deamon thread* som «kontinuerlig» lytter etter meldinger fra BeagleBoard og kommandoer som kommer fra *Mission Plan*. Det er også mulig å benytte fjernkontrollen via **aux3** signalet (tilordnet via QGC).

6.2 Web-grensesnitt

Webgrensesnittet er alle rede nevnt under http-grensesnitt. Det kommuniserer direkte med BeagleBoard over et trådløst nettverk. På dette grensesnittet ligger det beskrivende knapper med kommando for å ta bilde, ta bildeserie og video. De to siste må også stoppes igjen etter start, hvis ikke så fortsetter kamera å ta bilde til BeagleBoard blir slått av. Dette er skrevet i php og kjører i apache2.



Figur 2 – Webgrensesnitt

6.3 MAVLink

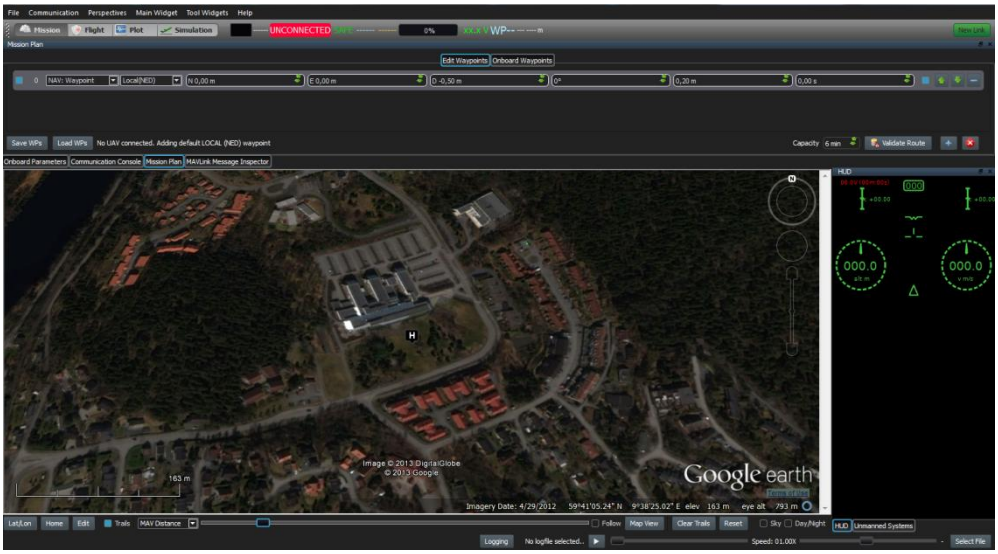
MAVLink-protokollen er bygd opp slik at man kan benytte egendefinerte meldinger i tillegg til de som er innebygd i biblioteket. Det finnes et stort antall meldinger definert, herunder systemkommandoer og waypoints. Både PX4 og QGC benytter MAVLink som kommunikasjonsprotokoll.

I vår løsning er det definert en ny melding som inneholder en sammenstilling av data ment for bruk ved geotagging av bilder. Denne meldingen kan i et senere tilfelle benyttes i bakkestasjonen for å markere hvilke deler av kartet som har blitt fotografert.

6.4 Bakkestasjon

QGroundControl (QGC) utvikles av et nettsamfunn i tett samarbeid med PX4, da de begge ble påbegynt i det samme prosjektet, PixHawk[3]. Funksjonaliteten i QGC er tilrettelagt for å fungere opp imot flere ulike autopiloter, såfremt de kan benytte MAVLink-protokollen. Det er dermed et bredere spekter av funksjonskall i QGC enn det som er implementert på PX4. Det er mulig å sende vilkårlige meldinger fra QGC, såfremt man benytter en MAVLink-ID på meldingen. IDen trenger ikke være definert da den kan skrives «on the fly».

I vår løsning er det lagt til noe funksjonalitet i bakkestasjonen. Det er laget en ny knapp som ligger i *Mission Plan widgeten* i QGC. Denne knappen starter en validering av oppdraget som er planlagt og estimerer om det vil være fysisk mulig for multirotoren å få til dette oppdraget. Denne kontrollen er designet som en offline test og kjøres på planleggingstidspunktet. Operatøren får en statusmelding tilbake fra testen som forteller hvorvidt vi tror det kommer til å gå bra å gjennomføre oppdraget. Sjekken omfatter en estimert maks flygetid. Den regner en konstant fart mellom veipunkter og konvertert dermed distanse til tid. Den beregnede flygetiden sjekkes opp mot estimert makstid. Den kontrollerer også om veipunkter ligger under bakken eller urealistiske store høyder.



Figur 3 – QGC

Sjekken blir kjørt uansett før oppdraget sendes til multirotoren. Det er kun en sjekk og det blir ikke gjort affære dersom testen mener oppdraget ikke er mulig. Denne avgjørelsen ligger hos operatøren.

Sjekken gjør en parsing av oppdraget og beregner ut lengden på oppdraget i meter og den summerer opp all tid der farkosten er satt til loiter (stå i ro).

6.4.1 Installasjon

Det er laget en egen installasjonsfil for QGC til Windows. Dette er nødvendig for at man skal kunne installere QGC med våre endrede *widgets.* I tillegg til dette får også QGC et spesielt STAR *splash*-bilde.

6.5 Dokumentasjon

Det er et ønske fra oppdragsgiver at kode skal dokumenteres med Doxygen. Da vi ikke hadde kompetanse på dette utnevnte vi et gruppemedlem til Doxygen-spesialist. Vedkommende utarbeidet et dokument som veileder i bruk av Doxygen, inkludert våre interne ønsker. Det har ikke blitt så mye egenprodusert kode som man kunne trodd da prosjektet startet, men det som er har blitt dokumentert i henhold til dette dokumentet.

7 RC-relatert

7.1 PID

Det å komme frem til en god PID-tuning for en flygende enhet med seks rotorer som har en usymmetrisk vektfordeling og skal takle vind er en utfordring. Det gikk dermed med endel tid på å oppnå en bedre forståelse av tuningparameternes innvirkning og å oppnå et akseptabelt prestasjonsnivå for plattformen.

PID-tuning er en prosess i seg selv der man gjennom gjentatte forsøk kommer frem til en optimal løsning.

7.2 Rigg

Inntil den flygende plattformen var stabil og vi var i stand til å kontrollere den manuelt, var det nødvendig å sikre deler (og mennesker) mot skade ved å sette opp ulike test-rigger. Det å bygge en rigg er også en kreativ prosess. Vi har benyttet paracord til å spenne opp kopterene i ulike rigger som lar oss teste oppførselen på en trygg måte.

7.2.1 Rigg 1

Første testrigg var en trepunkts forankring i det horisontale planet. Dette gjorde at vi kunne spinne opp rotorene uten fare for at kopteret skulle dra i noen som helst retning. Ved å gi slakk i forankringene kunne vi verifisere at kopteret hadde løft og til en viss grad at autopiloten stabiliserte.

7.2.2 Rigg 2

Topunkts forankring i horisontalplanet tillater test av stabilisering i én akse uten stor fare for person- og materiellskader.

7.2.3 Rigg 3

Etter stabilisering er verifisert som aktiv blir neste rigg en der kopteret får mer spillerom. Vi festet paracord til tre av armene og samlet disse over senterplaten, som en bæresele. Der festet vi så en lang paracord som kan slenges over tverrliggende bjelker eller liknende. En person sikrer så kopteret mot krasjlanding ved å dra inn slakk i paracorden etter hvert som kopteret stiger.

7.2.4 Rigg 4

Rigg 3 sikrer ikke kopteret mot å fly i taket eller i bjelken som paracorden ligger over. Rigg 4 er som rigg 3, men med en tilsvarende forankring i underlaget. Her brukte vi både Europall og stige. Denne riggen tillater å prøve ut funksjoner som *auto altitude hold* og liknende.

7.2.5 Rigg X – ikke ideelle løsninger

Vi har også testet to rigger som ikke var ideelle. Den første var en forlenging av landingsbena i horisontalplanet. Dette tillot flygning på veldig lav høyde uten fare for velt, men forlengelsene forstyrret luftstrømningene for mye, og turbulens ble en utfordring.

Den andre ikke ideelle riggen var en lukket loop. Denne er som rigg 4, men med den endringen at bæreselen i toppen og forankringen i bunn er et lukket system.

7.3 Verifikasjon av hardware

I tillegg til at det i prosjektet har vært mange komponenter som skal settes sammen og konfigureres, har det også vært nødvendig å verifisere at all hardware fungerer som den skal. Eksempelvis utførte vi test av telemetriradio ved å koble den opp mellom to PCer og benyttet PuTTY[8] til å kommunisere. På selve autopiloten oppdaget vi et svakt loddepunkt på kortet, og valgte å utbedre denne selv. Vi laget også et testprogram for BlinkM (statusmonitoren til px4) på PIC-18 MCU. Skolens PIC-DEM2 er utstyrt med PIC18F4520, denne brikken har I^2C som BlinkM benytter. Dette ble implementert for verifikasjon av BlinkM modulen.

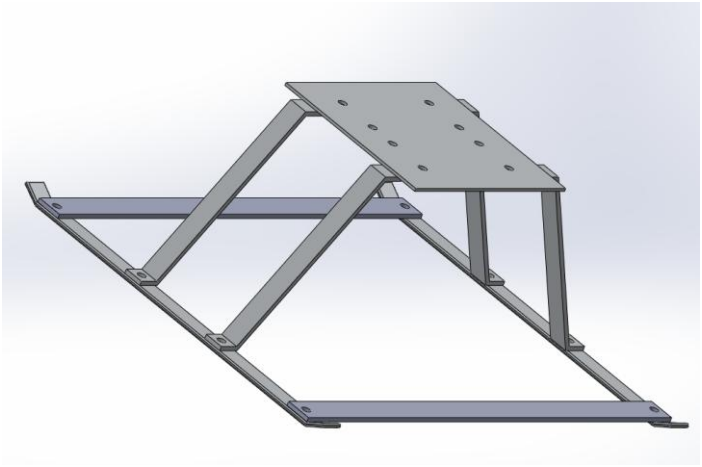
8 Fysiske anretninger

8.1 Konnektor for kamera

Kamera konnektoren til Leopard kameraet er en toraders 34-pinners konnektor med 2mm *pitch* (avstand mellom pinnene). For å få montert kameraet på undersiden av multirotoren ble det laget 2 spesialtilpassede kretskort forbundet med kabel ettersom BeagleBoardet står på oversiden. På disse kretskortene er det montert smd-konnektorer (smd: surface-mount device) til kameraet slik at dette blir en forlenger. Tilgangen på 34-pinners konnektorer i det formatet vi var på jakt etter var vanskelig å oppdrive ettersom det ikke var lagervare hos de store leverandørene. For å løse dette ble løsningen å kjøpe 40-pinners konnektorer. Konnektorene ble kappet og slipt til å bli 34-pinners konnektorer. Selve utlegget er håndtegnet og kretskortet er fremstilt på HiBus lab.

8.2 Understell

Byggingen av understell har gått fra 3D-tegninger, via kapping av aluminiumsplater ved hjelp av platesaks og håndverktøy, til ferdig monterte prototyper.



Figur 4 – Revidert understell

Når platene hadde fått rett dimensjon ble merking av hull kontrollert for å få riktig plassering. Dette arbeidet måtte gjøres nøye for å kunne feste delene til den flygende plattformen. Beinene er tilpasset ved å feste i en skrustikke og bøyd manuelt. Denne jobben krevde høy presisjon og kunne lett bli feil. Skinnene som skal ha kontakt med bakken må ha inn forsenkninger til festeskruer.

Figur 4 viser et redesignet understell. Redesignet kom etter at man skaffet seg flyerfaring og forsto hvordan kreftene ved både flygning og landing påvirker understellets egenskaper.

8.3 Bygging

Byggeprosessen fra ARF-kitet og fram til en flygedyktig maskin har inneholdt mange steg. Etter hvert som deler har kommet på plass så har det dukket opp flere utfordringer for å gjøre oppbygningen enklest mulig, både for å feste ting sikkert og for lett å kunne montere og demontere den flygende plattformen. Det at man har laget bananplugger i stedet for direktelodding har vist seg å være et veldig klokt valg.

Bygging av kasse for kamera var en tidkrevende prosess. Først ble pleksiglassplater kappet ut i korrekt størrelse og limet sammen med lim spesielt tilpasset denne jobben. Limingen er prosess som

gjerne tar døgn for hver av festene. Boring i pleksiglass er noe som må gjøres korrekt for å bevare den strukturelle integriteten til kassen.

8.4 Hexa-konfigurasjon

8.4.1 Topp

Tårnet på oversiden holder all elektronikken på plass slik den skal. Dette er en tung konstruksjon ettersom den ble laget av rustfritt stål, men plattformen har, som hexa-rotor, nok løftekraft til at dette ikke er noe begrensende faktor. Det vil derimot ha innvirkning på flygetiden og muligens manøverabilitet.

Tårnet består av to etasjer ettersom kopteret har med mye elektronikk opp i lufta. Den nedre etasjen under tårnet er dedikert til BeagleBoard. Utstikkeren på den nedre etasjen ble benyttet til å holde power-fordelingskort og 5V-BEC. Andre etasjen holder autopiloten, RC-mottageren, GPS og RF-Radio. Tårnet benytter fire lange skruer for å holde etasjene på plass og skruene er så lange at de skal ta imot hvis den flygende plattformen skulle rotere og lande på hodet. Denne konstruksjonen er ment for å knekke ved store støt og vil være nødvendig å bytte ut ved eventuelle skader.

8.4.2 Senter

Denne etasjen er dedikert som batteriholder og batteriet er den eneste komponenten som plasseres her. Batteriet akes inn og passer perfekt i høyde mellom de to senterplatene. Boltene som holder tårnet på plass går gjennom den øverste av senterplatene der de så holder batteriet på plass i side. I lengde er det montert strips som sikrer at batteriet ikke faller ut under flygning.

8.4.3 Bunn

Et landingsunderstell er både designet og bygget på den flygende plattformen. Dette gir mulighet for å ta av og lande skikkelig. Siden det er bygget i relativt mykt aluminium så vil dette også fungere som en støtdemper ved litt hardere landinger. Aluminiumskonstruksjonen har vist seg å ta i mot for harde landinger ved å bøye seg, men har latt seg rette opp i etterkant.

Kamerahuset er bygget i pleksiglass og skrudd fast i understellet. Det er laget ved å skjære ut rektangulære plater og lime disse sammen med spesial-lim som bedre skal tåle vibrasjoner, ettersom konstruksjonen er veldig utsatt for slike påkjenninger. Huset har en åpen side slik at kabel mellom kamera og BeagleBoard kan trekkes. Denne konstruksjonen ble skrudd fast under den nedre senterplaten på en slik måte at det var tilstrekkelig klaring til bakken.

Under implementasjonsfasen ble hexakopteret bygget ned til et quadkopter, som beskrevet i *S007 – Fysisk design,* etter en hard landing beskrevet i *P005 – Risikoanalyse*. Dette førte til en redesigning av komponentarrangeringen på den flygende plattformen.

8.5 Quad-konfigurasjon

Da hexa-rotoren krasjlandet og knakk senterplaten og en arm, bygget vi plattformen om til et kopter med fire armer, et såkalt quad-kopter, eller quad-rotor. Denne konfigurasjonen har en langt mindre senterplate og dermed et mindre areal til rådighet for å plassere komponenter. Denne ombyggingen ble derfor en utfordring. Hadde det ikke vært for det grundige arbeidet som ble gjort under byggingen av hexa-rotoren ville ikke byggingen av quad-rotoren vært mulig innen for de tidsrammene vi hadde til rådighet.

8.5.1 Topp

I stedet for et tårn sitter autopiloten og GPS skrudd fast i topplaten til hovedkroppen. Radio, 5V-BEC, og RC-mottaker har nå blitt satt ut på armene, så nærme senter som mulig.

8.5.2 Senter

Bildebehandlingsenheten er plassert i midten med ikke-ledende dempemateriale som holder denne komponenten på plass. Strømfordeling og kamera plasseres på to utstikkere fra underdelen av senterplata slik at disse ender opp på «utsiden» av senter etasjen. Disse stikker ut foran og bak. Kamera har fått plass på baksiden. Hva som er frem og bak, kan enkelt endres ved å invertere styresignalene for *aileron* (roll) og *elevation* (pitch) på RC-kontrolleren.

8.5.3 Bunn

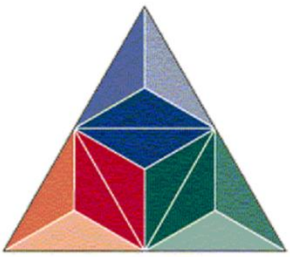
En spesialkonstruert bæreveske er festet fast til underdelen slik at batteriet ikke tar plass på festeplater og bidrar til å kunne senke tyngdepunktet for hele systemet. Denne holderen er laget av en indre del med glatt overflate slik at batteriet kan lett skli inn og ut av holderen. Den er så forsterket av fiberteip og har «paracord»-tau i mellomlagene. Paracorden benyttes til å lage et slitesterkt oppheng for batterikassen. Etter at batteriet er satt i bærevesken sikres det på plass av en elastisk hempe foran åpningen. Til slutt stabiliseres batteriet av en spesialisert Velcro-reim levert av batteriprodusenten. På denne måten oppnår vi et ideelt forhold mellom en enkel måte å skifte batteriet på og et sikkert oppheng for flygning.

I stedet for bein som går ut fra midten som på hexa-konfigurasjonen så har quad-oppsettet fått spikes festet ytterst i hver arm. Disse bør plasseres rett under motorene. Disse kan lages i aluminium og skrus fast i de små festehullene under motorene. Hver *spike* er laget av 2 aluminiumsplater som er gjennomhullet og boltet fast i eksisterende landingsben.

9 Referanser

1. S.T.A.R. *Wiki*. 16.05.2013]; Available from: http://home.hibu.no/atekstudenter1212/wiki.
2. Pixhawk. *GitHub PX4*. 16.05.2013]; Available from: http://github.com/px4.
3. Pixhawk. *PX4*. 26.02.2013]; Available from: http://pixhawk.ethz.ch.
4. ETH. *ETH Zürich*. 16.05.2013]; Available from: http://www.ethz.ch/index_EN.
5. Wikipedia. *PID*. 23.05.2013]; Available from: http://en.wikipedia.org/wiki/PID_controller#Loop_tuning.
6. Wikimedia. *Tait-Bryan angles*. 16.05.2013]; Available from: http://commons.wikimedia.org/wiki/Tait-Bryan_angles.
7. GitHub. *GitHub*. 24.05.2013]; Available from: https://github.com/.
8. Tatham, S. *PuTTY*. 24.05.2013]; Available from: http://www.chiark.greenend.org.uk/~sgtatham/putty/.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Utvidelser
Dokumenttype:	Systemdokument
Dokument ID:	S010
Versjonsnummer:	V1.0
Versjonsdato:	23.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	23.05.13	Første offisielle versjon

3 Introduksjon

Dette dokumentet har som hensikt å gi en oversikt over hvilken funksjonalitet som kan forbedres og mulige nye utvidelser som kan gjøres. I tillegg dekkes mulige anskaffelser av nytt materiell.

4 Utvidelser

Systemets funksjonalitet er beskrevet i utvikler- og brukerguidene som er opprettet på prosjektets wiki-sider, samt i dokumentet *S009 – Implementasjon*. Gruppen har hatt et bredt fokus i utviklingen av systemet. Mye jobb har blitt lagt i research og valg av deler, samt bygging og sammenstilling. Når systemet nå er i stand til å fly og det er opprettet grunnleggende kommunikasjon mellom de ulike komponentene, åpner det seg mange muligheter. Målet om å lage et *proof of consept* er oppnådd, konseptet fungerer. For den neste gruppen som skal utvikle systemet videre vil det være en god ide å sette et smalere fokus og gå dypere ned i et spesifikt fagfelt.

Plattformen vi har laget har fantastisk stort potensiale ettersom den har den mest avanserte åpenplattform autopiloten på markedet samt en meget kraftig embedded PC (1GHz ARMv7 – 512MB ram). Med dette som utgangspunkt er mulighetene for avanserte algoritmer for autonomitet oppnåelig. Det er allerede meget avanserte algoritmer implementert i autopiloten for estimering og navigasjon, men vi ser at utviklerne kontinuerlig adopterer de beste og mest grensesprengende algoritmene der ute.

BeagleBoard (BB), som også er i luften, er kraftig nok til å drive med en del krevende tallknusing, men vil nok ikke være kraftig nok for de mest krevende sanntidsalgoritmene for computer vision. Det er også et stort potensiale til å benytte BeagleBoard som en modul for å drive med machine learning. BB kan imidlertid muligens være kraftig nok for en del andre spennende problemstillinger man kan møte når man benytter en flygende plattform som basiselement for akademisk forskning.

Teknologidokumentet, *TEK006 – Introduksjon til AI*, inneholder mange spennende teknikker for problemløsninger rundt kunstig intelligens. Dette er absolutt noe som er relevant for videre utvikling av systemet og bør bli sett nærmere på.

Dette dokumentet ser nærmere på disse komponentene med fler i kapittelet om forbedringer av eksisterende system. Videre ser vi på hvilke nyanskaffelser, eller eventuelt utvikling av nye komponenter, som ville gjøre systemet og utviklingsmulighetene enda bedre.

5 Forbedringer av det eksisterende systemet

Både PX4-FMU og QGroundControl er modulært oppbygd og er relativt utviklervennlige. Da utviklingen foregår i et nettsamfunn, kan dokumentasjonen være noe manglende og spredt og vanskelig å sette seg inn i. Vi får vel kalle en spade en spade og si at dokumentasjon for open source programvare ofte rett og slett er dårlig – PX4 og QGC er dessverre ikke noe unntak. Programvaren er likevel svært godt bygd opp og gir et bra grunnlag for videre utvikling. Det at BeagleBoard, slik vi leverer det fra oss, kjører Ubuntu Linux gir også en relativt lav terskel for å utvikle mot denne. Ubuntu er en utbredt distribusjon og det bør være mulig å finne kompetente ressurser som kan veilede en.

5.1 BeagleBoard

BeagleBoard har vært testet med flere linux-kjerner i ulike versjoner. Den leveres med en Ångstrøm distribusjon, men kjører per i dag Ubuntu 12.10. Det er også mulig å legge inn Texas Instruments sin «BIOS» (deres minimale operativsystem) på denne plattformen dersom man ønsker nærmere kontakt med selve CPUen og DSPen i prosessoren; DM3730. Dersom man går for å benytte denne BIOSen kan man sannsynligvis få et mye større prosesseringsutbytte, men med den kostnaden at

utvikling er tyngre enn det vil være å jobbe i Linux, spesielt om man allerede har kompetanse på dette. Dersom man ikke har behov for all funksjonalitet som Linux byr på men er interessert i å implementere noen krevende algoritmer eller liknende, kan dette være en interessant retning å prøve.

Arbeidet med å prosessere bilder og video er ikke ferdigstilt, men det er med den tilgjengelige prosesseringskraften mulig å gjøre mer postprosessering ombord i plattformen.

Hurtigheten og responsiviteten til den flygende plattformen kan ha negativ effekt på bildene som tas og spesielt påvirke video. Det å implementere bildestabilisering er derfor noe som kan prioriteres. Om dette bør utføres i hardware eller programvare, og hva det vil kreve av prosesseringskraft er ikke avdekket.

Dersom man i tillegg utstyrer plattformen med en laser avstandsmåler, et kamera til, en (IR) prosjektor eller en kombinasjon av dette vil plattformen kunne benyttes til 3D oppgaver som å navigere i 3D rom.

Ettersom plattformen er utstyrt med kamera har plattformen *potensielt* (eller med noen utvidelser) muligheter for å drive med (for å nevne noen):

- Object tracking
- Pattern recognition
- Edge detection
- CV localization
- SLAM - Simultaneous localization and mapping
- 3D reconstruction

BeagleBoard er satt opp med WiFi og har dermed en begrenset rekkevidde for overføring av bilder og video. Det å utvide med en mobil datatilkobling (3G e.l.) vil kunne gi plattformen funksjonalitet som idag ikke er mulig. Overføring av bilder og video over vilkårlige avstander og til noe annet enn bakkestasjonen, f.eks. en sentralisert kilde som tar inn data fra flere UASer samtidig. Det vil også gi muligheten til at plattformen henter informasjon den trenger over internett. F.eks. kan kart og værdata oppdateres ombord og benyttes til å ta egne beslutninger om endringer i opprdaget.

Bakkestasjonen er idag på en bærbar arbeidsstasjon og oppdrag som planlegges der, overføres så til PX4. Det bør være mulig å sette opp BeagleBoard til å til å holde informasjon om oppdraget og også ta beslutninger om endringer av oppdraget. Hvis dette implementeres vil det være mulig å utføre fullt ut autonome oppdrag. Første skritt mot dette vil være å sette opp BeagleBoard til å kommunisere med PX4 via MAVLink.

5.2 PX4

PX4 er under stadig utvikling, og det kan forventes at funksjonaliteten er blitt utvidet og forbedret fra dette dokumentet skrives til det leses. Hvilke ting som bør forbedres på PX4 vil derfor være helt avhengig av når eventuell utvikling starter. Ettersom PX4 er et levende og pågående prosjekt i nettsamfunnet er det mulig å engasjere seg for å se hva man kan bidra med, og være med i utviklingen som alle kan dra nytte av.

PX4 er utstyrt med en Cortex-M4 CPU, noe som er blant det kraftigste på markedet i denne kategorien CPUer. Det betyr at som autopilotsystem er denne plattformen lagt mer fremsynt en de andre autopilotene på markedet per i dag. Visjonen til PX4-gruppen er å lage den «mest» avanserte plattformen ute på markedet, dette er ettersom plattformen originalt (annen versjon) ble utviklet for *computer vision* flyvning. Ettersom plattformen er såpass kraftig og godt planlagt er utvidelsesmulighetene på denne plattformen langt bedre en hos de andre autopilotene på markedet. Det må også nevnes at denne plattformen er åpen hardware og åpen software.

Det finnes også potensielt mange ukjente / uprøvde muligheter for denne autopiloten sett spesielt i forhold til *machine learning*. For eksempel adaptiv kontroll av tuning parametere for plattformen basert på de fysiske forholdene den opplever, for eksempel om du velger å fly i sterk vind. Eller muligheten for å kunne lære seg flykroppen den skal kontrollere (ved f.eks.: *unsupervised learning*).

5.3 Bakkestasjon

Gruppen har lagt til funksjonalitet i QGC for å validere gjennomførbarheten til planlagte oppdrag. Denne funksjonen gjør en statisk beregning av kapasiteten og benytter ikke geodata. Det kan med fordel utvikles en dynamisk beregning som tar hensyn til det faktiske batterinivået, aerodynamiske egenskaper for ulike plattformer (multirotor, helikopter, fly), fysiske kartdata, målte værforhold og temperatur, tidligere utførte oppdrag og batteriforbruk på disse. Tilgjengeligheten på data er den eneste begrensningen for hva man kan ta med i beregningen, og det kan skilles på data som krever ekstra sensorer om bord og data som er tilgjengelige over internett.

For å ta hensyn til f.eks. været kan alt løses på selve bakkestasjonen via nettbaserte leverandører av måledata fra lokale værstasjoner. For å bedre batterimålinger vil det trenges en ny sensor slik at vi kan måle strømforbruk, ett naturlig valg for dette er å benytte en *hall effect sensor*. Har man til enhver tid tilgang til strømforbruket er det mye enklere å modellere kostnader av flyvingen med hensyn på tilgjengelig strøm og man kan gi mye bedre estimater for gjenværende kapasitet.

6 Nyanskaffelser og utvidelser

6.1 Fysisk design

Flykroppen er per dags dato en quadrotor. Autopiloten skal være i stand til å stabilisere ulike flykropper (og eventuelt landgående kjøretøy). Det er derfor mulighet for å bygge opp en vilkårlig flykropp basert på de egenskaper man ønsker at plattformen skal ha. Det å kunne fly flere ulike flykropper i ulike størrelser med den samme programvaren gir muligheten til å endre designet av den flygende plattformen uten at dette gjør at elektronikken må byttes ut.

I overgangen fra hexa til quad, ble oppstartsscriptet på PX4FMU endret, og det ble utført ny PID-tuning. Ettersom alle tuningparametre kan lagres i bakkestasjonen, vil det være mulig å bytte frem og tilbake mellom plattformer hvis man bygger opp mer enn én plattform. Det å beskytte elektronikken mot regn ville gitt plattformen en økt nyttemulighet. Dette kan løses kjemisk, ved å beskytte de individuelle komponentene og kontaktpunktene; eller det kan løses mekanisk med en beskyttende flykropp. Hvis en kjemisk løsning er mulig vil man slippe innvirkning på de aerodynamiske egenskapene, men dette vil gå på bekostning av tap av kontaktpunkter som er viktige eller nyttige for utvikling og service. Om de aerodynamiske egenskapene best ivaretas av en kropp der luften strømmer fritt mellom alle komponenter eller en kropp der luften ledes av et ytre skall er uvisst.

Gruppen har laget noen enkle landigsbein av aluminium. Det var nødvendig å heve kroppen opp fra underlaget for å gi plass til batteriet. Disse beina kan med fordel byttes ut med noe som ikke har like store overflater, da disse gjør plattformen noe utsatt for vind. Disse beina bør vurderes i lette kompositmaterialer istedet for reint metall. Et nytt design bør ta hensyn til eventuell ny hardware som skal med, da dette kan endre behovet. I tidligere iterasjon, når den flygende plattformen var et hexakopter, hadde den et understell med noe svikt (svai). Fordelen med en mindre rigid konstruksjon er at den kan ta av for støt under en hard landing.

Propellene som brukes på plattformen idag er 8" standard *Dji flame wheel* propeller. Vi benytter også DJI-motorer og disse har en aksling som er annerledes en de andre leverandørene ettersom akslingen ikke er en glatt sylinder, men har heller 2 overstående utskjæringer av akslingen slik at propellene «låses» til akslingen. Dette medfører at valget av propeller er nesten tatt for oss, men det finnes propelladaptere på markedet. DJI propellene er relativt myke og elastiske, noe som gjør at de tåler en støyt, og dermed er perfekte hvis man er nybegynner. For presisjon og langtidsflygning kan det være en fordel å bytte til propeller av stivere plast eller kompositmateriale siden disse potensielt kan gi fartøyet et mer effektivt energibruk. Dette er ikke sikkert og heller ikke noe vi har testet.

6.2 Kamera

Valget av kamera (Leopard Imaging) var i stor grad styrt av pris og det at BeagleBoard har en konnektor som er spesielt laget for kamera fra Leopard Imaging. Det har vist seg at det er dårlig software-/driver-støtte for kameramodulen. Hvis det skal legges mer arbeid i bildebehandlingen vil det være en god ide å vurdere innkjøp av et bedre kamera, og ta mer hensyn til operasjonshøyde i valget av linse. Auto fokus og justerbar optisk zoom vil potensielt kunne gi bedre nytte av systemet.

Det kan være ønskelig å ha en *gimbal* til kameraet, men en slik anordning vil medføre mer vekt og høyere batteriforbruk. Dette er derfor kun anbefalt hvis plattformen bygges opp til å ha mer løftekapasitet og samtidig utvides med mer batterikraft. Fordelen med *gimbal* er muligheten for langt mer stabilt bilde og mye enklere *reference-tracking* og enklere *object-tracking*. De *high end gimbalene* som er på markedet (for hobbyentusiaster) finnes i flere prisklasser, men de som virkelig er presise har fortsatt en prislapp som gjør dette utstyret ganske utilgjengelig for folk flest.

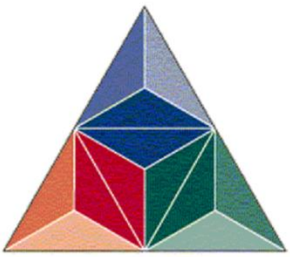
6.3 PX4FLOW

PX4 har utviklet en tilleggsmodul kalt PX4FLOW. Dette er en optisk sensor (omtrent som en PC-mus). Den kan benyttes til å holde posisjon i planet ved lave høyder og kan gi nøyaktige posisjonsdata uavhengig av GPS og kan dermed benyttes til autonom flygning innendørs. Da dette er en relativt ny modul er det fortsatt store muligheter til å utvikle egen kode og lage funksjonalitet som ikke allerede eksisterer.

6.4 RC-kontroller

RC-kontrolleren som brukes til systemet idag har bare 5 tilgjengelige kanaler. For å utnytte autopilotens fulle potensial, burde en kontroller med flere kanaler anskaffes. Hovedgrunnen til dette er de forskjellige tilstandene autopiloten kan være i, samt gi mulighet for bilde/video kommando. Tilstandsdiagrammet for PX4 finnes i vedlegget til *S005 – Programvaredesign.* Hvis systemet skal utvikles mot å ha mer funksjonalitet, f. Eks servoer, gimbal osv. Vil det også være nødvendig å ha en kontroller med fler kanaler. For utvikling på PX4 (altså egne apper) har vi sett nytten av å kunne koble oss inn på RC-signaler for å styre / teste ulike moduler i koden.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Prosjektplan
Dokumenttype:	Planleggingsdokument
Dokument ID:	P006
Versjonsnummer:	V3.0
Versjonsdato:	23.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	04.01.13	Første offisielle versjon
V2.0	06.03.13	Oppdatert ifm progresjon i prosjektet
V3.0	23.05.13	Oppdatert sprintverskiter

3 Involverte personer

Gruppens medlemmer	
Navn:	Carl Sandaker
Ansvar:	Product owner
Linje:	Virtual Systems
Telefon:	93862663
E-post:	csandaker@gmail.com
Navn:	Jørgen Markussen
Ansvar:	Scrum master
Linje:	Virtual Systems
Telefon:	41644541
E-post:	jormar2k@gmail.com
Navn:	Espen Klein Nilsen
Ansvar:	Team member
Linje:	Cybernetics and Mechatronics
Telefon:	92205465
E-post:	espen.k.nilsen@gmail.com
Navn:	Gunnar Sandaker
Ansvar:	Team member
Linje:	Embedded Systems
Telefon:	40408224
E-post:	gsandaker@gmail.com
Navn:	Liam Jensrud
Ansvar:	Team member
Linje:	Audio Technology
Telefon:	41354537
E-post:	liam.jensrud@gmail.com

Andre personer	
Ekstern Veileder/ sensor	Per Wollebæk
Intern Veileder	Karoline Moholth
Intern Sensor	Hallstein Hansen
Representant KDS	Trond Neumann

4 Introduksjon

Prosjektet S.T.A.R. er et hovedprosjekt utført ved Høgskolen i Buskerud, Avdeling Kongsberg, Fakultet for Teknologi. Studentene går siste året i en Bachelorutdanning innen ingeniørfag. Oppdragsgiveren for prosjektet er Kongsberg Defence Systems.

Gjennom forstudien *P002 – Forstudie* har vi satt oss inn i hva dette prosjektet går ut på. Vi har sett på mange ulike løsninger og falt på noen valg som definerer hva vi skal utvikle og hvordan vi skal gjennomføre prosjektet. Dette dokumentet vil forklare hvordan vi planlegger å gjennomføre prosjektet og vil oppdateres og endres etter hvert som utviklingen går fremover og utfordringer må takles.

Prosjektplanen vil støtte seg på flere frittstående dokumenter, slik at det er mulig å fordype seg i de ulike temaer ved å lese dokumentene de bygger på.

5 Mål for prosjektet

Prosjektet utføres på oppdrag for en ekstern bedrift samtidig som det utgjør studiepoeng ved høgskolen.

Ovenfor oppdragsgiver har vi som målsetning å levere et system som oppfyller kravene som er gitt (se dokument *P003 - Kravspesifikasjon*). Det er også et mål at systemet skal tilfredsstille de forventninger oppdragsgiver har utover kravene. For å klare det må vi ha en god dialog med oppdragsgiver gjennom prosjektet og holde dem oppdatert på fremdriften i prosjektet.

Høgskolen stiller visse krav til gjennomføringen av prosjektet. Kravene fra skolen er i høyeste grad kvantitative, ikke kvalitative. I hvilken grad vi oppfyller disse kravene vil påvirke karakterene vi oppnår. Det er derfor et mål for gruppen å gjennomføre prosjektet på en slik måte at vi tilfredsstiller høgskolens kvalitative forventninger i tillegg til de kvantitative kravene.

Som studenter ønsker vi å fremstå som gode kandidater for fremtidige arbeidsgivere. Det er derfor et mål å bruke prosjektet til å vise de kunnskaper og egenskaper vi har tilegnet oss gjennom studiene. Samtidig vil vi lære mer om prosjektarbeid generelt og våre tekniske fagområder spesielt.

6 Prosjektmodell

Valget av prosjektmodell ble tatt i begynnelsen av november. Vi syntes det var viktig å bruke tid på å lese nok om de ulike modellene. På den måten kunne vi gjøre et velinformert valg. Vi valgte å bruke Scrum [1]. Scrum er en smidig utviklingsmodell som er godt egnet for programvareutvikling, og prosjekter som er vanskelige å detaljplanlegge i starten. Det er en inkrementell utviklingsmodell som legger stort ansvar på hver enkelt deltaker.

Det er visse ulemper, eller fallgruver, med Scrum. For eksempel legger Scrum opp til hyppige, korte møter. Disse kalles Daglig Scrum. Hvis man ikke klarer å holde møtene så korte som de skal være, kan de gå ut over produktiviteten. Det er viktig at modellen og verktøyene man benytter er til hjelp, ikke til hinder for arbeidet som skal utføres.

Iterasjonene i Scrum kalles Sprinter. Hver Sprint starter med et planleggingsmøte og avsluttes med et evalueringsmøte, samt et tilbakeblikk. I evalueringsmøtene skal gruppen vise hva som er oppnådd i løpet av Sprinten. På slutten av evalueringsmøtet tar man et tilbakeblikk på hvordan man har jobbet. Hensikten med tilbakeblikket er å avklare om noe må endres. Vi har i utgangspunktet valgt å holde to uker lange Sprinter, men gjør tilpasninger på enkelte sprinter da arbeidsmengden pr uke vil endres gjennom semesteret.

Mer utdypende informasjon om Scrum blir beskrevet i *PR002 – Prosjektmodell, Overskrift 7.*

6.1 Verktøy for prosjektstyring

En viktig del av det å bruke Scrum, er for oss verktøyet OnTime [2]. OnTime er et online verktøy som organiserer krav, oppgaver og brukere i en database. OnTime gir en visuell oversikt over alle oppgaver og deres relasjoner til personer, frister, komponenter osv. En full brukermanual for OnTime er tilgjengelig på nett [3]. Kostnadene for bruk av OnTime er beskrevet i dokumentet *OK001 – Budsjett*.

7 Avgrensninger

Kravene fra oppdragsgiver beskriver det endelige systemets egenskaper. Det vil likevel finnes mange løsninger på oppgaven. Vi i gruppen må ta mange valg angående hvordan vi ønsker å løse oppgaven og hva vi ønsker å fokusere på. Det ferdige systemet skal bestå av mange komponenter. Vi ser det ikke som hensiktsmessig å utvikle alt fra bunn av. Noen komponenter kjøpes ferdige og må tunes eller det må utvikles et grensesnitt for at de skal kommunisere sammen i det ferdige systemet. Det å gjøre systemet i stand til autonom flygning vil trolig være det området som krever mest utvikling. Oppgaven vil dermed ha sterkest fokus innen programvareutvikling. Det ferdige systemet vil være et sanntidssystem med harde sanntidskrav. Vi velger derfor å ikke utvikle sensorer, aktuatorer og annen hardware selv.

Gjennom design- og utviklingsfasen mellom første og andre presentasjon har vi sett at noen arbeidsområder krever mer tid enn først antatt. Selve sammenstillingen av det innkjøpte hexakopteret krevde at vi designet og bygget monteringsanretninger for alle komponenter som den flygende plattformen består av. Dette medførte at første planlagte testflygning av den fysiske plattformen ble forsinket. Oppgavene i prosjektet er likevel planlagt på en slik måte at dette ikke bremser utvikling på andre områder.

7.1 Økonomi

Oppdragsgiver ga i utgangspunktet et budsjett på NOK 10.000,-. Vi satte opp tre budsjettforslag med ulike komponenter som vi tok med til oppdragsgiver. Basert på disse ble budsjettet revidert til en total på NOK 15.000,-. Sammen med oppdragsgiver kom vi til enighet om en flytype og et rammeverk for hva som er nødvendig av komponenter. Budsjettet er beskrevet i dokumentet *OK001 – Budsjett*.

Gruppen har opprettet en egen konto og oppdragsgiver har overført pengene til oss. Vi må så produsere et regnskap og fremvise kvitteringer for det som er brukt i løpet av prosjektet.

8 Arbeidsmengde og oppfølging

Prosjektarbeidet utgjør 20 studiepoeng, dette tilsvarer ca. 550 timers arbeid per student. Vi er fem studenter i gruppa som gir en total på 2.750 timer. Kvaliteten på arbeidet som utføres måles ikke i timer. Det er derfor viktig å kunne spore hva timene har gått med til.

Vi har benyttet en felles timeliste fra starten av. I denne fører vi timer fortløpende og fører inn hva som er jobbet med. Denne gir en god oversikt over det totale antallet timer som er brukt, samt hvor mange timer hver enkelt student har jobbet. I verktøyet OnTime logger vi timer på hver spesifikke oppgave. Oppgavene har et originalt estimat, som oppdateres og eventuelt korrigeres underveis. Såfremt at vi bruker verktøyet riktig, skal timene i OnTime og timene i timelista stemme overens. Hvis vi oppnår dette er timelistene overflødige. Verktøyet OnTime kan produsere timelister og statistikker basert på loggførte timer. Disse kan vi ta ut periodisk slik at både vi og intern veileder har oversikt over hva som skjer i prosjektet.

Gjennom prosjektet gang, har vi merket oss at det er hensiktsmessig å fortsette å føre timelister. Det er mange oppgaver, som f.eks. møtevirksomhet som ikke fremgår i OnTime, men likevel skal føres timer på. Sammenligner man med en arbeidssituasjon, vil timelistene gjenspeile arbeidsdagene, og OnTime vil gjenspeile fakturerbare timer i prosjektet.

8.1 Møter

Prosjektgruppen holder interne møter i henhold til Scrum. Gruppen har møter med intern veileder ukentlig, innkalling med agenda sendes ut to virkedager i forveien.

Ekstern veileder inviteres, med fire dagers varsel, til evalueringsmøtene etter hver Sprint. Vår eksterne veileder har erfaring med Scrum. Dermed kan han bidra både teknisk og i selve prosessen.

8.2 Star-labs.no

Gruppen har opprettet en nettside [4] for å kunne kommunisere med alle som ikke er direkte involvert i prosjektet. Nettsiden har grunnleggende informasjon om hva prosjektet går ut på. Vi legger ut bilder og oppdateringer på forsiden slik at siden skal gi et dynamisk inntrykk. Sidene inneholder også et lukket forum som gruppen benytter til intern kommunikasjon. Det er et krav fra skolens side at nettsidene til prosjektet skal bevares utover prosjektets varighet. Nettsidene vil dermed bli lokalisert på skolens server, og tilgjengelig gjennom HiBu sine nettsider så vel som domenet vi selv har satt opp.

9 Prosjektets faser

Vi har delt prosjektet opp i tre hovedfaser. Fasene samsvarer med presentasjonene som skal holdes. Presentasjonene markerer slutten på en fase, men fasene vil likevel kunne flyte over i hverandre der dette er hensiktsmessig.

9.1 Fase 1 – Forstudie

I fasen frem til første presentasjon den 11. januar 2013, skal vi få en oversikt over hva prosjektet går ut på. Dette er en fase med mye planlegging og tilvenning til verktøy og metoder. Ved avslutningen av denne fasen skal vi ha kommet godt i gang med gruppearbeidet. Vi skal også ha bestilt det mest kritiske av hardware.

9.2 Fase 2 – Utvikling

Fra første presentasjon og frem mot påske vil vi i hovedsak drive design og utvikling. Ettersom vi jobber i Scrum så skal alt som utvikles også testes underveis. Metodikk for testing er nærmere beskrevet i *P004 – Testspesifikasjon*. Presentasjon nummer to, som skal holdes før påske, kommer til å gå nærmere inn på de tekniske detaljene i hvordan vi løser oppgaven. Mot slutten av denne fasen skal vi ha et fungerende system med grunnleggende funksjonalitet.

9.3 Fase 3 – Ferdigstille

I den siste fasen skal systemet ferdigstilles og systemtestene skal gjennomgås. Det vil også være en del arbeid med å samle dokumentasjon til et helhetlig dokument før endelig innlevering, samt utføre en etteranalyse. Det kan være hensiktsmessig å avslutte utviklingen av nye ideer etter påske og fokusere på å ferdigstille det vi har designet.

10 Fremdrift

Den detaljerte planlagte og estimerte fremdriften i prosjektet beskrives i dokumentene *Sprint Planning* og *Sprint Review*. Den estimerte prosjektstatusen for leveranse vil for hver Sprint fremgå av *Sprint Planning*. Statusen estimeres på grunnlag av hvilke A- og B-krav vi estimerer å fullføre, basert på hvor mange timer vi har til rådighet i prosjektet. Disse dokumentene produseres av *Scrum Master* i henholdsvis starten og slutten av hver sprint. Disse dokumentene leveres til intern veileder fortløpende.

10.1 Sprinter

En mer detaljert oversikt og når de forskjellige sprintene starter og avslutter samt hvor mye ressurser som er tilgjengelig i hver sprint er en viktig del av planlegging av prosjektet. Sluttdatoen er en endelig dato og kan ikke flyttes etter at sprinten har startet.

Sprint	Fra:	Til:	Arbeidsdager	Merknader
3	16.01	29.01	4	
4	30.01	12.02	4	
5	13.02	05.03	6	Sprint-lengde økt til 3 uker
6	06.03	14.03	6	En «minisprint» mot presentasjon 2, jobbes alle dager
7	15.03	17.04	6	Påske fra 25.3 tom.1.4. Eksamen ca 4.4 tom. 12.4
8	18.04	30.04	8	Sprint lengde redusert til 2 uker
9	01.05	14.05	8	
10	15.05	28.05	8	Tentativ frist for ferdigstilling 17.05
11	29.05	04.06	3	Tredje presentasjon 03.06

10.2 Ressurser tilgjengelig

Fulle arbeidsdager vil tilsvare 7,5 timer pr person. Dette betyr da at gruppa som helhet har 40 timer tilgjengelig pr arbeidsdag. Før påske vil dette gi ca. $37,5t \cdot 4dager = 150t$ pr sprint. Etter påske kommer en eksamensperiode hvor prosjektet vil være mindre i fokus. Etter denne perioden vil det være 10 arbeidsdager i hver sprint. Dette gir en total på ca. $37,5t \cdot 10dager = 375t$ pr sprint.

Timene vi har satt opp per person per sprint er en mal vi jobber ut fra. Et mer nøyaktig estimat gjøres under sprint planning møtet som holdes før hver sprint. På denne måten kan vi justere opp eller ned timeberegningene basert på individuelle behov.

10.3 Planlagte milepæler

10.3.1 Bilde tatt og lagret

Bilder som er tatt med bestilt kamera vil være lagret på bildebehandlingsenhet.

10.3.2 Egen kode på PX4

Vi har klart å kjøre egen kode på autopilotsystemet vårt.

10.3.3 Plattformen tar av/lander autonomt

Den flygende plattformen skal nå kunne både lette og lande uten interaksjon fra operatør på bakkestasjon.

10.3.4 Demonstrasjon for skolen.

Den flygende plattformen skal kunne virke slik at vi kan invitere til en åpen livedemo ved skolen.

10.3.1 Oppdrag fra bakkestasjon

Den flygende plattformen skal nå kunne greie å motta oppdrag og så utføre oppdragene.

10.3.2 Merking av video/bilde

Bildene og videoen som er tatt med eget kamera har fått metadata.

10.3.3 Bilde/video

Laste ned bilde og video under flygning er nå ferdigstilt og kan demonstreres.

10.3.4 Livevisning

Kamera sender livebilder fra plattformen og til bakkestasjon.

11 Dokumenter

I forbindelse med prosjektarbeidet er det mange dokumenter som må opprettes. Vi har opprettet dokumentmaler for å holde et uniformt utseende på alle dokumenter. Vi har alle dokumenter liggende i Dropbox. Dette gjør at alle dokumenter kan gjenopprettes til tidligere tilstander. En utfordring med Dropbox er at flere brukere ikke kan jobbe i samme dokument på samme tid. Hvis dette viser seg å bli et problem kan vi bli nødt til å se på andre løsninger.

Vi benytter Microsoft Office internt i gruppen, slik at interne dokumenter er i Word-format. For innleveringer produserer vi PDF-filer. Vi benytter *EndNote X6* [5] til referanser. Word har innebygde funksjoner for å legge inn merknader i teksten, disse benytter vi for å kommentere hverandres arbeid, og dermed ytterligere kvalitetssikre dokumenter samt beholde sporbarheten til endringer. Word har også funksjonalitet for å sammenligne og flette sammen ulike versjoner av et dokument.

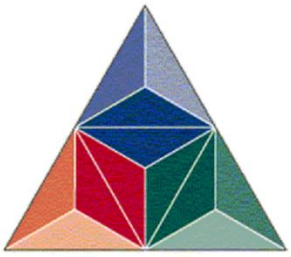
I mappestrukturen vi har bygd for prosjektet har vi valgt å gi hvert dokument en ID som en del av navnet. Eksempelvis heter dette dokumentet *P006 – Prosjektplan*. Dette er dermed det sjette planleggingsdokumentet vi opprettet. Etter navnet legger vi til et versjonsnummer på formen *P006 – Prosjektplan v0.3*. Dokumenter som er klare for innlevering får et helt tall, f.eks.: *v2.0*.

Hvert dokument har en medfølgende logg der vi legger inn en kommentar hver gang vi har lagt til noe eller endret noe. Det gjør at det er mulig å spore endringer og se hvem som har medvirket til dokumentet. Loggdokumentene er uformelle og det utføres ingen korrektur på disse.

12 Referanser

1. Scrum.org. *What is Scrum*. 28.11.2012]; Available from: http://www.scrum.org/Resources/What-is-Scrum.
2. Axosoft. *OnTimeNow*. 22.11.2012]; Available from: http://www.ontimenow.com/.
3. Axosoft. *OnTime for Web User's Guide*. 05.12.2012]; Available from: http://www.ontimenow.com/support/docs/web12/flash/ontime_help.htm.
4. S.T.A.R. *System for Tactical Aerial Reconnaissance*. Available from: http://star-labs.no/.
5. Thomson_Reuters. *EndNote X6*. 04.12.2012]; Available from: http://endnote.com/.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Risikoanalyse
Dokumenttype:	Planleggingsdokument
Dokument ID:	P005
Versjonsnummer:	V3.0
Versjonsdato:	23.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	04.01.13	Første endelige versjon.
V2.0	03.03.13	Lagt til inntrufne risiko
V3.0	24.04.13	Uferdig kildekode

3 Introduksjon

Dette dokumentet skal se på og drøfte ulike risikoer forbundet med prosjektgjennomføringen. Risiko dekker alle aspekter, fra hva som skjer hvis hardware ikke kommer som bestilt, til frafall av medlemmer i prosjektgruppen. Dokumentet vil være en del av *P002-Forstudie*, samt følges opp gjennom prosjektets gang.

For å analysere risiko ser vi på to faktorer; Sannsynligheten for at en hendelse inntreffer, og alvorlighetsgraden av en hendelse. Alvorlighet går fra 1-4, hvor 1 er lite alvorlig og 4 er veldig alvorlig. Sannsynligheten ligger i tre nivåer; 1. Liten sannsynlighet, 2. Middels sannsynlighet, og 3. Veldig stor sannsynlighet.

Eksempel:

Alvorlighet (4) x Sannsynlighet (2) = Total risiko (8)

Eksempeltabell:

Vurdering:	**Alvorlighet: 4**	**Sannsynlighet: 3**	**Total risiko: 12**
Risiko:	Her settes risikoscenarioet opp og forklares		
Forebyggende Tiltak	Her diskuteres diverse løsninger for å minimalisere risikoen i forrige felt.		
Ved inntruffet hendelse:	Hva gjøres om/når risikoen treffer inn		

4 Risikoområder

4.1 Gjennomføring

4.1.1 Prosjektet

Vurdering	Alvorlighet: 4	Sannsynlighet: 1	Total risiko: 4
Risiko:	Hvis ett eller flere av risikoscenarioene inntreffer og tiltak ikke settes i gang, vil man risikere at prosjektet ikke godkjennes.		
Forebyggende tiltak	Følge opp risikodokumentet.		
Ved inntruffet hendelse:	Hvis prosjektet ikke er godkjent må studentene tilbake med et nytt prosjekt neste år.		

4.1.2 Forarbeid og utforskning

Vurdering	Alvorlighet: 3	Sannsynlighet:3	Total risiko: 9
Risiko:	Dette prosjektet strekker seg over veldig mange disipliner og fagfelt, så det er en risiko for at utforskning av løsninger ikke blir grundige nok på den tiden som er tilgjengelig. Dette kan føre til at løsningene ikke er implementerbare.		
Forebyggende tiltak	Sørge for å dokumentere vurderte løsninger selv om de blir avvist på et tidlig stadium.		
Ved inntruffet hendelse:	Gå igjennom mulighetene man ikke har vurdert så grundig og se om dette kan implementeres i systemet på nåværende tidspunkt		

4.2 Sykdom/frafall

4.2.1 Sykdom

Vurdering	Alvorlighet: 3	Sannsynlighet: 3	Total risiko: 9
Risiko:	Sykdom og skade er en naturlig del av grupper med mennesker i. Hvis ett eller flere gruppemedlem blir for syk til å jobbe 100 % medfører dette en risiko for å gjennomføre deler av prosjektet. Enkelte gruppemedlemmer har allerede rapportert inn potensielle medisinske problemer.		
Forebyggende tiltak	Kommunikasjon er viktig! Jo mer gruppa er informert om relevante medisinske problemer, desto lettere er det å planlegge med dette i bakhodet.		
Ved inntruffet hendelse:	Skulle man bli uforutsett syk slik at man ikke er i stand til å jobbe, må gruppa organisere om slik at det aller viktigste arbeidet blir gjort, dvs. A-krav.		

4.2.2 Frafall av gruppemedlemmer

Vurdering	Alvorlighet: 4	Sannsynlighet: 2	Total risiko: 8
Risiko:	Valget av Scrum som prosjektmodell gir en del friheter i forhold til dette. Her vil frafall av gruppemedlemmer ha en mindre påvirkning enn ved for eksempel Unified Process. Siden ingen har helt faste ansvarsområder og arbeidsoppgaver, men er på hugget for å få tak i oppgaver, så vil frafall av ett gruppemedlem ikke være så dramatisk. Gruppa blir jobbende hardere for å nå minstekravet, som igjen kan gå utover den endelige karakteren for gruppa.		
Forebyggende tiltak	Frafall og sykdom henger ofte sammen, men man kan falle fra av andre grunner. Disse grunnene bør informeres til gruppa så fort de er kjente, slik at gruppa kan planlegge og legge om arbeidet slik at man kan bli ferdig med kravene.		
Ved inntruffet hendelse:	Forutsette frafall må planlegges inn så tidlig som mulig. Uforutsette frafall gjør at gruppa må omorganisere slik at man blir ferdig med minimum A-krav i tide.		

4.3 Software

4.3.1 Programmer

Vurdering	Alvorlighet: 3	Sannsynlighet: 3	Total risiko: 9
Risiko:	Ved bruk av programmer med åpen kildekode kan man risikere ustabilitet som gjør at man kan bli sittende fast av uforutsette problemer. Dette skjer gjerne ved de nyeste versjonene av programmene hvor det finnes ukjente bugs og glitcher. Ved lisens-programmer så er denne risikoen mindre, men fortsatt til stede.		
Forebyggende tiltak	Hvis man kun oppdaterer programvare når man må, så vil man holde seg til versjoner som er kjent stabile. Ligge noen versjoner bak nyeste versjonen vil minimalisere bug-risikoen. Alt skal ha minst én backup!		
Ved inntruffet hendelse:	Mister man arbeid så går man tilbake til siste backup og fortsetter derfra. Vurder andre programmer.		

4.3.2 Implementering

Vurdering	Alvorlighet: 2	Sannsynlighet: 3	Total risiko: 6
Risiko:	Her kan man støte på problemer ved at utviklet programvare krasjer eller kommunikasjon mellom enhetene bryter sammen.		
Forebyggende tiltak	Kan løses ved å implementere programvare så fort det er ferdig, og teste at ting fungerer fortløpende.		
Ved inntruffet hendelse:	Defekter havner i «defect backlog», ny prioritetsvurdering må utføres før videre arbeid med enheten/ modulen fortsetter.		

4.3.3 Avhengighet utenifra

Vurdering:	Alvorlighet: 4	Sannsynlighet: 3	Total risiko: 12
Risiko:	Siden prosjektet baserer seg i all hovedsak på open source kode under stadig utvikling, så kan vi risikere at noe funksjonalitet til systemet ikke er tilgjengelig innen prosjektperioden.		
Forebyggende Tiltak	Holde seg oppdatert om hva som skjer i de ulike «samfunnene» på hva som er utviklet og hva som er tilgjengelig.		
Ved inntruffet hendelse:	Implementere funksjonaliteten så langt det lar seg gjøre med det som finnes. Lete opp utestet betaprogramvare fra utviklere på ting man vet det blir jobbet med, og utvikle/debugge selv for å få på plass ønsker funksjonalitet		

4.3.4 Styring

Vurdering	Alvorlighet: 3	Sannsynlighet: 3	Total risiko: 9
Risiko:	Styring er spesielt utsatt, fordi dette kan få hele plattformen til å styrte. Dette kan ødelegge armer, rotorer, og motorer		
Forebyggende tiltak	Ha kontroll på plattformen når styringen testes. Ha en form for oppheng og sikkerhet slik at man kan finne feil uten å ødelegge plattformen. Testing bør heller ikke foregå i høye hastigheter. Grundig testing underveis i utviklingen er essensielt.		
Ved inntruffet hendelse:	Finne ut hvilket element som feilet (styring, testrigg eller plattform). Rett opp hva som er feil, evt. Bytte ødelagte deler.		

4.4 Hardware

4.4.1 Leveranser

Vurdering	Alvorlighet: 4	Sannsynlighet: 2	Total risiko: 8
Risiko:	Den største risikoen ved all hardware som skal bestilles fra eksterne leverandører vil være leveringstider og lagerbeholdning. Hvis sentrale hardwaredeler ikke er på plass i tide, vil framgangen i prosjektet delvis eller fullstendig stoppe opp. Dette vil igjen påvirke karakteren.		
Forebyggende tiltak	Bestill i god tid. Bestill så mye som mulig fra norske butikker, da leveringstiden pleier å være kortere.		
Ved inntruffet hendelse:	Mangler man hardware, arbeid så langt med det man har eller jobb rundt det man mangler. Både dokumentasjon og hardware. Får man ikke ting fra en butikk må man finne andre butikker som kan levere.		

4.4.2 Plattformer

Vurdering	Alvorlighet: 2	Sannsynlighet: 3	Total risiko: 6
Risiko:	Jo flere antall rotorer og motorer, desto mer sannsynlighet for at enkelte deler går i stykker eller har defekter som ikke kan oppdages før man har tatt produktet i bruk.		
Forebyggende tiltak	Kjøpe inn et fornuftig antall reservedeler. Ha mulighet for å kjøpe inn flere fra norske nettbutikker med kort leveringstid.		
Ved inntruffet hendelse:	Finnes det ikke flere på lager må nye butikker letes opp. Vi har også muligheten til å bygge om fra hexakopter til quadkopter, slik at vi har noe fysisk ferdig.		

4.4.3 Styringsenhet

Vurdering	Alvorlighet: 4	Sannsynlighet: 2	Total risiko: 8
Risiko:	Styringsenheten har usikkerhetsmomenter. Spesielt med tanke på følsomheten på elektronikken til styringschipen. Det er en risiko for ukjente defekter. Hvis styringsenheten ikke er programmert rett kan den flygende plattformen feile.		
Forebyggende tiltak	Elektronikk skal behandles som ESD-sensitivt, og sikres mot fysisk skade under bruk.		
Ved inntruffet hendelse:	Får vi ikke hva vi trenger, eller at det vi får er defekt så har vi muligheten til å teste enkelte deler ved midlertidig å benytte en privat CC3D som gruppa har tilgjengelig slik at denne delen av prosjektet ikke stopper opp.		

4.4.4 Kamera

Vurdering	Alvorlighet: 2	Sannsynlighet: 2	Total risiko: 4
Risiko:	Hvis levering av kamerautstyr drøyer for lenge, eller er defekt når vi mottar det, så risikerer vi at deler av prosjektet blir forsinket eller ikke gjennomført.		
Forebyggende tiltak	Lete opp flere leverandører for å gi flere muligheter for å skaffe kamera.		
Ved inntruffet hendelse:	Feste et vilkårlig kamera som kan enten ta bilder/video og lagre det på den flygende plattformen. Dette vil være en midlertidig løsning som et "proof of concept" Slik at man ikke stopper fullstendig opp.		

4.4.5 Batteri

Vurdering	Alvorlighet: 4	Sannsynlighet: 1	Total risiko: 4
Risiko:	Dersom batteriet til hexacopteret blir ladet på feil måte, eller om andre feil skulle oppstå i ladeprosessen er det en fare for at batteriet tar fyr. Dette forårsaker tapt batteri, men kan også forårsake brannskader på annet materiell og personell.		
Forebyggende tiltak	Tilstrekkelig opplæring på korrekt bruk av batterilader før bruk. Batteri lades alltid i brannsikker pose.		
Ved inntruffet hendelse:	Bestille nytt batteri fra raskeste leverandør. Dette får konsekvenser både for prosjektplan og budsjett.		

4.4.6 Flygning

Vurdering	Alvorlighet: 3	Sannsynlighet: 2	Total risiko: 6
Risiko:	Friflygning ute medfører en del risikoer fra vær og vind. De fleste kan resultere i styrt og/eller ødeleggelse av den flygende plattformen eller enkeltkomponenter.		
Forebyggende Tiltak	Kjøre så mange kontrollerte innetester som mulig før man går ut. Kjøre ute i så stabile værforhold som mulig. Beskytt følsom elektronikk så godt som mulig		
Ved inntruffet hendelse:	Her kan man også bygge ned fra hexakopter til quadkopter om det skulle mangle på reservedeler.		

4.4.1 Manuell flygning

Vurdering	Alvorlighet: 3	Sannsynlighet: 2	Total risiko: 6
Risiko:	Ved kun å ha én pilot som kan fly manuelt så kan man lett risikere at testflygning stopper opp/tar lengre tid enn nødvendig om denne personen blir hindret av uforutsette forhold som skader eller ulykker.		
Forebyggende Tiltak	Lær opp minst to testpiloter slik at de begge kan brukes for testing		
Ved inntruffet hendelse:	Hvis begge som kan fly er hindret så må nestemann lære seg å fly, helst uten å hindre sitt eget arbeid		

4.5 Økonomi

4.5.1 Budsjett

Vurdering	Alvorlighet: 2	Sannsynlighet: 3	Total risiko: 6
Risiko:	Ikke greie å holde oss innenfor budsjettrammene satt av bedriften (15000,- NOK).		
Forebyggende Tiltak	Sette opp budsjett og planlegge pengebruken skikkelig. Større utgifter godkjennes i plenum.		
Ved inntruffet hendelse:	Kommer man uansett over budsjett for å oppfylle A-krav så bør en forhandle med bedriften på nytt. Overskrides budsjettet for B- eller C-krav bør man vurdere å kutte disse kravene.		

4.5.2 Skade på andre personer og andres eiendom

Vurdering	**Alvorlighet: 4**	**Sannsynlighet: 1**	**Total risiko: 4**
Risiko:	Ved fri flygning kan man risikere at plattformen ødelegger andres eiendeler eller skader andre mennesker.		
Forebyggende tiltak	Sørge for at man er i rimelig avstand fra andre mennesker og eiendeler når man tester flygning. Finne ut hvem som står for forsikring/ansvar (skolen, oppdragsgiver, studenten), før vi utfører testflygning.		
Ved inntruffet hendelse:	Erstatte det som blir ødelagt.		

5 Inntrufne/ håndterte risiko

5.1 Leveranser

Flyktig tilgjengelighet og usikre leveringstider har preget flere av bestillingene vi har gjort. Vi har likevel planlagt og jobbet på en slik måte at dette ikke har hindret oss i prosjektarbeidet. Manglende utstyr kan selvsagt medføre forsinkelser, men ikke i den grad at noen i gruppa har stått uten oppgaver.

5.2 Defekter

Vi har opplevd defekter/inkompatibilitet på enkelte ikke-essensielle komponenter. Dette har blitt løst ved å benytte eget utstyr imens vi venter på leveranse. Konsekvensene har begrenset seg til bortkastede arbeidstimer (brukt på feilsøking), men i lavt antall.

5.3 Sykdom / frafall

Vi har hatt tilfeller av sykdom/sykemelding i gruppa, samt frafall. Dette har vi håndtert fortløpende på Sprint Planning og Daily Scrum møter. Konsekvensene har blitt minimale, og de tapte timene har med tiden også jevnet seg ut.

5.4 Avhengighet utenifra

Åpen kildekode gir alltid en viss risiko i forhold til hvor komplett funksjonaliteten til et system er. Det vil som regel avhenge av et community eller en frivillig gruppe mennesker som utvikler og jobber på systemet. Denne risikoen har inntruffet.

Etter å ha vært i kontakt med oppdragsgiver og endret prioritet på flere krav for å snevre inn funksjonaliteten vi skal levere, fikk vi dykket dypere ned i materien som omhandlet kravene med høyest prioritet. Det vi da fant ut, som mer eller mindre ikke var mulig å finne ut på et tidligere tidspunkt, var at store biter funksjonalitet ennå ikke var implementert.

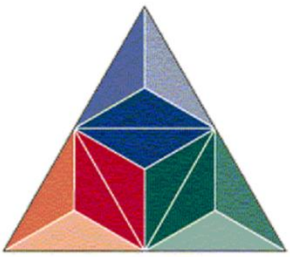
De som jobber aktivt for å utvikle programvare til autopiloten hadde ikke kommet like langt som vi var avhengig av at de hadde på dette tidspunktet. Denne jobben med å implementere autonom flyvning, er både for stor for oss å gjøre på den tiden som gjenstår, samt unødvendig for utviklingen videre. Siden autopilotens programvare stadig blir jobbet med, er det mer hensiktsmessig både for vår og produktets del at vi fokuserer på å utvikle den delen av systemet som ikke fokusert på av nettsamfunnet. Dette er den delen av produktet som oppdragsgiver hovedsaklig er interessert i, nemlig bilde og video som inneholder geografiske data og tidsmerking. Følgelig blir produktet som en helhet påvirket, men vi vil gjøre vårt beste for å legge alt til rette så systemet er klart for å ta i bruk delene som mangler på det tidspunktet de blir implementert.

5.5 Flygninger

Under en testflygning med kamera fikk den flygende plattformen en kraftig krasjlanding som resulterte i at senterplatene til hexa-konfigurasjonen, to propeller og to armer knakk. Dette er en veldig høy risiko ved utendørs flygning når man ikke har kontroll på alle elementer.

Dette har resultert i at vi fulgte opp forutsett risiko og har bygd «ned» fra seks til fire rotorer. Dette gjør at vi har mindre løftekraft og vil dermed si at man må stokke litt om på komponentene som skal opp i lufta.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Systemdesign
Dokumenttype:	Systemdokument
Dokument ID:	S001
Versjonsnummer:	V2.0
Versjonsdato:	23.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Figurliste

3 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	07.03.13	Første versjon
V2.0	23.05.13	Oppdatert figur

4 Introduksjon

Dette dokumentet viser hvilke komponenter som inngår i systemet, samt relevant informasjon om hver av komponentene og grensesnittene mellom dem. Med «systemdesign» mener vi design av det helhetlige systemet. Videre har vi valgt å dele inn design i to undergrupper; hardwaredesign og programvaredesign. Disse er mer omtalt i dokumentene *S005 – Programvaredesign*, *S007 – Fysisk design og S008 – Om utstyret.*

Valg av systemets komponenter og måten disse brukes på er basert på en rekke krav fremsatt av oppdragsgiver. Mer informasjon om disse kravene finnes *i* dokumentet *P003 – Kravspesifikasjon*.

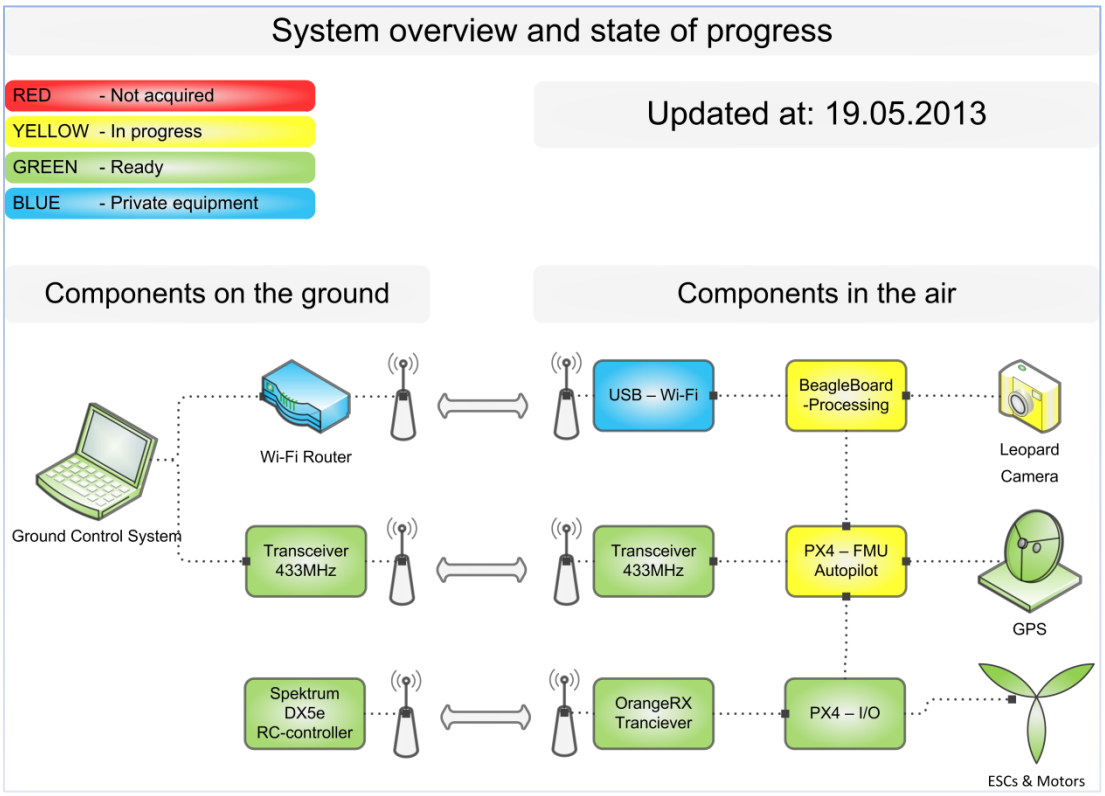
For omfattende informasjon om de ulike komponentene og deres egenskaper henviser vi til *S008 – Om Utstyret.*

5 Ordliste

AI	Artificial Intelligence
BEC	Battery Eliminator Circuit
CMOS	Complementary metal-oxide-semiconductor
FTDI	Future Technology Devices International, USB til Seriell konvertering
GPS	Global position system
RC	Radio control
RGB	Red-Green-Blue
Wi-Fi	Wireless Fidelity, trådløst nettverk

7 Det overordnede systemet

I Figur 1 ser vi oversikten over komponentene som danner systemet. Fargekodingen gir en indikasjon på fremdriften i prosjektet.



Figur 1 – Systemoversikt

7.1 Kommunikasjon mellom bakke og luft

Følgende krav er relevante i forhold til kommunikasjonskanalene vi har valgt:

- *K010*, OnTime id: 53
- *K011*, OnTime id: 54

Vi har planlagt tre separate kommunikasjonskanaler mellom bakkestasjonen og den flygende plattformen. Hver kanal har sitt eget formål.

7.1.1 Spektrum RC-link

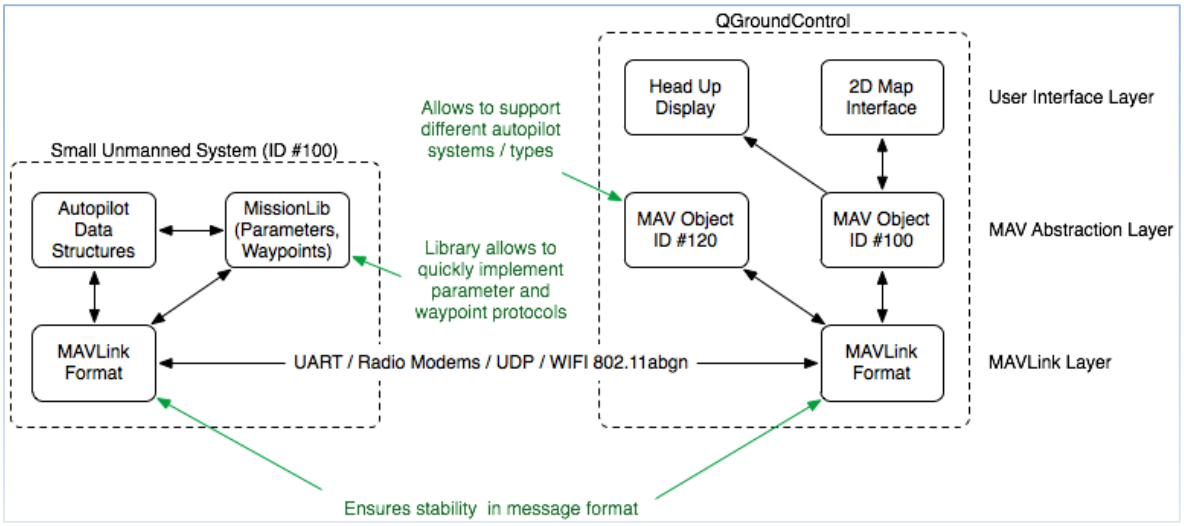
Den første og mest grunnleggende er kommunikasjon via RC-kontrolleren. Denne vil brukes til testflygning under utviklingen av systemet, men også som en «fail-safe» under bruk. Det bør være mulig å ta over kontrollen av systemet med RC-kontrolleren. Av sikkerhetshensyn skal RC-kontrolleren alltid være med når systemet brukes. De to enhetene, DX5e og OrangeRX R620, er fabrikkferdige enheter og det gjøres ingen utvikling på disse av prosjektgruppa. R620 er ikke mottakeren som ble levert med kontrolleren – det var Spektrum AR600. Dette fordi AR600 manglet noe kompatibilitet opp mot PX4.

7.1.2 433MHz Radiolink

Radiolinken skal benyttes for toveis seriell kommunikasjon mellom bakkestasjonen og autopiloten i PX4-FMU. Fra bakkestasjonen kan det sendes oppdatering/ endring av planlagt rute til autopiloten. Fra autopiloten sendes det posisjonsdata til bakkestasjonen.

7.1.2.1 Protokoller

Vi vil benytte MAVLink protokollen [1] som er spesielt utviklet for denne typen kommunikasjon. MAVLink støttes av blant annet QGroundControl [2] og APM [3], som er blant systemene som har blitt vurdert benyttet på bakkestasjonen. MAVLink er også støttet i PX4.



Figur 2 - MAVLink kommunikasjon[4]

7.1.3 Wi-Fi link

Wi-Fi linken opprettes mellom bakkestasjonen og bildebehandlingsenheten, BeagleBoard. Denne linken vil benyttes til å kommunisere bilder og video fra lufta til bakken.

8 Komponenter på bakken

Følgende krav var relevante i forhold til komponentene vi har valgte:

- *K003*, OnTime id: 39, 50, 115, 116 - Operasjonsplanlegging
- *K004*, OnTime id: 51 - Systembegrensningsvarsel
- *K010*, OnTime id: 53 - Endring av oppdrag
- *K011*, OnTime id: 54 - Dataoverføring
- *K012*, OnTime id: 34 - Live dataoverføring
- *K013*, OnTime id: 55 - Læringskurve

Utstyret vi bruker på bakken omtaler vi med begrepet «bakkestasjon». Dette inkluderer flere komponenter, detaljert under.

8.1 Ground Control Station

Bakkestasjonen inkluderer en bærbar pc som er levert av oppdragsgiver. Denne vil kjøre en modifisert utgave av QGroundControl [2]. Mer informasjon i dokumentet *S005 – Programvaredesign*.

8.2 Wi-Fi ruter

Bakkestasjonen må tilby det trådløse nettverket for overføring av bilde/video. Det er pr medio februar ikke planlagt noen integrert løsning for dette.

8.3 433MHz Transceiver

Kontrollsignaler (MAVLink-pakker) til den flygende plattformens autopilot kommuniseres over en egen radiolink. Dette gjelder både signaler fra QGroundControl.

8.4 RC-kontroller

Lovbestemte sikkerhetshensyn krever at en RC-kontroller kan benyttes til manuell styring, selv om farkosten er autonom. Vi har gått til innkjøp av en Spektrum DX5e som benyttes til dette formålet.

9 Komponenter i lufta

Følgende krav er relevante i forhold til komponentene vi har valgt:

- *K001*, OnTime id: 40 - Merking av bilder
- *K002*, OnTime id: 114 - Merking av video
- *K003*, OnTime id: 39, 50, 115, 116 - Operasjonsplanlegging
- *K010*, OnTime id: 53 - Endring av oppdrag
- *K006*, OnTime id: 38 - Bildelagring
- *K011*, OnTime id: 54 - Dataoverføring
- *K012*, OnTime id: 34 - Live dataoverføring

9.1 PX4 – Flight Management Unit

PX4FMU [5] er selve autopiloten i systemet. Vi benytter PX4-I/O til å tilpasse autopiloten til vårt bruksområde.

9.2 GPS

For å få korrekte koordinater og ha tilgang på korrekt klokke under flygning benyttes en GPS-brikke. Denne kommuniserer direkte med PX4FMU.

9.3 BeagleBoard

BeagleBoard er en utviklingsplattform som er i stand til å kjøre et Linux operativsystem. Denne vil vi benytte til all prosessering som ikke har sanntidskrav. BeagleBoard har tilkoblingsmuligheter for kamera og minne som vi vil benytte.

9.4 Kamera

Vi benytter en enkelt kameramodul til bilde, video, og videostrøm. Kameramodulen kontrolleres direkte av BeagleBoard.

9.5 OrangeRx R620 RC-Mottaker

RC-mottakeren vi benytter pares med senderen og gir muligheten for å ta over manuell kontroll av den flygende plattformen.

10 Hvordan passer dette sammen?

10.1 Autopilot

Autopiloten består av flere komponenter. I hovedsak er det «PX4 FMU» og «PX4 I/O» som i kombinasjon er selve autopiloten. Dette er en sannhet med modifikasjoner ettersom «PX4 FMU» er selve enheten som styrer og stabiliserer hexakopteret og «PX4 IO» er påkrevd ettersom denne står for mottak av RC kontroller signalet.

Ettersom kraften av en RC-modell som treffer et objekt kan bli katastrofal, spesielt hvis den treffer et menneske, er det kritisk at vi kan ta over dersom systemet skulle kommet ut av kontroll. Når vi skal ha autonom flygning er «systemet» avhengig av bakkestasjonen for å få instrukser om hvor den skal fly.

Det sendes da meldinger «MAVLink» mellom bakkestasjonen og «PX4 FMU» over radiolinken (433MHz radio) i et serialisert format. Dette gjøres ettersom kartsystemet ligger på PCen og ikke er kjent for autopiloten. Dersom vi skal fly ut av rekkevidde for radioen bør vi vurdere en «failover løsning» til for eksempel GSM-nettet. Det er også en mulighet å benytte en «onboard computer» for dette, for eksempel BeagleBoard.

Internt i PX4 benyttes et programmerings paradigme «Publish–subscribe pattern» [6]. Dette er implementert i form av et program som heter «uORB» (Object Request Broker) [7]. Dette er kjerneprinsippet som PX4 plattformen benytter for intern kommunikasjon. Som en kort introduksjon til ORB paradigmet kan man si at man får en tenkt «bus» der man kan sende og lese meldinger. Prosessene er ikke klar over hvem de kommuniserer med men snakker bare med «busen». Sensor data og lignende er publisert på denne «busen» og kan leses av alle som «abonnerer» på disse meldingene.

Ettersom systemet skal kunne ta bilder og video via kameraet som er tilkoblet BeagleBoard er en presis GPS og GPS-klokke meget viktig for kvaliteten på «produket» vi leverer. Vi benytter derfor 3DR sin GPS modul ettersom den har en meget høy oppdateringsrate (fem ganger i sekundet) og høy presisjon. Denne modulen støtter posisjoneringssystemene; «NAVSTAR-GPS» [8], «European Galileo» [9] og det russiske systemet «GLONASS» [10]. Ettersom vårt system er i bevegelse kommer det til å være en utfordring å sørge for at vår merking (geo-tagging og tidsmarkering) blir korrekt (tilfredsstillende). Ett aspekt som dukker opp i denne forbindelsen er prosjektering av koordinatene fra luften ned mot terrenget. Koordinatene til hexakopteret i kombinasjon med de tre orienteringsretningene (roll, pitch, yaw) byr på spennende geometriske beregninger for å kalkulere hvilket område av terrenget som har blitt fotografert.

Dette systemet har stort potensial når det kommer til funksjonalitet og den modulære oppbyggingen. Ettersom dette systemet er Open Source og Open Hardware byr det på muligheter for å tilpasse systemet ut i fra de kravene som stilles til systemet. Dette åpner dørene for mange spennende utvidelser innen områder som *computer vision*, kunstig intelligens og sensor forskning. Vi fokuserer på å implementere funksjonalitet som i utgangspunktet vil være forenklede løsninger på meget kompliserte problemer, der vi ser for oss at senere prosjektgrupper kan utvide løsningene til å inkludere mer sofistikerte AI-metoder for å løse problemene. Med dette tenker vi spesielt på metodikken for å fotografere områder. Slik vi har sett for oss problemet tenker vi en serie med POI (Point of interest) som indikerer posisjoner i luften der vi ønsker at bilde skal fotograferes fra. Denne

metodikken oppfordrer til å implementere AI-systemer der man heller kan tegne opp området i kartet man ønsker å få dekket med bilde/video og så la datamaskinen ta avgjørelser om hvordan dette området dekkes, da også innberegnet det mulige (lovlige/tillatte) luft området.

Feilmeldinger og statuskoder tenker vi å sende ut på «BlinkM» (RGB LED) og / eller et OLED-display. Dette kan være en god hjelp til en operatør ettersom ulike meldinger kan indikere om for eksempel: alle moduler er tilegnelig og kommunikasjonen mellom dem er ok (eller ikke), batteristatus og lignende.

10.2 Bildebehandling

For bildebehandling skal vi benytte «BeagleBoard xM». Valget av dette kortet er hovedsakelig på grunnlag av kortets støtte for kameramodulene til «Leopard Imaging». Det er flere nyere varianter på markedet, som «pandaboard» [11], men ved bestillingstidspunktet var det ingen kamera moduler som var tilegnelige for det kortet. BeagleBoard kommer til å håndtere kamera, CMOS modulen som sitter på kamera er fra «Aptina» og har modellnummer MT9P031 [12].

I henhold til NATO-standard skal bilder og video geo-tagges, altså merkes med geologiske lokasjonsdata. Om BeagleBoard også skal håndtere geo-tagging eller om det skal post-prosesseres er enda ikke bestemt. BeagleBoard kommer uansett til å ta seg av lagring av bilder og video til USB-tilkoblet flashminne. I tilfelle vi velger å postprosessere geo-taggingsjobben, kommer BeagleBoard trolig til å håndtere den eventuelle datastrukturen som da må lages for å bevare opplysningene om referansene til bilde og video.

Det kan også være at BeagleBoard kommer til å måtte kjøre «MAVProxy» [13] for å instruere flyruten til hexakopteret. PX4 og BeagleBoard kommer trolig til å ha en UART link mellom dem slik at det kan kommunisere serielt (dette kommer sannsynligvis til å være over en FTDI kabel, USB-til-serie).

BeagleBoard får power direkte fra 5V BEC, men vi vurder mulighetene for å benytte PX4 IO sine releer til å styre av/på av BeagleBoard. Dette kan resultere i et noe lavere totalt strømforbruk når oppstart av BeagleBoard ikke er nødvendig.

BeagleBoard har en kraftig DSP og er i helhet en kraftig ARM datamaskin. Dette åpner også dørene for å drive med *computer vision* algoritmer. Dette er ikke noe som prosjektet vårt dekker men som kan være spennende tanker å leke med i en eventuell videreføring av prosjektet. Som eksempler på mulige CV algoritmer kan vi nevne; objekt sporing/følging, bildestabilisering og «pattern recognition». Med objekt sporing, vil man kunne lokalisere bevegelige objekter i videoen, noe som kunne vært en meget spennende algoritme å implementere. Med bildestabilisering vil man være i stand til å levere video som har roligere bilde slik at brukeren får en mer oversiktlig video. «Pattern recognition» gjør at man har muligheten til å kunne kjenne igjen objekter og lignende, på denne måten kan man for eksempel klare å sikte seg inn mot en tidligere definert landingsplattform uten å kjenne dens posisjon.

10.3 Plattformen

Selve plattformen bygger på «DJI Innovations Flame Wheel F550» og er videre modifisert med understell og et «tårn» som holder komponentene vi benytter. Til rammen er det også festet en «buzzer».

På prototypen benyttet vi rustfritt stål for å lage tårnet, ettersom dette var lett tilgjengelig. I en senere utgave kommer tårnet (trolig) til å være laget av aluminium for vektbesparelse. For å montere sammen de ulike etasjene i konstruksjonen har vi benyttet noen lange skruer der etasjene er holdt på plass av låsemuttere. Disse skurene stikker langt opp over øverste etasje slik at komponentene som er montert der har noe beskyttelse dersom vi skulle velte. Dette designet er svært lite bestandig mot vær og vind og en senere versjon bør inkludere en hette (eller lokk) slik at de kritiske delene er mindre utsatt.

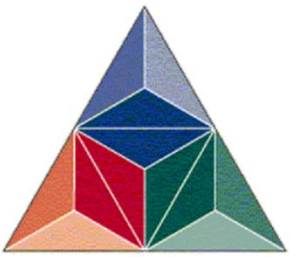
Understellet ble laget av aluminium fra starten av, men er fortsatt bare på prototyp-stadiet. Dette understellet er designet for å montere kameraet på, i tillegg til å fungere som landingsunderstell, for mer informasjon om dette se dokumentet: *S007 Fysisk design.* Armene til hexakopteret er laget av aluminium og selve senterplatene som holder armene er laget av en kombinasjon av glassfiber og metaller for armering av denne [14].

Batteriet får plass i mellom platene og festes på plass der på en slik måte at det er enkelt å ta inn og ut. Selve plattformen har vi utstyrt med et strømfordelingskort slik at det skal være enkelt å koble inn og ut moduler. Det ligger da montert en hovedtilførselsledning som batteriet kobles inn i. For mer informasjon om strømdistribusjon, se dokumentet: *S007 Fysisk design.*

11 Referanser

1. QGC. *Micro Air Vehicule Protocol*. 07.02.2013]; Available from: http://qgroundcontrol.org/mavlink/start.
2. QGroundControl. *QGC*. 11.02.2013]; Available from: http://qgroundcontrol.org/.
3. DIY. *APM - ArduPilotMega*. 11.02.2013]; Available from: http://code.google.com/p/ardupilot-mega/wiki/Mission.
4. QGroundControl. *QGC-MAVLink Architecture*. 06.03.2013]; Available from: http://qgroundcontrol.org/_detail/dev/qgroundcontrol-architecture.png?id=dev%3Amavlink_onboard_integration_tutorial.
5. Pixhawk. *PX4*. 26.02.2013]; Available from: http://pixhawk.ethz.ch/px4/.
6. Wikipedia. *Publish-subscribe Pattern*. 07.03.2013]; Available from: http://en.wikipedia.org/wiki/Publish–subscribe_pattern.
7. Pixhawk. *uORB*. 07.03.2013]; Available from: https://pixhawk.ethz.ch/px4/dev/shared_object_communication.
8. gov, U. *GPS*. 07.03.2013]; Available from: http://www.gps.gov/
9. Agency, G. *Galileo*. 07.03.2013]; Available from: http://www.gsa.europa.eu/.
10. Agency, S. *GLONASS*. 07.03.2013]; Available from: http://www.glonass-ianc.rsa.ru/en/.
11. Pandaboard. *pandaboard*. 06.03.2013]; Available from: http://pandaboard.org/.
12. Aptina. *MT9P031 Image Sensor*. 06.03.2013]; Available from: http://www.aptina.com/products/image_sensors/mt9p031i12stc/.
13. QGC. *MAVProxy*. 06.03.2013]; Available from: http://qgroundcontrol.org/mavlink/mavproxy_startpage.
14. DJI. *Flamewheel Features*. 06.03.2013]; Available from: http://www.dji-innovations.com/products/flame-wheel-multi-rotor/features/.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Programvaredesign
Dokumenttype:	Systemdokument
Dokument ID:	S005
Versjonsnummer:	V1.0
Versjonsdato:	07.03.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Figurliste

3 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	07.03.13	Første versjon

4 Introduksjon

Dette dokumentet, i tillegg til vedlegget, viser det planlagte designet av programvaren. Dokumentet tar for seg hva vi har gjort og hva vi ønsker å gjøre i forhold til design. For å visualisere designet bruker vi diagrammer og bilder, hvor alle relevante diagrammer ligger vedlagt i dokumentet; *vedlegg til S005*.

Det overordnede designet av systemet og kommunikasjonskanalene mellom komponenter er beskrevet i dokumentet *S001 – Systemdesign*.

5 Ordliste

Begrep	Beskrivelse
QGC	Qgroundcontrol, programvare for bakkestasjon
MAVLink	Micro air vehicle link, kommunikasjonsprotokoll
UML	Unified modelling language
PX4 FMU	Autopilot, PX4 flight management unit
VP	Visual Paradigm
MGRS	Military Grid Reference System

6 Unified Modelling Language(UML)

UML[1] er et visuelt modelleringsspråk som vi tar i bruk for å designe systemets programvare. UML bistår med nyttige diagrammer som gir en god visuell representasjon av systemets design, krav og samspill. Ved å bruke UML kan vi synliggjøre for oppdragsgiver hvordan vi planlegger å bygge systemet, samt skape en uniform oppfatning innad i gruppen om hvordan oppgaven skal løses.

I kapitlene nedenfor vil vi ta for oss verktøy vi har tatt i bruk og alle diagrammene vi har produsert for å oppnå en effektiv og ryddig designprosess. Alle designdiagrammene vi har produsert ligger i vedlegget.

6.1 Visual Paradigm

Visual Paradigm[2], heretter omtalt som VP, er et verktøy for UML-modellering som vi bruker for å produsere UML-diagrammer. Vi bruker VP for med en akademisk lisens, og alle diagrammer vi produserer vil derfor ha et vannmerke øverst i venstre hjørne.

6.2 Use case diagrammer

User stories i Scrum og *use cases* i UML er ganske så like. I både Scrum og UML brukes de for å beskrive krav/scenarioer brukeren vil at systemet takler. *Use case diagrammer* i UML er derfor viktige for å sikre at funksjonaliteten oppdragsgiver ønsker blir ivaretatt, og at systemet blir designet etter brukerens behov.

En *use case* har som oppgave å beskrive høynivå strukturell funksjonalitet i systemet, og viser til hvilke deler av systemet som kontrollerer forskjellig funksjonalitet.

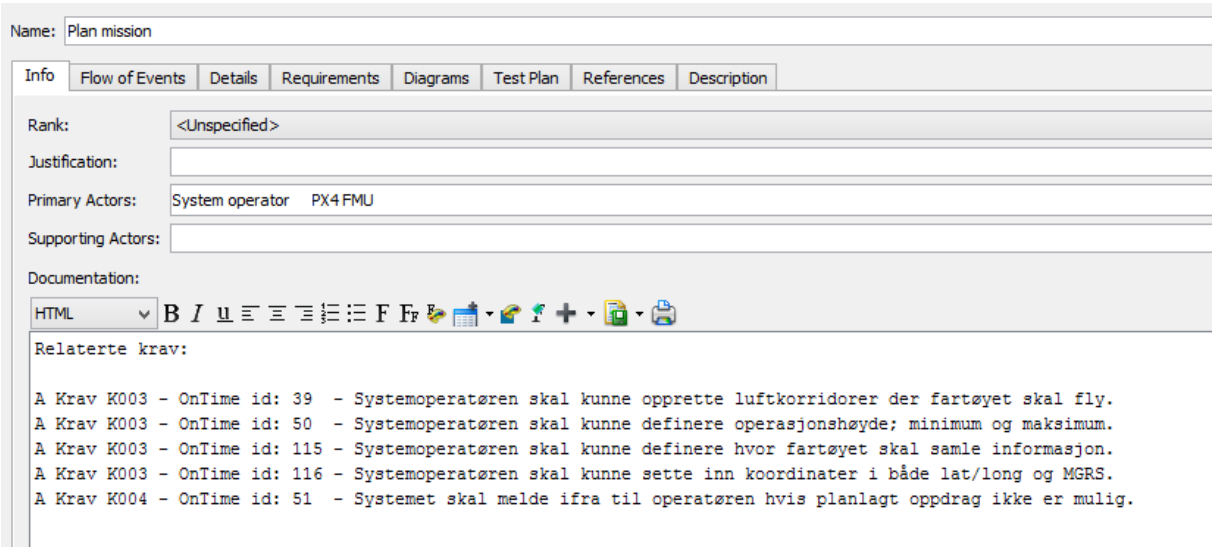
I *use case diagrammene* våre, som er vedlagt i eget dokument, relateres krav fra oppdragsgiver direkte til *use casene*. For å gjøre dette mer oversiktlig har vi to forskjellige farger avhengig av hvilken type krav det er snakk om; A krav er røde og B krav er grønne. I tillegg er kravene beskrevet i *use case description*.

Vi har valgt å dele opp systemet i to delsystemer, bakke og luft. Totalt har vi fire *use case diagrammer,* hvorav det første gir en oversikt over hele systemet og hvilke aktører som virker på det. Det andre beskriver hvilke aktører som virker på de to delsystemene. De to siste diagrammene går mer i detalj på hvilke *use cases* som angår de ulike aktørene i systemet. Hovedgrunnen til at disse er delt opp på denne måten er både for å få en bedre oversikt over hva det er som skal skje på hvilken plattform, og for å dekke alt av funksjonalitet i systemet.

En *use case* har flere attributter vi ønsker å bruke for gi et oversiktlig bilde over systemets ønskede funksjonalitet. Hvis vi f. eks tar en av *use casene* våre; «Plan mission», kan vi se at den har et beskrivende navn som forklarer hva slags funksjonalitet den representerer.

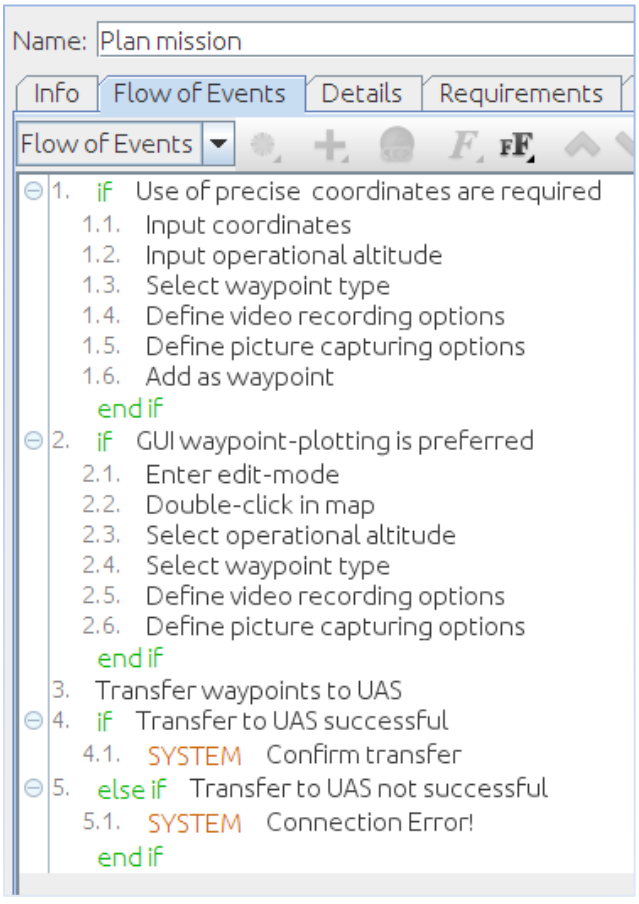
Figur 1 - Use case

Når vi ser nærmere på *use casen* vil den også ha oversikt over hvilke krav og *user stories* den relaterer til. Den forteller også om hvilke aktører som styrer den. Siden alle relaterte krav er «A krav» vil *use casen* ha rød farge.



Figur 2 - Use case spesification

Ved hjelp av *flow of events* kan vi utdype hvordan funksjonaliteten skal oppføre seg i praksis.



Figur 3 - Flow of events

6.3 Sekvensdiagrammer

Mens en *use case* bare tar for seg funksjonalitet på sitt enkleste format, vil sekvensdiagrammer ta for seg en mer detaljert flyt av funksjonaliteten til en *use case*. Sekvensdiagrammet beskriver interaksjonen og den sekvensielle logikken mellom moduler i systemet som skal oppnå ønsket funksjonalitet. Relevante sekvensdiagrammer ligger i vedlegget.

6.4 Aktivitetsdiagrammer

I et aktivitetsdiagram modellerer vi den naturlige flyten mellom aktiviteter for å illustrere den dynamiske strukturen til en eller flere funksjonaliteter i systemet. Et aktivitetsdiagram kan fungere som en utvidelse av *flow of events*. Vi har tatt i bruk disse diagrammene for å visualisere funksjonalitet som baserer seg på brukerinteraksjon, mens kommunikasjon mellom moduler i systemet er bedre illustrert ved et sekvensdiagram. Relevante aktivitetsdiagrammer ligger i vedlegget.

6.5 Deployment-diagrammer

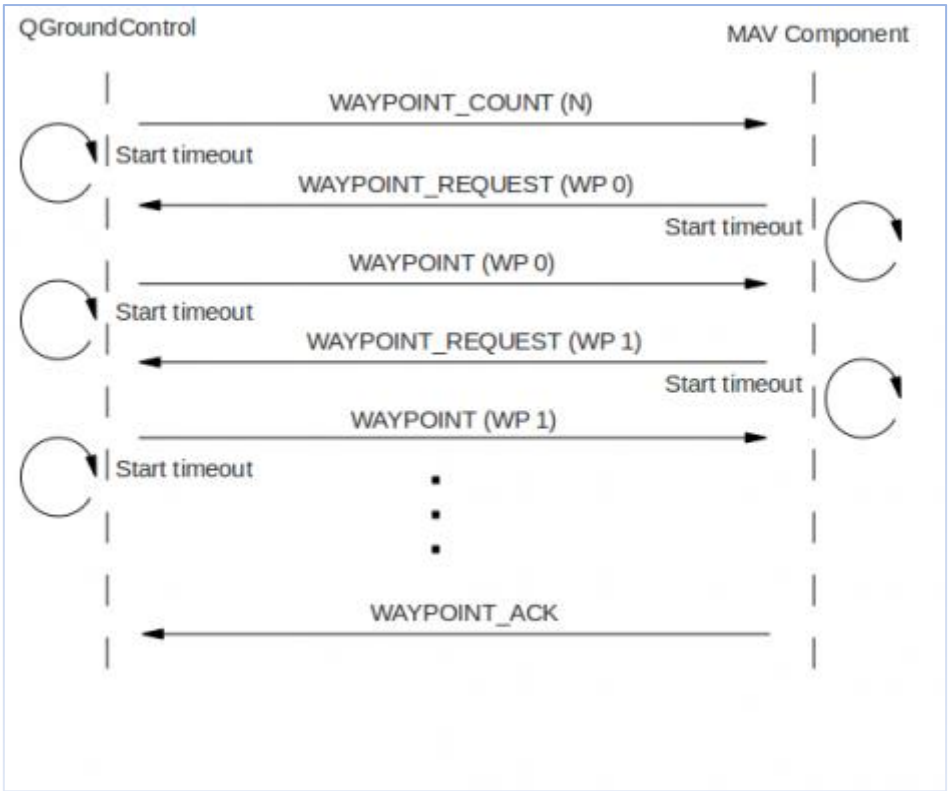
Vi bruker *deployment-diagrammer* for å visualisere hvordan komponenter og programvare i systemet fungerer sammen. Diagrammet tilbyr en oversiktlig måte å vise til hvordan systemets noder vil kommunisere sammen og hvor programvare blir implementert i form av *artifacts* på noder. Relevante *deployment-diagrammer* ligger i vedlegget.

7 MAVLink

MAVLink[3] er en «*Micro air vehicle communication protocol*», altså en protokoll for kommunikasjon mellom bakkestasjon og flygende plattform. MAVLink er grundig testet med PX4 og QGroundControl. Programvaren vi designer og skriver vil derfor implementere MAVLink som kommunikasjonsprotokoll.

7.1 Waypoint list

Overføring av *waypoints* er en sentral del av systemet, og er viktig i forhold til hvordan systemet blir designet. *Waypoint* protokollen er allerede implementert i QGC og PX4 FMU, og er en sekvensiell overføringsprosess av *waypoints* mellom to enheter som allerede implementerer MAVLink. I figuren under kan man se hvordan overføringsprosessen fungerer i praksis.



Figur 4 - *Waypoint protocol* [3]

7.2 Waypoint filformat

Under er en figur over den foreslåtte strukturen til et waypoint.

```
QGC WPL <VERSION>

<INDEX> <CURRENT WP> <COORD FRAME> <COMMAND> <PARAM1> <PARAM2> <PARAM3>
<PARAM4> <PARAM5/X/LONGITUDE> <PARAM6/Y/LATITUDE> <PARAM7/Z/ALTITUDE>
<AUTOCONTINUE>
```

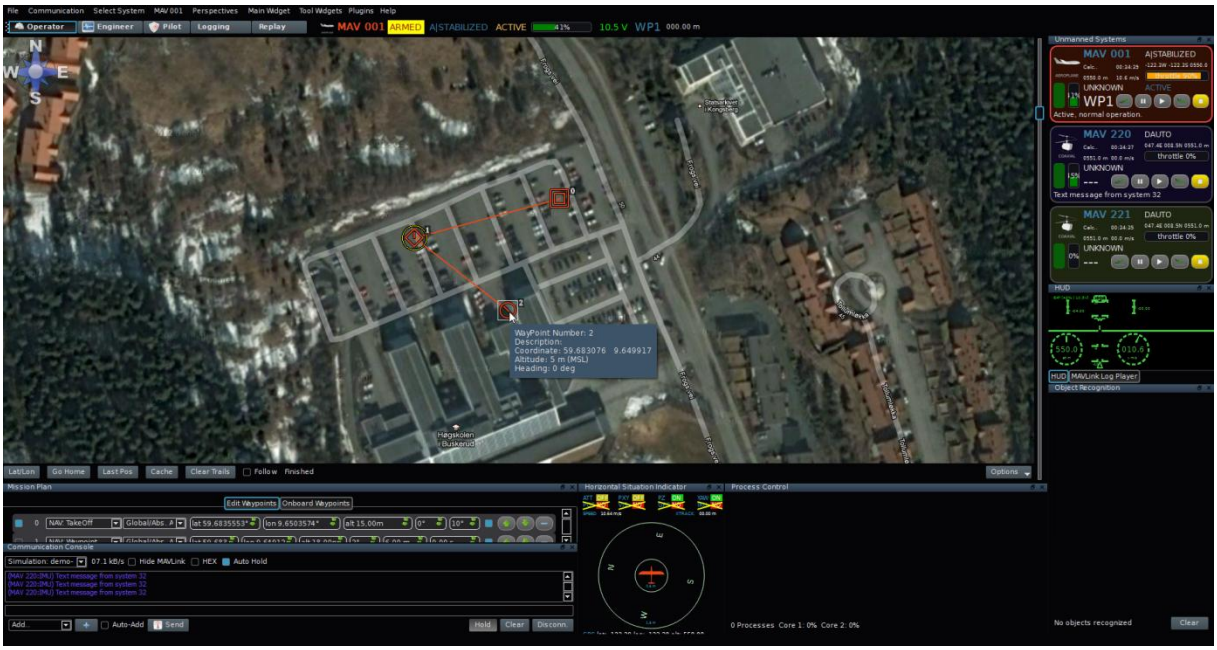
Som vi ser av strukturen over behandles posisjonen til et *waypoint* i lat/long format. For programvare- og GUI-design betyr dette at vi ikke kommer til å sende *waypoints* som MGRS om brukeren ønsker å benytte dette koordinatsystemet. Dette er en konvertering vi gjør før dataene blir lagret som *waypoints*.

8 Graphical User Interface(GUI)

Vi skal benytte oss av et brukergrensesnitt som allerede eksisterer, QGroundControl. Vår oppgave når det kommer til denne biten blir derfor å implementere den funksjonaliteten som ikke finnes allerede. Grensesnittet vi ser for oss å bruke har åpen kildekode og en modulær oppbygning, derfor vil de kravene vi har fått for ekstra funksjonalitet bli implementert som moduler i et allerede eksisterende rammeverk.

Vi vil her gi en oversikt over den funksjonaliteten som bør inkluderes. Spesifikke beslutninger angående design er noe vi ønsker å gjøre så sent som mulig. Her vil det derfor være punkter der vi spekulerer i hva som kan være den beste eller mest naturlige måten å løse et problem på, men endelige beslutninger vil mest sannsynlig ikke bli tatt før på implementasjonsstadiet.

8.1 QGroundControl (QGC)

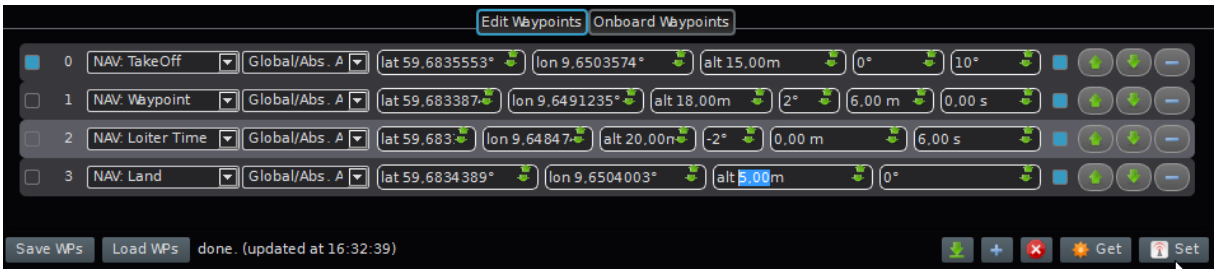


Figur 5 - QGroundControl *User Interface*

Vi har vurdert noen ulike programmer å basere bakkestasjonen vår på. QGroundControl fremstår som det beste valget for oss og programvaredesignet vårt reflekterer dette. En del av funksjonaliteten vi trenger er allerede dekket i QGC, men krever i de fleste tilfeller utvidelser eller modifikasjoner.

QGC har implementert mulighet for å opprette *waypoints* som den flygende plattformen kan følge. Disse blir satt ved å dobbeltklikke i kartet, hvorpå man får opp dataene i en dialog der man kan endre dem.

Denne dialogen gir brukeren muligheten til å bestemme handling i et *waypoint*, koordinater i lat/long, *altitude*, *attitude*, og også noen ulike alternativer som vil avhenge av hva slags handling man har valgt. *Waypoints* lastes opp til den flygende plattformen ved å trykke på en knapp. Brukeren har også anledning til å laste ned *waypoints* fra tilkoblet flygende plattform.



Figur 6 - QGC *Waypoint* redigering

8.2 Krav relevant for GUI

Kravene som er relevant i forbindelse med planlegging av GUI er:

- A krav K003 – OnTime id: 39, 50, 115, 116 – Operasjonsplanlegging.
- A krav K004 – OnTime id: 51 – Systembegrensningsvarsel.
- B krav K010 – OnTime id: 53 – Endring av oppdrag.
- B krav K011 – OnTime id: 54 – Dataoverføring.
- B krav K012 – OnTime id: 34 – Live dataoverføring.
- B krav K013 – OnTime id: 55 – Læringskurve.

8.2.1 K003 – Operasjonsplanlegging

Kravet fra oppdragsgiver, som går på hvordan operasjonsplanlegging skal foregå, er delt opp i flere mindre krav. Kravene omfatter:

- Opprette luftkorridorer der fartøyet skal fly
- Definere operasjonshøyde; minimum og maksimum
- Definere hvor fartøyet skal samle informasjon
- Sette inn koordinater i både lat/long og MGRS

Kravet sier at det skal være mulig å opprette luftkorridorer. Dette tolkes som en boks med tillatt luftrom fartøyet kan bevege seg i. Denne boksen vil ha en bredde i tillegg til minimum og maksimum høyde over bakken. Det er ingen støtte for dette i det eksisterende brukergrensesnittet, og er noe vi må implementere selv.

Når vi skal definere hvor fartøyet skal samle informasjon, bør dette skje samtidig som man definerer et *waypoint*. *Waypoint*-funksjonalitet og protokoller, som sagt tidligere, er implementert fra før. *Waypoint*-filformatet inneholder veldig mange forskjellige parametere, det vil da kanskje være lurt å benytte disse parameterne for å sette noen flagg for å definere hvor fartøyet skal samle informasjon. MAV-komponenten som mottar *waypoints* vil også returnere en melding når et *waypoint* blir nådd. Denne responsen kan være nyttig informasjon for systemet i forbindelse med «innsamling» av data.

Kravet definerer også at det skal være muligheter for å legge inn koordinater på MGRS format. Dette er noe vi må implementere i brukergrensesnittet selv. Vi ser for oss at en modul for å oversette

koordinater fra det ene formatet til det andre vil være tilstrekkelig. Dette gjør at brukeren av systemet selv kan velge hvilket format koordinatene skal settes inn på.

8.2.2 K004 – Systembegrensningsvarsel

Et systembegrensningsvarsel vil komme hvis fartøyet ikke kan gjennomføre den planlagte ruten. Denne beregningen vil være en kontroll som skjer under planlegging, og før endelig opplasting av oppdrag til den flygende plattformen blir gjort. Følgende punkter bør være med:

- Batterikapasitet i forhold til lengde på ruten.
- Operasjonshøyde må ikke overskride hexakopterets kapasitet
- Operasjonshøyde må ikke være under bakkenivå
- Billedtakning og videoopptak i forhold til kameraets kapasitet

Denne funksjonen, varsel om systembegrensninger, eksisterer ikke i QGC og må derfor implementeres av oss. QGC kan derimot hente batteristatus fra den flygende plattformen når den er koblet til. Vi må derfor implementere en advarsel som lar brukeren vite når han har planlagt et oppdrag som går ut over nåværende batterilevetid.

QGC har som sagt funksjonalitet for å definere høyde. Vi må implementere funksjonalitet for å definere tillatt maksimum og minimum høyde. I tillegg må vi presentere for brukeren at han har satt inn en høyde som er utenfor den flygende plattformens kapasitet.

Om brukeren forsøker å planlegge et scenario som bryter med kapasiteten til kameraet som er montert på den flygende plattformen må han få en advarsel om dette. Eksempel på dette kan være begrensninger i lagringskapasitet eller umulige kombinasjoner av bilde og video, typisk på samme tid.

Det kan være naturlig at disse advarslene kommer som et forgrunnsvindu. Det viktige er at det presenteres på en slik måte at brukeren ikke kan unngå å få med seg informasjonen, slik at umulige oppdrag ikke lar seg iverksette.

8.2.3 K010 – Endring av oppdrag

Med endring av oppdrag forstår vi følgende:

- Endre planlagt rute
- Endre når og hvor det skal filmes eller tas bilder

Nåværende *waypoint*-system i QGS tillater brukeren å legge til og fjerne *waypoints* som han selv vil, for så å laste opp disse til den flygende plattformen.

Siden eksisterende system ikke har funksjonalitet for å planlegge opptak av video og bilder, har det heller ikke funksjonalitet for å endre dette. Når vi får dette inn som en del av *waypoint*-planleggingen vil det også være naturlig å implementere muligheter for å endre det.

8.2.4 K011 – Dataoverføring

Med «dataoverføring» forstår vi i dette tilfellet; nedlastning av bilder og video som er lagret på den flygende plattformen. Brukeren bør kunne:

- Koble opp Wi-Fi link
- Velge filer for nedlastning
- Overvåke progresjon.

QGC kobler seg opp til den flygende plattformen over radiolinken for styring av plattformen. Dataoverføringen ønsker vi derimot å gjøre over en Wi-Fi forbindelse til *BeagleBoard*. Her trenger vi derfor å implementere GUI-elementer for tilkobling av Wi-Fi og nedlastning av data. Det kan også være nyttig med en status på hvor mye som gjenstår av nedlastningen. Før en laster ned kunne det også være nyttig med en dialog der man kan se over hvilke filer som ligger på den flygende plattformen og merke av de filene man ønsker å laste ned.

8.2.5 K012 – Live dataoverføring

Med «live dataoverføring» forstår vi en direktesending og visning av video fra kameraet som er montert på den flygende plattformen. Dette krever følgende:

- En måte å starte *streaming* på
- Status på Wi-Fi link
- Visning av video

QGC har allerede støtte for visning av inntil to videostrømmer. Det må allikevel implementeres funksjonalitet for å koble til Wi-Fi og vise status på denne.

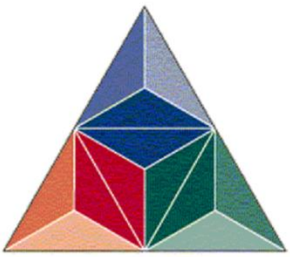
8.2.6 K013 – Læringskurve

Det er stilt et krav om at en bruker skal kunne lære seg å operere systemet på under 60 minutter. QGC, slik som den fremstår umodifisert, er et komplisert program som tar lang tid å lære seg dersom man ikke allerede er godt kjent med denne typen programvare (altså Ground Control Station systemer). Resultatet er at vi bør forsøke å eliminere ubrukte elementer fra det grafiske grensesnittet, slik at en bruker ikke har for mye ukjent å forholde seg til.

9 Referanser

1. OMG. *UML*. 07.03.2013]; Available from: http://www.uml.org/.
2. Paradigm, V. *VP for UML*. 07.03.2013]; Available from: http://www.visual-paradigm.com/.
3. Qgroundcontrol. *mavlink*. 03.03.2013]; Available from: http://qgroundcontrol.org/mavlink/waypoint_protocol.

KONGSBERG

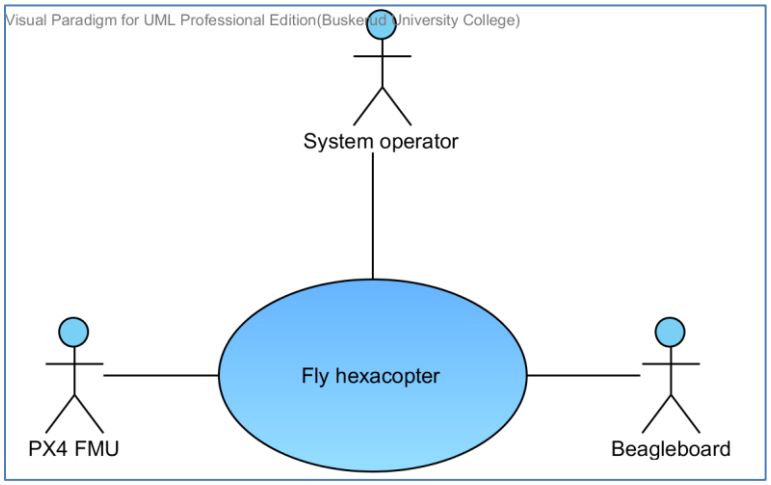


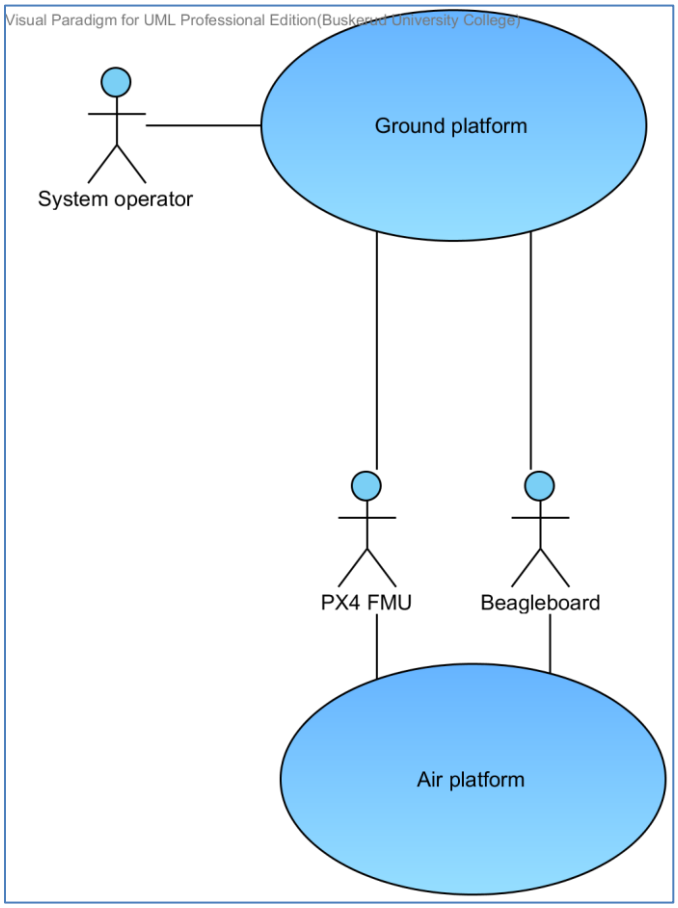
HØGSKOLEN
i Buskerud

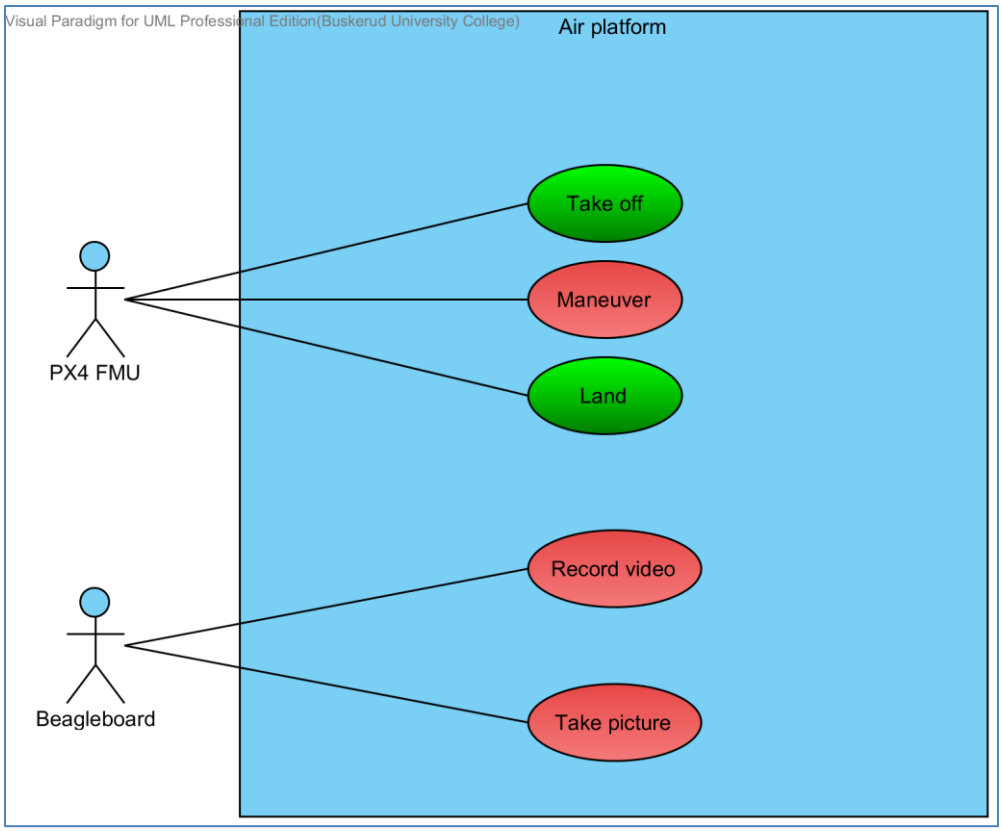
S.T.A.R.
System for Tactical Aerial Reconnaissance

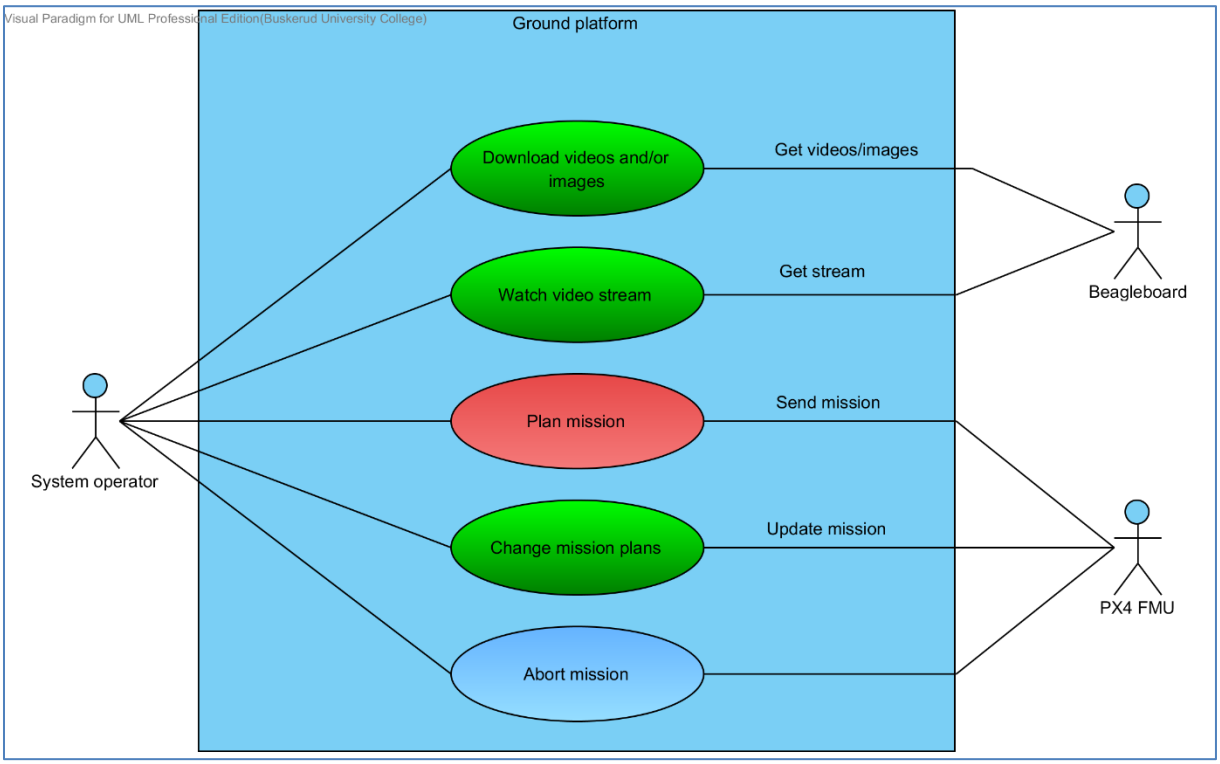
Vedlegg til S005 - Programvaredesign

1 Use case diagrammer

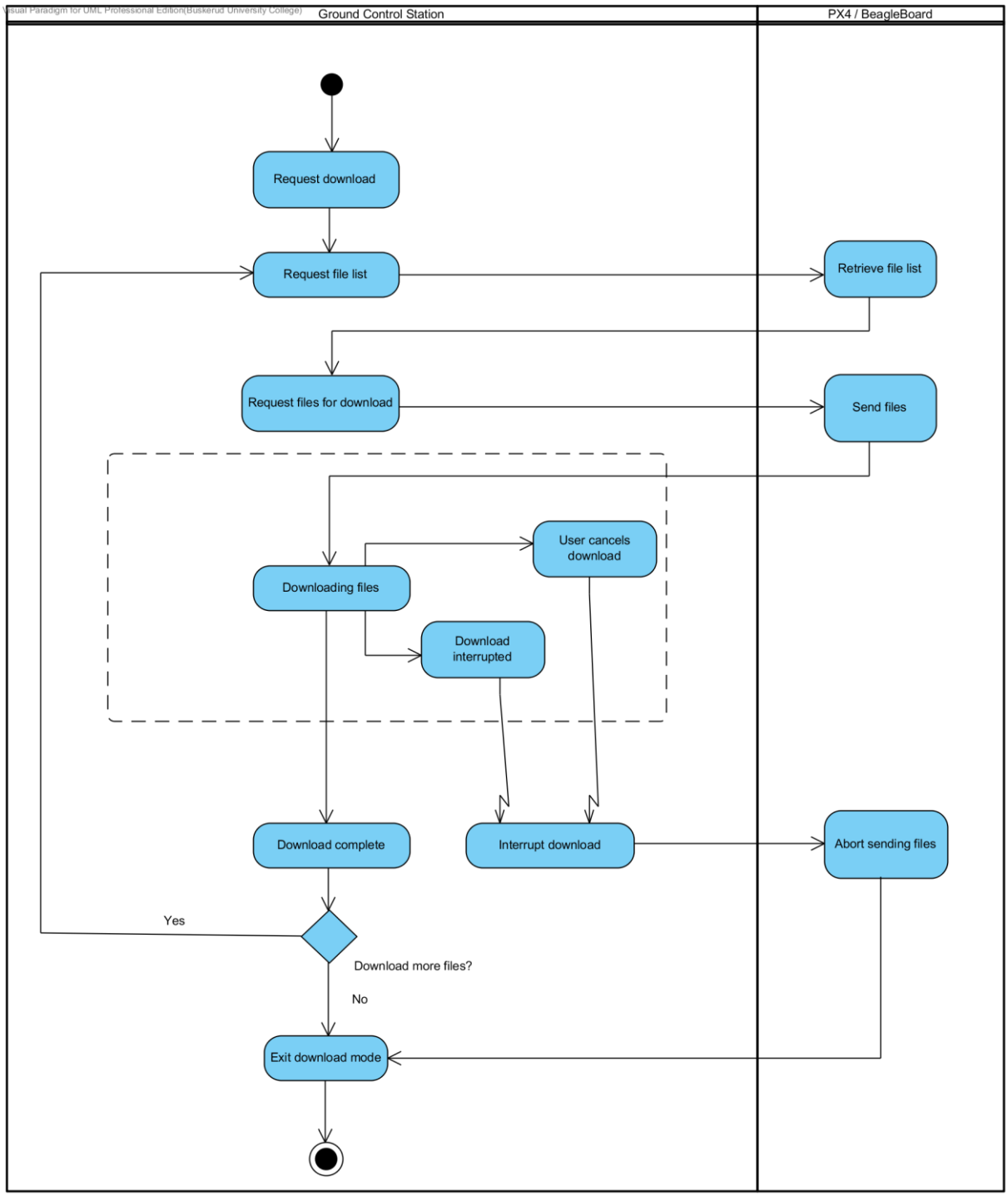


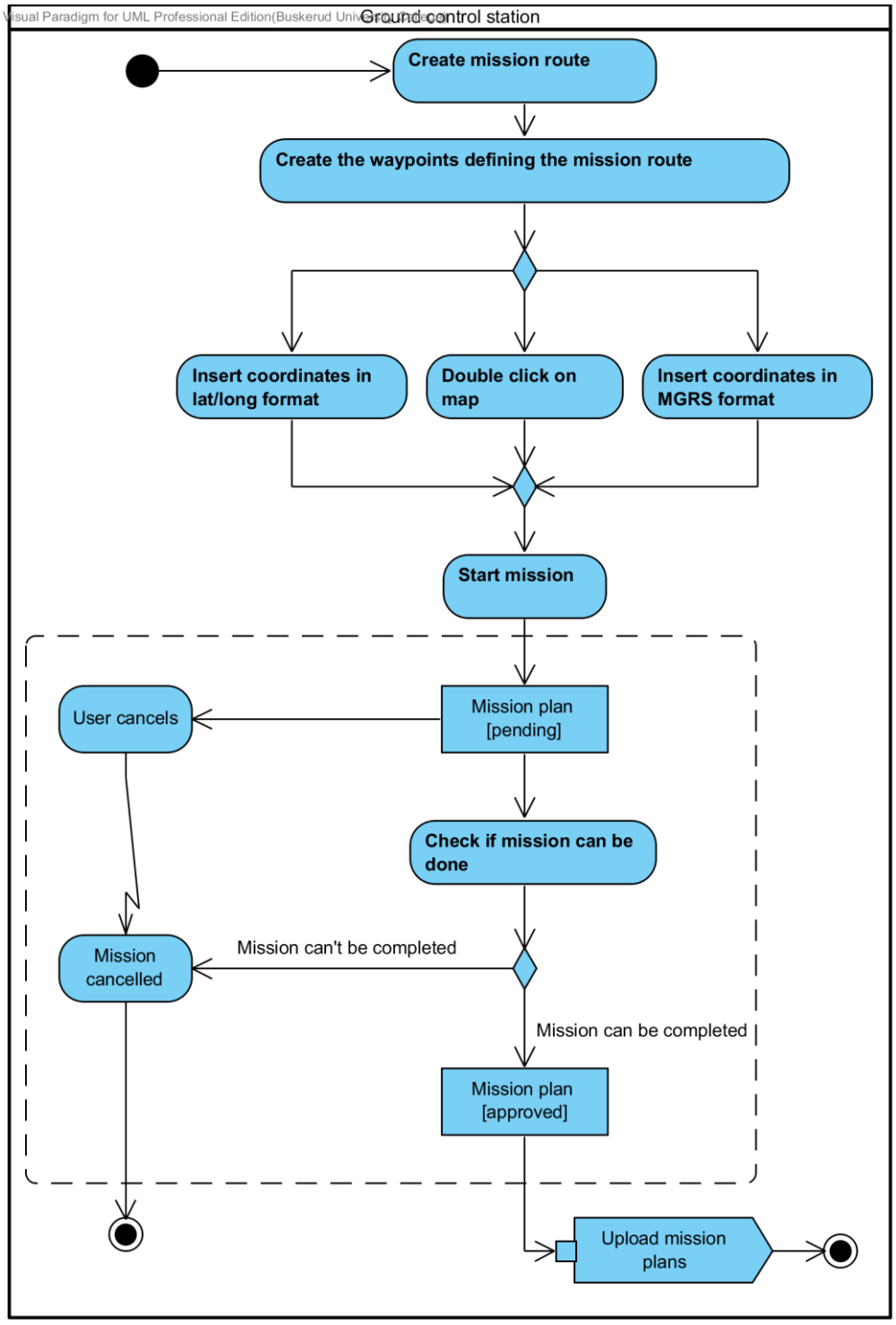



Visual Paradigm for UML Professional Edition(Buskerud University College)
Air platform
Take off
Maneuver
Land
PX4 FMU
Record video
Take picture
Beagleboard

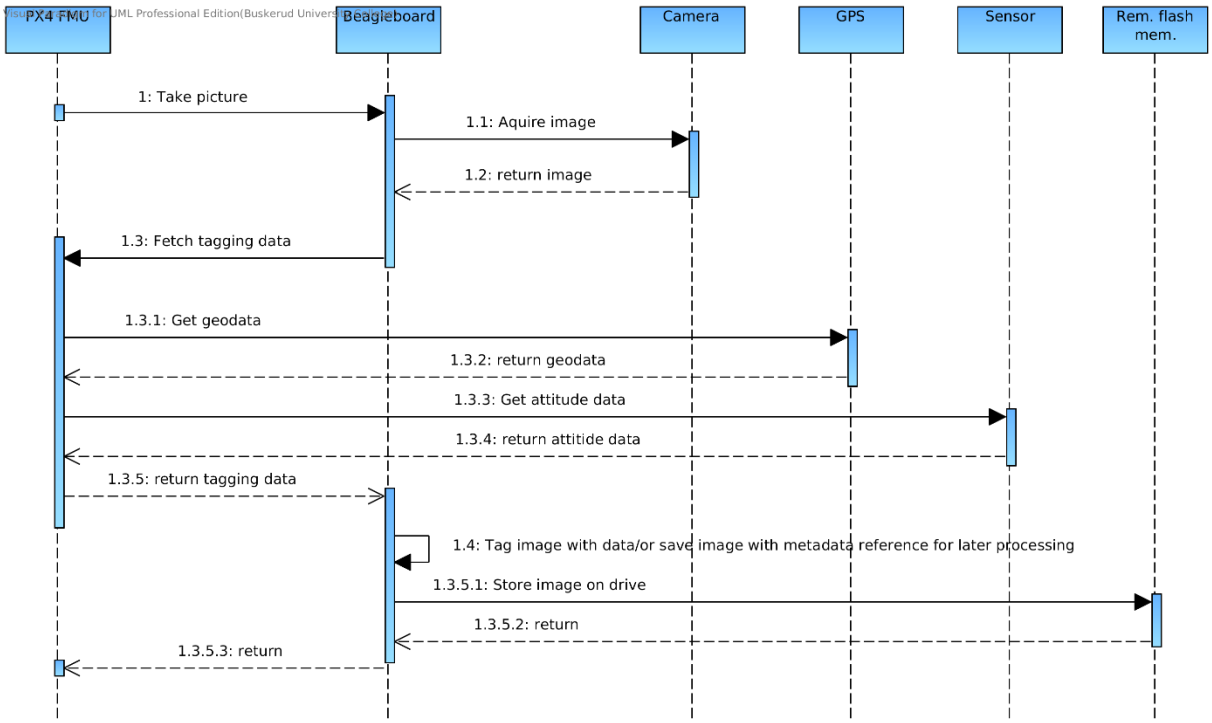

Visual Paradigm for UML Professional Edition(Buskerud University College)
Ground platform
Download videos and/or images
Get videos/images
Watch video stream
Get stream
Beagleboard
Plan mission
Send mission
System operator
Change mission plans
Update mission
PX4 FMU
Abort mission

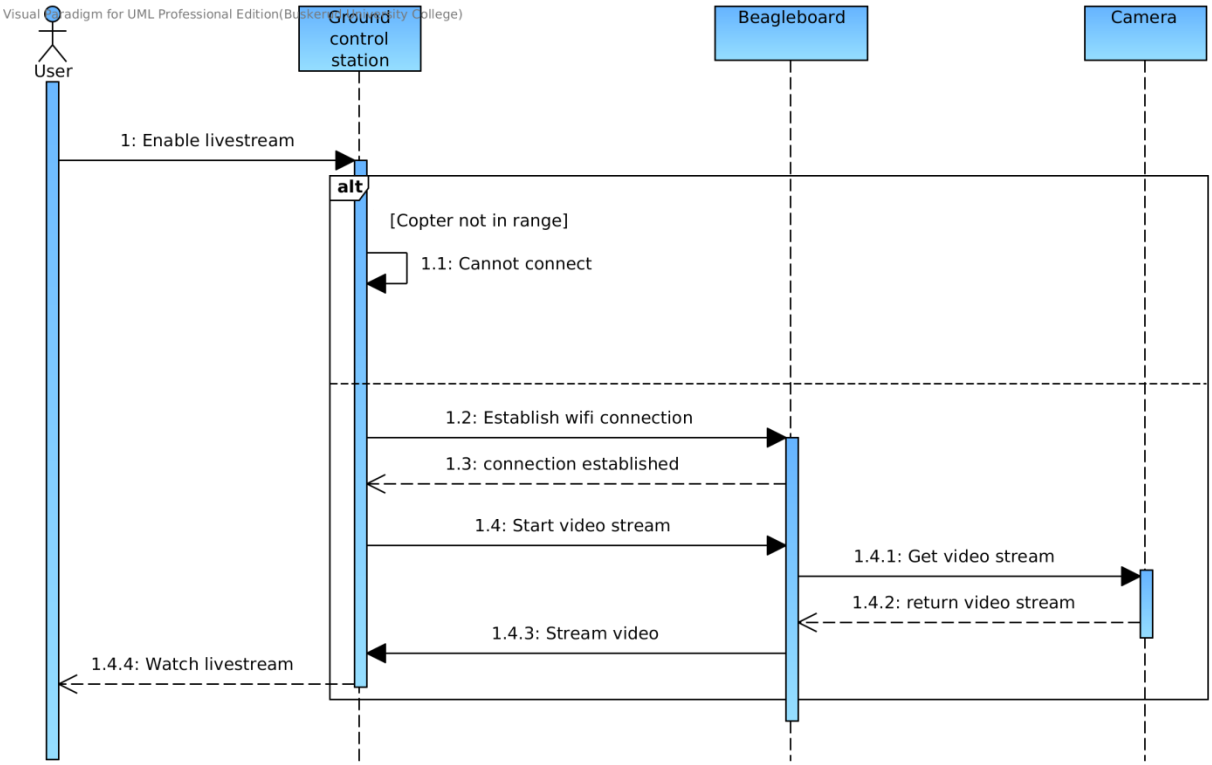
2 Aktivitetsdiagrammer



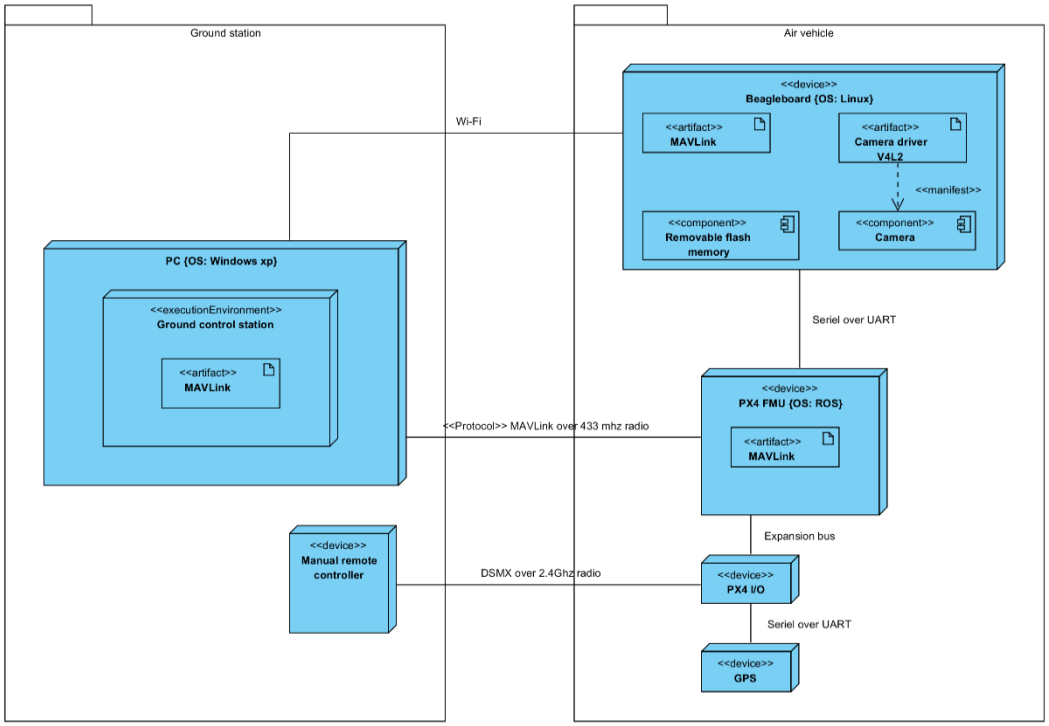

Visual Paradigm for UML Professional Edition(Buskerud University College)
Ground control station
Create mission route
Create the waypoints defining the mission route
Insert coordinates in lat/long format
Double click on map
Insert coordinates in MGRS format
Start mission
User cancels
Mission plan [pending]
Check if mission can be done
Mission cancelled
Mission can't be completed
Mission can be completed
Mission plan [approved]
Upload mission plans

3 Sekvensdiagrammer

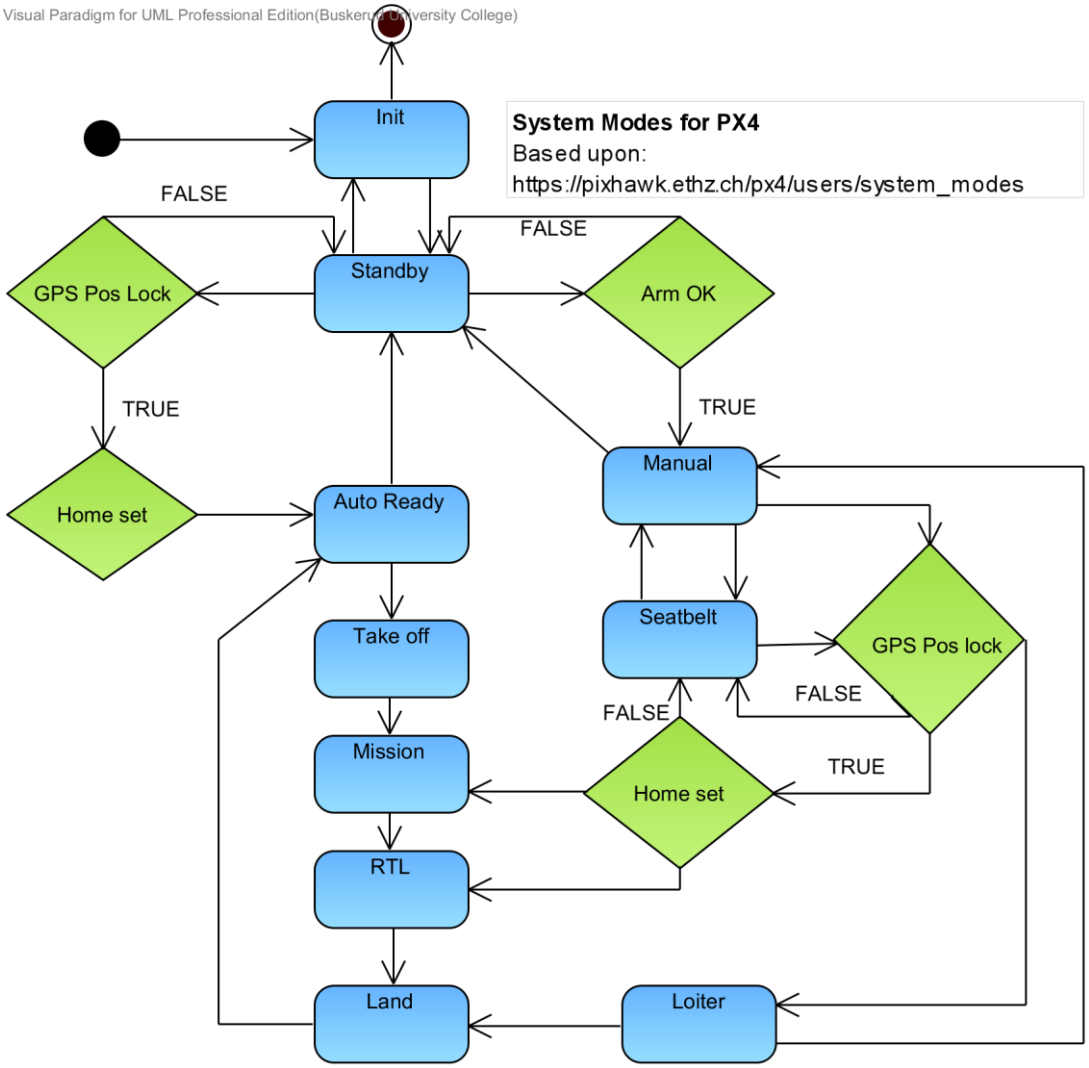




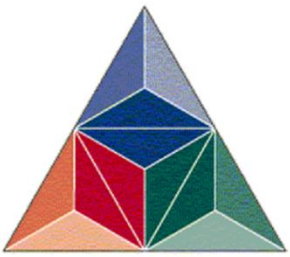
4 Deployment-diagrammer



5 Tilstandsdiagram



KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Fysisk design
Dokumenttype:	Systemdokument
Dokument ID:	S007
Versjonsnummer:	V2.0
Versjonsdato:	22.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Figurliste

3 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	06.03.13	Første versjon
V2.0	22.05.13	Endringer i prototyp

4 Introduksjon

Dette dokumentet gir en innføring i hvordan det fysiske oppsettet av kretskort, batteri, fordeling, landingsmekanisme og styring vil se ut på den flygende plattformen. Enkelte deler av systemet er ferdig designet og skal bare monteres og kobles opp. Andre deler må designes og konstrueres helt fra råmaterialer.

5 Ordliste

ESC	Electronic Speed Controller
BEC	Battery Elimination Circut
I/O	Input / Output
FMU	Flight Management Unit
ARF	Almost Ready to Fly
SW	Solid Works
OPTO ESC	Opto-koblet mellom signalet og ESC
BEC	Battery eliminator circuit
PWM	Puls bredde modulasjon (pulse width modulation)
DC	Direct current

6 Fysisk oppsett

Hvordan komponentene arrangeres på hexakopteret har mye å si for hvor håndterbart, oversiktlig og servicevennlig systemet vil bli. Enkelte komponenter vil måtte få en helt fast plass grunnet størrelse, masse eller andre hensyn.

Hele systemet er designet på prinsippet om å være modulært. Enkeltmoduler skal kunne lett monteres fra den flygende plattformen uten at hele systemet må demonteres.

Det elektriske systemet på prototypen er også designet slik at alle komponentene lett kan kobles ut uten bruk av loddebolt eller annet utstyr.

Enkelte festeplater til ulike deler av den flygende plattformen samt landingsbein blir designet og kappet til fra råmaterialer. Materialvalg for disse komponentene er viktig. Ved valg av for tunge materialer vil dette påvirke total masse til den flygende plattformen, dette vil også koste mer energi jf. Krav: *K005 - Flygetid*. Lette og sterke materialer er ofte dyre, noe som også må tas hensyn til, jf. Krav: *K008- Budsjett.* Aluminium kombinerer styrke, letthet, tilgjengelighet og pris veldig godt.

En risiko vurdert i dokumentet *P005 – Risikoanalyse* nevner at ved uhell så bør den flygende plattformen kunne bygges ned fra et hexakopter til et quadkopter. Design av festeplatene er gjort slik at dette fortsatt er en mulighet.

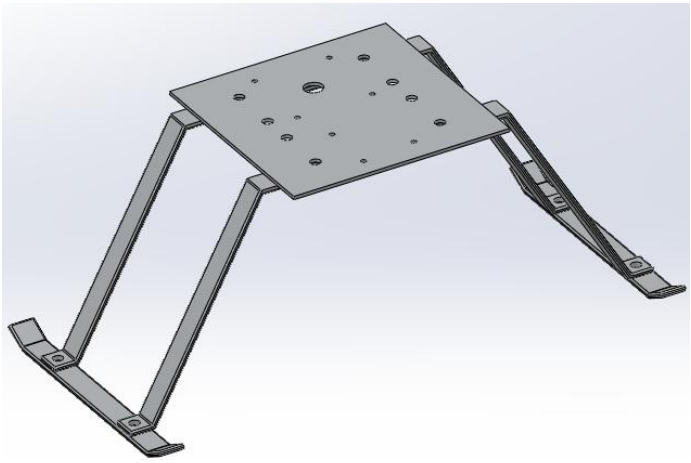
6.1 Tårn

Kretskortene: BeagleBoard, PX4 FMU + I/O, GPS og RC mottaker vil stå i et tårn på toppen av hexakopteret. Strømfordelingskort og spenningsregulator kan også stå her, endelig avgjørelse om plassering tas i implementasjonsfasen. Ved å samle og gjøre tårnet mest mulig kompakt, så øker vi også stabiliteten. Selve tårnet vil stå i to etasjer opp fra topplaten til hexakopteret. BeagleBoard legges nederst fordi det er mye større i areal enn de andre kortene. Denne «etasjen» kan ha mulighet for spenningsregulator og strømfordelingskort.

Som en beskyttelseskappe over tårnet bør det være en klar og forholdsvis fast struktur. Dette gir godt innsyn visuelt og for all kommunikasjon inn og ut av den flygende plattformen. Denne strukturen bør være lett men sterk nok til at den beskytter tårnet mot værforhold og fysisk skade.

6.2 Underside

Bildene vil tas med et fugleperspektiv, derfor må kamera enten festes på undersiden, eller plasseres med forlenger ut fra tårnet så sikten under ikke blir hindret. Den siste løsningen kan fort bli ustabil og medføre større risiko for skade på kamera. Dette gjør at det å feste kamera på undersiden er mer logisk. Dette medfører da at den flygende plattformen må heves opp fra bakken. Dette har vi valgt å legge til rette for ved å montere landingsbein på undersiden. Kamera kan stå i samme festeplate. Strømfordelingskort kan også sitte her, endelig avgjørelse tas i implementasjonsfasen.



Figur 1 - Sammenstilling av understell

Landingsbeina er konstruert ved å bøye til aluminium som settes på skinner. Beina skrus til en plate på undersiden. Som en del av designprosessen er det utført en enkel statisk styrkeberegning. Noe av designet og styrkeberegningen vises her i rapporten, de fullstendige rapportene ligger vedlagt. Dette arbeidet er utført i SolidWorks[1].

Dette er viktig med tanke på vekt og landing. Designet vil da få noen minimumskrav, avhengig av hva materialet tåler av stress. Ettersom vi ikke har kjennskap til materialets legering kan vi ikke trekke noen nøyaktig konklusjon basert på styrkeberegningen. Den statiske flaten i simuleringen er satt til å være underflaten på skia (kontaktflaten mot bakken), noe som blir urealistisk i forhold til den virkelige verden hvor beina ville sklidd ut mot sidene før noen annen form for stress ville blitt påført konstruksjonen.

Ettersom vi har tilgang på 3 mm aluminium, er dette et naturlig valg av materiale for understellet. Dette vil kappes fra en plate. Deretter bøyes, kappes og borres til som spesifikasjonene i vedlagte tegninger beskriver.

6.3 Hovedkropp

Hovedkroppen med topp- og bunnplate til hexakopteret er et ferdig sett og designes ikke av gruppa. Dette er et «ARF kit» fra «DJI Innovations»[2]. ESC til hver enkelt motor blir festet med strips fast til hver respektiv arm.

Batteriet plasseres midt inni hexakopteret. Ved for høy eller lav plassering vil batteriet påvirke stabiliteten og ytelsen til hexakopteret. Selv om plattformen greier å kompensere for ustabiliteten, så vil det koste mer energi å holde denne stabiliteten jf. *K005- Flygetid*.

En «buzzer» for lydsignaler kan også plasseres på selve hovedkroppen.

6.4 Quad-konfigurasjon

Som et alternativt design til den flygende plattformen vil være å bruke kun fire rotorer eller et quad-kopter. Siden denne konfigurasjonen har mindre overflateareal til å plassere komponentene på, så må plasseringen bli annerledes.

6.4.1 Topp

I stede for et tårn, så vil autopiloten og GPS sitte skrudd fast i topplaten til hovedkroppen. Radio, 5V, og RC-mottaker har nå blitt satt ut på armene, helt inn mot senter.

6.4.2 Senter

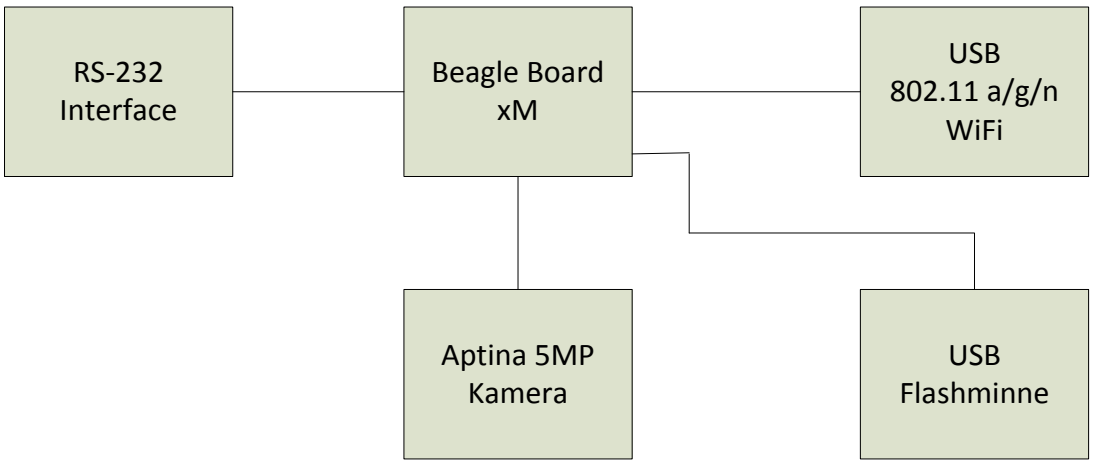
Bildebehandlingsenheten kan plasseres i midten med nok ikke-ledende dempemateriale som holder denne komponenten på plass. Strømfordeling og kamera plasseres på to utstikkere fra underdelen av senterplata. Disse stikker ut foran og bak. Kamera har fått plass på baksiden

6.4.3 Under

En spesialkonstruert veske er festet fast til underdelen slik at batteriet ikke tar plass på festeplater og bidrar til å senke tyngdepunktet for hele systemet. I stede for bein som går ut fra midten som på hexa-konfigurasjonen så har quad-oppsettet fått spikes festet ytterst i hver arm. Disse bør plasseres rett under motorene. Disse kan lages i aluminium og skrus fast i de små festehullene under motorene.

7 Elektronisk oppsett

7.1 Oppkobling av BeagleBoard



Figur 2 – Blokkskjema over de ytre modulene tilkoblet «BeagleBoard xM»

Blokkskjema viser de ytre komponentene som er tilkoblet BeagleBoard. Se vedlagt datablad for BeagleBoard for å se kontaktplassering på kortet

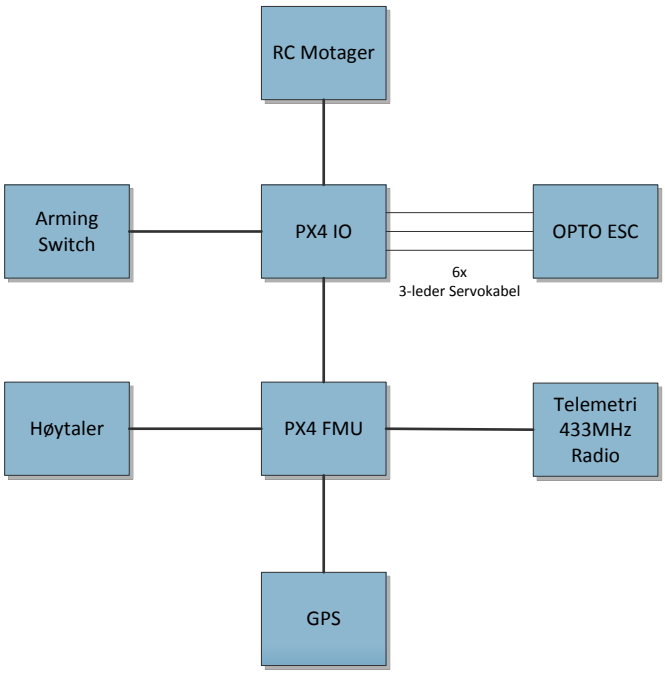
RS-232 grensesnittet brukes for terminaltilgang til BeagleBoard. Dette er en svært nyttig «debugging» tilkobling ettersom man her kan se utputt fra bootloaderen. Det er også mulig å benytte Ethernet tilkoblingen (ikke tegnet inn) eller USB-WiFi for terminal tilkobling til BeagleBoard.

USB flashminne kobles inn i USB-huben for primærlagring av bilder og video, resten av filsystemet (rootfs) ligger på et microSD kort.

Kameraet vi benytter har en proprietær tilkobling bestemt av Leopard Imaging, produsenten av kamerakortet.

For kommunikasjonen mellom BeagleBoard og PX4 er det flere muligheter, men å kjøre det over en FTDI-kabel (3,3V UART) via USB virker som det mest trolige valget.

7.2 Oppkobling av PX4



Figur 3 - Modulene tilkoblet PX4

Blokkskjema viser de ytre komponentene tilkoblet PX4. Se vedlagt datablad for å se plasseringen av kontaktene på kortene.

PX4 FMU og PX4 I/O er koblet sammen med en ekspansjonskontakt som er montert på kortene, denne er plassert slik at modulene stables oppå hverandre. Denne kontakten fører en del av signalene fra «FMU» opp til kontakter på «I/O».

RC mottageren er en egen fysisk boks som går med et seriegrensesnitt til PX4 I/O, der er det 3 mulige grensesnitt å koble seg inn med; PPM, Futaba S.Bus og Spektrum Satelite. Vi har i skrivende stund bestilt mottager som har S.Bus og Satelite, men hvilket av grensesnittene som vi endelig kommer til å benytte er ikke avgjort enda.

OPTO ESC (kun én er tegnet inn, det vil være fire stykk for quad, seks for hexa) er koblet opp med en tre-leder servokabel. Denne kabelen overfører signal, GND og den siste lederen (midtre) er ikke koblet opp internt i ESC, normalt ville denne benyttes for overføring av 5V. Disse er også koblet opp til batteri, se *Figur 4 - Blokkskjema over strømfordeling*. Ettersom 5V lederen ikke er tilkoblet i OPTO ESC benyttes det derfor ekstern BEC. I ESC som ikke er OPTO, sitter det også en BEC for å tilby 5V til RC-mottager og annet utstyr. Signalet som blir sendt til ESC er PWM og DC bestemmer hastigheten. Disse ESC lager trefase til motorene, hver motor har sin dedikerte ESC (motorer ikke illustrert i *Figur 3 - Modulene tilkoblet PX4*).

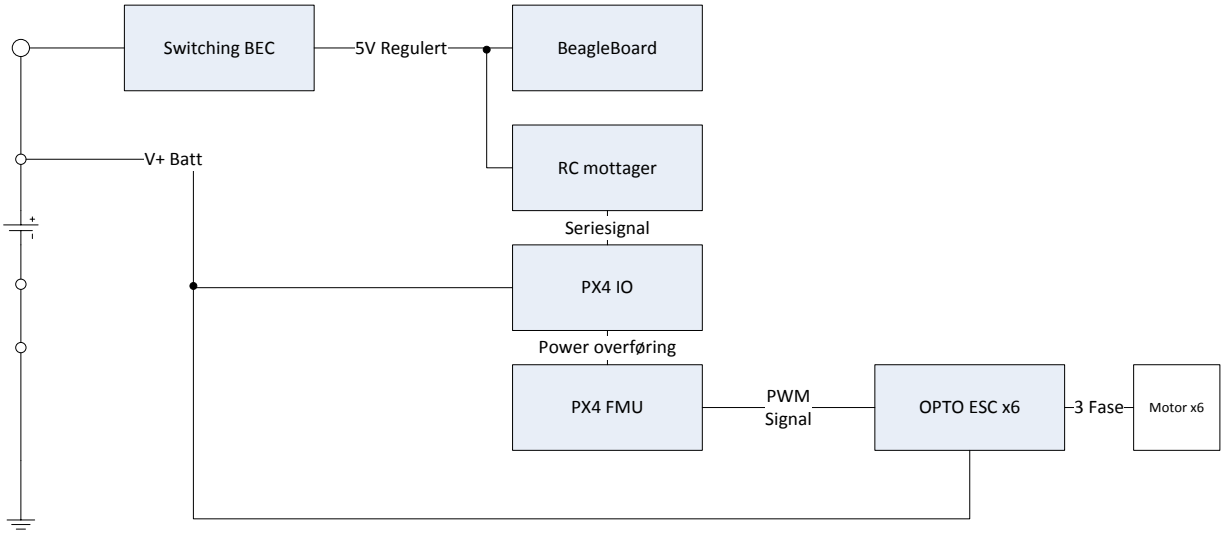
En Arming Switch er i utgangspunktet en helt vanlig impuls bryter, men den er også utstyrt med en LED som indikerer status på systemet. Bryteren benyttes for å fortelle autopiloten at den får lov til å starte motorene.

«Buzzeren» er en piezoelektrisk høyttaler som benyttes av autopiloten for å indikere statusen på systemet, blant annet gir den lyd når den autopiloten har «bootet» opp.

Telemetrien er koblet opp med et 3,3v UART grensesnitt. Denne radioen er en mer eller mindre transparent UART forbindelse.

GSP kobles inn i en egen dedikert kontakt på PX4 FMU, grensesnittet til GSPen er UART. Selve GPS-modulen støtter også USB og I^2C.

7.3 Oppkobling av strømfordeling



Figur 4 - Blokkskjema over strømfordeling

Det er to strømsløyfer i hexakopteret en batterisløyfe og en regulert 5V sløyfe levert av en switching BEC med kapasitet på 5A. Begge sløyfene er lagt ut på et strømfordelingskort (ikke illustrert i figuren), dette brukes i stedet for de ferdige kobberbanene på hovedkroppen til den flygende plattformen(se: *S008 - om utstyret*). Dette gjøres fordi systemet blir mindre modulært ved å integrere strømfordeling i plattformens kropp. Alle strømledere har påmontert bananplugger for enklere montering / demontering, slik at man ikke trenger å lodde av kablene.

Batterikursen er koblet til alle OPTO ESC, SBEC og PX4 I/O. PX4 I/O reguler batterispenningen og leverer også power til PX4 FMU. PX4 FMU har en batterifølekrets (trolig A/D converting) som gir alarm når batterispenningen begynner å bli lav.

OPTO ESC har ingen integrert BEC og vi benytter derfor en ekstern SBEC for å levere power til RC-mottageren og BeagleBoard. OPTO ESC har trefase utganger for tilkobling av motor.

SBEC kan velges til enten å levere 5V eller 6V, men i vårt system låses denne til 5V. BeagleBoard har innebygget overspenningsvern og RC-mottageren klarer ganske gode variasjoner i spenning både oppover og nedover (for at den ikke skal slutte å fungere dersom man begynner å ha lite strøm).

I skrivende stund er det ingen eksterne indikatorer eller liknende som krever tilførsel utover dette, men systemet designes modulært nok til at de vil takle videre utvikling.

8 Vedlegg

- SolidWorks eDrawing pdf:
    - Bunnplate
    - Mellomplate
    - Basebein
    - Bein
- SolidWorks styrkeberegningsrapport

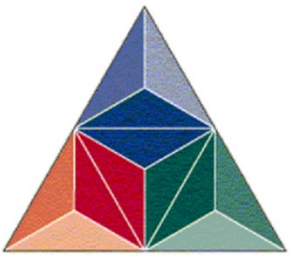
- Koblingsskjemaer «MS Visio» dokumenter:
    - BeagleBoardKobling
    - Koblingsskjema
    - PX4kobling
- Skjema over kontakter på PX4
    - px4fmu-manual-v1.7
    - px4io-manual-v1.3

For mer dokumentasjon om PX4 og tilhørende utstyr refereres det datablader mappen.

9 Referanser

1. Dessault. *SolidWorks*. 05.03.2013]; Available from: http://www.solidworks.no/.
2. Innovations, D. *DJI main page*. 05.03.2013]; Available from: http://www.dji-innovations.com/.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Kravspesifikasjon
Dokumenttype:	Planleggingsdokument
Dokument ID:	P003
Versjonsnummer:	V2.0
Versjonsdato:	26.04.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	4.1.2013	Første offisielle versjon.
V2.0	26.4.2013	Justert prioriteter og splittet krav etter møte med oppdragsgiver.

3 Introduksjon

Oppgaven, som er gitt av *Kongsberg Defence Systems*, går ut på å utvikle en autonom flygende plattform med videokapasitet. Denne tenkes brukt som substitutt for mer avanserte droner i en demosituasjon. De krav som oppdragsgiver stiller til systemet vil i første omgang fremkomme i dette dokumentet. Kravene skal være formulert på en slik måte at det kan testes om kravet er oppnådd. Eventuelle krav gitt av Høgskolen i Buskerud og prosjektgruppa behandles ikke i dette dokumentet, men kun i OnTime prosjektstyringsverktøy.

4 Kravenes struktur

Kravene til et system kan ha ulik viktighet, eller prioritet. Det er vanlig praksis å presisere ulike prioriteter. Alle krav vil derfor få en prioritetsangivelse. De ulike prioritetsnivåene er som følger:

- A-krav -De høyest prioriterte kravene, må oppnås.
- B-krav -Mindre viktig enn A-krav, men fortsatt strengt ønskelig å oppnå.
- C-krav -Krav som det er ønskelig, men ikke nødvendig å oppnå.

Det kan stilles krav til utførelsen av prosjektet, uten at det har innvirkning på systemet som utvikles. Disse er å anse som ikke-funksjonelle krav, men er likevel ulike de ikke-funksjonelle kravene til systemet. I et mer komplekst system vil det derfor kunne være hensiktsmessig å dele inn i flere typer krav. Det vi derimot observerer er at det like før første presentasjon ikke foreligger en veldig stor mengde krav. Det kan i tillegg være vanskelig å kategorisere kravene korrekt dersom hver kategori er for smal. Derfor velger vi å dele inn i følgende to kravtyper:

- Systemkrav
- Prosjektkrav

Hvert krav vil ha en unik ID som består av bokstaven K, for krav, etterfulgt av tre tall. Eksempelvis: *K006*. Hvis et krav endres skal det beholde sin ID slik at det er sporbart. Hvis et krav bortfaller kan IDen ikke gjenbrukes. Hvert krav bør ha et navn, dette trenger ikke være unikt. Den som fremsatte kravet er kilden til kravet. Hvert krav skal ha en entydig beskrivelse. Hvert krav skal ha minst én relatert test, men i mange tilfeller vil det være naturlig å ha flere.

5 Tilpasning av kravspesifikasjon for Scrum

Prosjektgruppa kommer til å jobbe etter prosjektmodellen Scrum. I forbindelse med dette vil prosjektstyringsverktøyet OnTime være sentralt i organisering og styring av prosjektet. Krav vil dermed bli behandlet som user stories i OnTime. For å tydeliggjøre kravene som stilles fra oppdragsgiver bedre enn det som er mulig i OnTime, velger vi å beholde dette dokumentet og føre opp alle krav her.

For oppfølging og arbeid i OnTime vil hvert krav / hver user story videre deles inn i mindre omfattende user stories. Dette gjøres for at arbeid skal være lettere å fordele, og for å bedre holde kontroll på hvor mye tid som påkreves til hvert gjøremål.

5.1 Krav-ID / User story ID

Hver user story får en automatisk, auto-inkrementert tallverdi tildelt når den opprettes. Hvert unikt krav med en unik krav-ID vil derfor være koblet til en eller flere user story ID. Hver user story ID er kun koblet til én unik krav-ID.

5.2 Prioritet

A, B og C prioritet vil bli videreført til OnTime i form av informasjon i kommentarfeltet på hver user story. Videre vil hver user story bli tildelt en prioritet som relaterer til arbeidsflyt i prosjektet og hvilke arbeidsoppgaver som prioriteres gjennomført. Disse prioritetene er henholdsvis: *very low, low, medium, high* og *very high*. Prioriteten bestemmes i utgangspunktet av *product owner*.

5.3 Navn

Kravenes navn, slik de fremgår i dette dokumentet, er ikke videreført til OnTime. Det er krav-ID og user story ID som er det unike bindeleddet mellom et krav og en user story. I dette dokumentet trenger ikke hvert krav et unikt navn, men i OnTime vil det derimot kunne skape forvirring dersom ulike user stories har samme navn. Navnene som blir tildelt user stories i OnTime er mer lik det som er testens beskrivelse i dette dokumentet.

5.4 Krav-type

I OnTime er prosjektet delt inn i to hovedkategorier; Prosjektstyring og System. Prosjektkrav har sin egen mappe under Prosjektstyring, og systemkrav har sin egen mappe under System.

5.5 Relaterte tester

Relaterte tester er én eller flere test-ID på formatet Txxx (for eksempel T001). Disse testene finnes i P004 – Testspesifikasjon. P004 omhandler også hvordan testing oppfølges i OnTime.

6 Kravskjema

Følgende skjema skal benyttes for å spesifisere hvert enkelt krav, her utfylt med eksempelinformasjon eller en kort beskrivelse slik det passer seg.

Krav-ID:	K001	**Navn på krav:**	Kort navn (ikke unikt)
Prioritet:	A, B eller C	**User story ID:**	Tallverdi fra OnTime
Krav-type:	Prosjektkrav eller Systemkrav	**Relaterte tester:**	T001, T002
Dato opprettet:	01.01.1970	**Dato endret:**	31.12.2012
Beskrivelse:	Beskrivelse av et testbart krav. **Endring 31.12.2012:** Beskrivelse av endringer.		

7 Krav

Krav-ID:	K001	**Navn på krav:**	Merking av bilder
Prioritet:	A	**User story ID:**	40
Krav-type:	Systemkrav	**Relaterte tester:**	T001
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Bilder skal inneholde geografiske referanser og tidsmerking		

Krav-ID:	K002	**Navn på krav:**	Merking av video
Prioritet:	B	**User story ID:**	114
Krav-type:	Systemkrav	**Relaterte tester:**	T002
Dato opprettet:	4.10.2012	**Dato endret:**	26.04.2013
Beskrivelse:	Video skal inneholde geografiske referanser og tidsmerking. **Endring 26.04.2013:** Justert ned til B-krav.		

Krav-ID:	K003	**Navn på krav:**	Operasjonsplanlegging
Prioritet:	A	**User story ID:**	39, 50, 115
Krav-type:	Systemkrav	**Relaterte tester:**	T003, T004
Dato opprettet:	4.10.2012	**Dato endret:**	26.04.2013
Beskrivelse:	Systemoperatøren skal kunne planlegge oppdrag. Dette inkluderer: 1. Opprette rute der fartøyet skal fly i form av veipunkter 2. Definere operasjonshøyde for veipunkter 3. Definere hvor fartøyet skal samle informasjon **Endring 26.04.2013:** • Punkt 1 omformulert fra “luftkorridorer” til “rute i form av veipunkter”. • Punkt 2 omformulert fra minimum og maksimum høyde til høyde for veipunkter. • Punkt 4 om koordinatsystemer er skilt ut til et eget B-krav, K014.		

Krav-ID:	K004	**Navn på krav:**	Systembegrensningsvarsel
Prioritet:	A	**User story ID:**	51
Krav-type:	Systemkrav	**Relaterte tester:**	T005
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Systemet skal melde fra til systemoperatøren dersom et planlagt oppdrag ikke er mulig på grunn av systemets/fartøyets begrensninger		

Krav-ID:	K005	**Navn på krav:**	Flyvetid
Prioritet:	B	**User story ID:**	47
Krav-type:	Systemkrav	**Relaterte tester:**	T006
Dato opprettet:	4.10.2012	**Dato endret:**	05.01.2013
Beskrivelse:	Fartøyet skal kunne fly i minst 15 minutter. **Endring 05.01.2013:** Prioritet justert ned fra A til B etter rådføring med ekstern veileder.		

Krav-ID:	K006	**Navn på krav:**	Informasjonslagring
Prioritet:	A	**User story ID:**	38
Krav-type:	Systemkrav	**Relaterte tester:**	T017
Dato opprettet:	4.10.2012	**Dato endret:**	26.04.2013
Beskrivelse:	Bilder skal kunne lagres om bord på fartøyet. **Endring 26.04.2013:** Video skilles ut til eget B-krav, K015.		

Krav-ID:	K007	**Navn på krav:**	Autonom takeoff og landing
Prioritet:	B	**User story ID:**	36, 37
Krav-type:	Systemkrav	**Relaterte tester:**	T008, T016
Dato opprettet:	4.10.2012	**Dato endret:**	5.10.2012
Beskrivelse:	Fartøyet skal kunne ta av og lande uten menneskelig interaksjon. **Endring 5.10.2012:** Prioritet justert ned fra A til B.		

Krav-ID:	K008	**Navn på krav:**	Budsjett
Prioritet:	B	**User story ID:**	117
Krav-type:	Prosjektkrav	**Relaterte tester:**	T013
Dato opprettet:	4.10.2012	**Dato endret:**	20.12.2012, 26.04.2013
Beskrivelse:	Budsjett for fartøyet er NOK 10.000,-. Denne grensen er mulig å forhandle dersom det foreligger god grunn. **Endring 20.12.2012:** Budsjett satt til NOK 15.000,- i samråd med oppdragsgiver. **Endring 26.04.2013:** Prioritet justert ned fra A til B.		

Krav-ID:	K009	**Navn på krav:**	Programvaredokumentasjon
Prioritet:	A	**User story ID:**	118
Krav-type:	Prosjektkrav	**Relaterte tester:**	T014
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Programvaren skal dokumenteres med Doxygen, Javadoc eller liknende.		

Krav-ID:	K010	**Navn på krav:**	Endring av oppdrag
Prioritet:	B	**User story ID:**	53
Krav-type:	Systemkrav	**Relaterte tester:**	T009, T015
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Systemoperatøren skal kunne endre det planlagte oppdraget under flygning.		

Krav-ID:	K011	**Navn på krav:**	Dataoverføring
Prioritet:	B	**User story ID:**	54
Krav-type:	Systemkrav	**Relaterte tester:**	T010
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Det skal være mulig for systemoperatøren å laste ned video og bilder under flygning.		

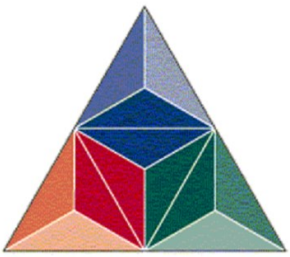
Krav-ID:	K012	**Navn på krav:**	Live dataoverføring
Prioritet:	B	**User story ID:**	34
Krav-type:	Systemkrav	**Relaterte tester:**	T011
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Det skal være mulig for systemoperatøren å se direkteoverført video under flygning.		

Krav-ID:	K013	**Navn på krav:**	Læringskurve
Prioritet:	B	**User story ID:**	55
Krav-type:	Systemkrav	**Relaterte tester:**	T012
Dato opprettet:	4.10.2012	**Dato endret:**	
Beskrivelse:	Det skal være mulig å lære seg å bruke systemet på under 60 minutter.		

Krav-ID:	K014	**Navn på krav:**	Koordinatsystemer
Prioritet:	B	**User story ID:**	116
Krav-type:	Systemkrav	**Relaterte tester:**	T018
Dato opprettet:	26.04.2013	**Dato endret:**	
Beskrivelse:	Systemoperatøren skal kunne sette inn koordinater i både lat/long og MGRS.		

Krav-ID:	K015	**Navn på krav:**	Informasjonslagring
Prioritet:	B	**User story ID:**	297
Krav-type:	Systemkrav	**Relaterte tester:**	T007
Dato opprettet:	26.04.2013	**Dato endret:**	
Beskrivelse:	Video skal kunne lagres om bord på fartøyet.		

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.
System for Tactical Aerial Reconnaissance

Vedlegg til P003 - Kravspesifikasjon

1 Avledede krav

1.1 K001 Bilder skal inneholde geografiske referanser og tidsmerking

- PX4-Krav
    - Opprette en UART for kommunikasjon med BB
    - Abonnere på GPS posisjon
    - Abonnere på attitude
    - Svare på klokkeforespørsel over UART med GPS-tid
    - Svare på forespørsel over UART med attitude/pos
    - Sende kommando over UART for start av video
    - Sende kommando over UART for stopp av video
    - Sende kommando over UART om å ta bilde
    - Motta ack fra BB på om billedtakning er OK, evt m/feilmelding
    - Sende melding til bakkestasjon om billedtakning med tid og sted
    - Hente ut informasjon om billedtakning fra relevante meldinger/waypoints
    - Hente ut informasjon om videotakning fra relevante meldinger/waypoints
    - Autostart av relevante script ved oppstart
- BeagleBoard-Krav
    - Opprette UART for kommunikasjon med PX4
    - Motta forespørsel over UART om å ta bilde
    - Motta forespørsel over UART om å starte video
    - Motta forespørsel over UART om å stoppe video
    - Autostarte relevant script ved oppstart
    - Forespørre/ synkronisere / motta GPS-tid over UART
        - Sette systemtid med mottatt tid
    - Starte script som leser ut og lagrer bilder
    - Starte script som leser ut og lagrer video
    - Navngi opprettede filer / generere filnavn
    - Loggføre korrekt tid for tid for et bilde som er tatt
    - Loggføre korrekt tid for alle frames i video som er tatt
    - Lagre bilder på SD-kort
        - Opprette path for lagring av bilder
        - Evt. Opprette en database
    - Verifisere at ønsket operasjon er vellykket
    - Sende melding over UART for tatt bilde
        - Inneholder filnavn, tid og posisjon, evt WP-nummer
    - Sende melding over UART for startet video
        - Inneholder filnavn, tid og posisjon, evt WP-nummer
    - Sende melding over UART for stoppet video
        - Inneholder filnavn, tid og posisjon, evt WP-nummer
    - Motta posisjon over UART
    - Motta attitude over UART

    - Kjør selv-test ved oppstart
        - Kontroller om oppgitt filbane for lagring er tilgjengelig
            - Hvis nei, opprette filbanen
            - Kontroller skrivetilgang
        - Kontroller ledig plass på lagringsmedia
- Organisere / merke bilder og video med metadata
    - Skrive header
    - Lese header
    - Operere på mer enn én fil av gangen
    - Klargjøre data ihht STANAG
    - Lese fra fil
    - Skrive til fil
    - Kopiere data (ikke jobbe på originaler)
    - Organisere / omdøpe filer

1.2 K006 Video og bilder skal kunne lagres om bord på fartøyet

- Lagringsmedie må være tilgjengelig
    - Gi feilmelding om det ikke er tilgjengelig
    - Bruk eventuelt primærlager (SD-kort)
    - Skrivetilgang må finnes
    - Lagringsmediet må ha tilstrekkelig plass

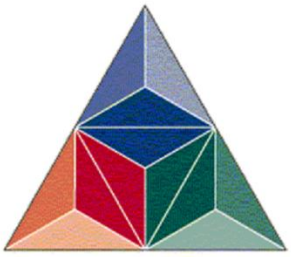
1.3 K003 Definere hvor fartøyet skal samle informasjon

- Bakkestasjon må støtte funksjonalitet som definerer billedtakning
- Bakkestasjon må støtte funksjonalitet som definerer start og stopp videoopptak
- Instruksjoner om billed- og videotakning må sees i forbindelse med koordinater

1.4 K003 Sette inn koordinater i både lat/long og MGRS

- Konvertere fra lat/long til MGRS
- Konvertere fra MGRS til lat/long
- Input / koordinater må valideres
    - Feilmelding ved feil format
    - Feilmelding ved ugyldig data
- GUI / brukergrensesnitt må støtte både lat/long og MGRS

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Testspesifikasjon
Dokumenttype:	Planleggingsdokument
Dokument ID:	P004
Versjonsnummer:	V5.0
Versjonsdato:	23.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	04.01.13	Ferdigstilt versjon, ny logo
V2.0	11.01.13	Avsnitt om Design by contract
V3.0	03.03.13	Oppdaterte testbeskrivelser
V4.0	23.05.13	Endret ordlyden i tester etter møte med oppdragsgiver
V5.0	24.05.13	Endring av veipunkter

3 Introduksjon

Dette dokumentet drøfter ulike testmetoder og hvordan vi vurderer å benytte disse. Videre er det beskrevet de tester vi skal gjennom for å verifisere kravene som er stilt til systemet.

4 Ordliste

Radiolink	433MHz dataforbindelse mellom bakkestasjon og flygende plattform
Wi-Fi link	Wi-Fi forbindelse mellom bakkestasjon og beagleboard / flygende plattform
RC-link	2.4GHz forbindelse mellom manuell kontroller og flygende plattform
Bakkestasjon	PC med QGroundControl, manuell kontroller, Wi-Fi router
Lat/long	Latitude / longitude, koordinatsystem
MGRS	Military Grid Reference System, koordinatsystem

5 Testmetoder

Vi vil her ta for oss ulike relevante testmetoder.

5.1 Test under utvikling

Siden vi bruker Scrum som prosjektmodell, vil testing skje både fortløpende og iterativt. En user story i Scrum er ikke ferdig før den er ferdig testet, og et av stegene i utviklingen er «Ready for Test».

5.2 Testing i OnTime

Når en user story ligger i «Ready for Test» skal den testes før den kan flyttes til «Completed», hvordan en funksjonell user story skal testes vil være definert i beskrivelsen. Alle tester som kan relatere til krav fremsatt av oppdragsgiver, hovedsakelig systemtester av større user stories, vil beskrives på en mer formell måte.

5.3 Unit-testing

Unit testing er en måte å teste programvare på som tar for seg oppførselen/funksjonaliteten til den minste kodebiten som kan bli testet i et isolert miljø. Det finnes mange måter å bruke unit-testing på, og vi skal ta for oss noen av disse.

5.3.1 Test-Driven Development

Mens unit-testing bare forteller oss hva vi skal teste, vil TDD definere når vi skal teste. TDD er et rammeverk hvor testene bestemmer alt du trenger å vite, de tar for seg design, hva man skal gjøre videre, og når man er ferdig. I TDD er det testene som driver utviklingen fremover.

[1] TDD er en utviklingsprosess som baserer seg på repetisjon av en veldig kort utviklingsfase hvor man tester før man implementerer. Stegene i TDD er som følger:

- Lag en automatisert test.
- Kjør test.
- Implementer en løsning som passerer testen.
- Se at testen blir passert.
- Rydd opp i koden.
- Repeter.

Utviklingen av hver enkelt modul starter med å skrive en test. Denne testen kommer uansett til å feile gitt at koden ikke er skrevet enda. For å skrive en test må den som skal utvikle dette forstå alle spesifikasjonene og kravene til modulen. Det som gjør denne metoden så effektiv er at utvikleren må fokusere på kravene først, og deretter skrive den nødvendige koden.

Det neste steget blir å kjøre testen, testen sørger for at koden utvikleren skriver skal dekke alle krav og spesifikasjoner, og skal feile første gangen den blir kjørt. Dette gjør vi for å sikre at testen ikke blir passert uten å trenge noe ny kode.

Neste logiske steg vil være å skrive koden som skal til for å passere testen som vi nettopp implementerte. Koden trenger hverken være elegant eller ryddig, og har kun som hensikt å bestå testen. Hvis koden vår passerer testen vil alle krav til modulen nå være oppfylt.

Vi har nå passert testen og det er nødvendig å rydde opp i koden vi nettopp skrev, det er viktig at testen blir kjørt flere ganger mens opprydningen skjer for å hindre at ønsket funksjonalitet blir borte i løpet av opprydningsprosessen.

Deretter lager vi neste test. For å øke systemets funksjonalitet skal disse stegene typisk repeteres helt til alle moduler er ferdig utviklet. Stegene mellom hver test bør ikke være for store.

5.3.2 Assertions

En Assertion [2] er en sjekk for en egenskap som må være sann, og brukes typisk aktivt i programvareutvikling for å sjekke om funksjoner ikke gir ut feil verdier. Dette gjøres ved å si sette inn en kodelinje som sier; hvis ikke dette er sant, avbryt og gi beskjed. Det som gjør Assertions så nyttig er at det kan brukes av alle som eventuelt jobber med samme kode og kan bli stående å teste programmet helt til utviklingen er ferdig. Dette gjør det lettere for personen som ikke har skrevet koden der feilen oppstår, fordi personen som skrev det har lagt inn en Assertion som sier hva feilen er, og hvor den oppstår.

5.3.3 Design by contract

[3] Dette er en metode innenfor programvaretesting som går godt sammen med Assertions. Poenget med denne typen test er å sette opp en slags «kontrakt» mellom en klasse/modul og en klient, hvor klassen garanterer utlevering av riktige verdier(postcondition) hvis verdiene som kommer inn er korrekte (precondition). På denne måten kan man lett se hvilke sider av rutinen feilen ligger på.

En precondition kan være at en verdi som skal sendes inn i modulen må være positiv, hvis dette går galt vil feilen ligge i klienten. Når en precondition ikke blir oppnådd vil ikke rutinen i klassen kunne garantere leveranse av riktige verdier, og postcondition vil heller ikke bli oppnådd.

5.3.4 Code Coverage

[4] Code Coverage er et veldig nyttig hjelpemiddel og verktøy i systematisk unit-testing. I programvareutvikling brukes Code Coverage til å holde oversikt over hvilke deler av kildekoden som ble inkludert i testen. Alle kall i koden som ikke blir kjørt av testen, f. eks en del av en if-setning eller et retur-kall fra en funksjon, vil vises av Code Coverage verktøyet.

Utvikleren kan da gjøre om på testen så koden som ikke ble kjørt også kan bli testet, ofte kan dette vise til bugs tidligere i koden som utvikleren har oversett fordi enkelte kall ikke ble dekket av testen.

5.3.5 xUnit

[5] xUnit er en samlebetegnelse for unit-testing biblioteker, som (oftest) er open source og gratis.

Rammeverk for unit-testing er veldig mye brukt og finnes for stort sett alle språk. Et rammeverk av dette slaget vil gi utviklere en stor fordel ved å holde styr på hva tester skal resultere i og tilby automatiserte løsninger uten å måtte skrive de samme testene flere ganger. De støtter testing av forskjellige enheter av programvaren som f.eks. funksjoner og klasser.

5.3.5.1 Boost

Boost er et xUnit bibliotek for C++ som også inneholder endel andre funksjoner.

5.3.5.2 JUnit

JUnit [6] er rammeverket for unit-testing i Java.

5.3.5.3 CUnit

Et lettvekt unit-testing rammeverk for C.

5.4 Testing av dokumenter

Gitt at dette er et skoleprosjekt vil dokumenter også bli testet når de er ferdigstilte, testing av et dokument vil f. eks være korrekturlesning og kvalitetsjekking. Personen ansvarlig for dokumentet bør nødvendigvis ikke skrive eller teste dette dokumentet, men er ansvarlig for at det blir ferdig i tide og gjort riktig.

Når personen ansvarlig føler dokumentet er ferdigstilt vil testing foregå i form av at dokumentet flyttes til steget i administrasjonsverktøyet navngitt "Ready for Test", og kan ikke flyttes til "completed" før en annen person i gruppa har sett på det. Vi gjør dette for å kvalitetssikre dokumentasjonen vi produserer.

5.5 Systemtesting

Det ferdige produktet vårt, som i utgangspunktet kun skal være et «proof of concept», skal kunne fly autonomt, filme, ta bilder og geolokalisere innsamligsdata, dette skal lagres til plattformen mens video også skal sendes til bakkestasjonen. Bakkestasjonen skal være brukerstyrt og bestå av et grensesnitt som bestemmer ruten til plattformen samt vise brukeren flydata fra plattformen. Dette systemet må testes som en helhet for å forsikre oss om at det fungerer som det skal.

Det ferdige systemet kan deles opp i mindre systemer, det vil si at vi kan jobbe med og teste funksjonaliteten til disse uavhengig av hverandre. For å ta et eksempel vil det systemet som skal filme/ta bilder og sende til bakkestasjonen kunne utvikles og testes uten at plattformen er i luften.

Funksjonalitet som skal testes for systemet i sin helhet og som kan brytes ned i mindre biter er:

- Kommunikasjon/overføring
- Datalagring
- Flytid
- Flyrute
- Brukergrensesnitt og kartsystem
- Geotagging av bilder og video

Poenget blir da å lage caser som dekker disse forskjellige funksjonalitetene til systemet for å mer eller mindre teste systemet i sin helhet.

6 Testenes struktur

6.1 Testskjema

Følgende skjema skal benyttes for å spesifisere hver enkelt test.

Test-ID:	*Unik ID*	**Relaterte krav:**	*Krav-ID*
Hva testes:	*Beskrivelse av hva som skal testes*		
Optimalt scenario:	*1. Første steg* *2. Videre instruksjoner* *3. Avslutning*		
Utvidet scenario:	*2a. En forutsett, akseptabel feilsituasjon i steg 2* *a. Forslag til løsning* *b. Annet forslag til løsning* *2b. Uakseptabel feilsituasjon i steg 2* *a. Test feilet*		

7 Tester

Alle tester kan relateres til et bestemt krav, kravet som testen relaterer seg til står også oppgitt i testskjemaet. Krav-ID blir skrevet på formatet «Kxxx», f. eks K005, og blir beskrevet i dokumentet *P003 – Kravspesifikasjon*.

7.1 Systemtester

Test-ID:	T001	**Relaterte krav:**	K001
Hva testes:	Bilder skal inneholde geografiske referanser og tidsmerking.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer et oppdrag med billedtakning 3. Systemet utfører oppdraget 4. Operatør laster bilder over til bakkestasjon 5. Operatør kontrollerer bilder for merking i henhold til NATO standard		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2a. GPS signaler ikke tilgjengelig a. Test feilet 2b. Minnekort ikke tilgjengelig a. Test feilet 2c. Oppdrag lar seg ikke planlegge / eksekvere a. Test feilet 3a. Autonom takeoff/landing ikke implementert a. Operatør tar av manuelt b. Operatør lander manuelt 3b. Autonom flygning er ikke implementert a. Operatør gjennomfører testen med manuell kontroll 4. Filer ikke tilgjengelig a. Test feilet 5. Korrekt merking ikke på plass a. Test feilet		

Test-ID:	T002	Relaterte krav:	K002
Hva testes:	Video skal inneholde geografiske referanser og tidsmerking.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer et oppdrag med videofilming 3. Systemet utfører oppdraget 4. Operatør laster videofiler over til bakkestasjon 5. Operatør kontrollerer video for merking i henhold til NATO standard		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2a. GPS signaler ikke tilgjengelig a. Test feilet 2b. Minnekort ikke tilgjengelig a. Test feilet 3a. Autonom takeoff/landing ikke implementert a. Operatør tar av manuelt b. Operatør lander manuelt 3b. Autonom flygning er ikke implementert a. Operatør gjennomfører testen med manuell kontroll 4. Filer ikke tilgjengelig a. Test feilet 5. Korrekt merking ikke på plass a. Test feilet		

Test-ID:	T003	Relaterte krav:	K003
Hva testes:	Systemoperatøren skal kunne planlegge oppdrag. Dette inkluderer: 1. Opprette rute der fartøyet skal fly i form av veipunkter 2. Definere operasjonshøyde for veipunkter		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør programmerer en rute gjennom de kjente, lett gjenkjennelige punktene A, B og C 3. Operatøren legger inn en kjent, målbar operasjonshøyde for det planlagte oppdraget 4. Systemet utfører det planlagte oppdraget 5. Operatøren observerer og vurderer utførelsen av oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke Waypoints a. Test feilet 3. Systemet aksepterer ikke operasjonshøyde a. Test feilet 4. Autonom takeoff og/eller landing er ikke implementert a. Operatøren tar av manuelt b. Operatøren lander manuelt 5. Systemet utfører ikke det planlagte/programmerte oppdraget a. Test feilet		

Test-ID:	T004	Relaterte krav:	K003
Hva testes:	Systemoperatøren skal kunne planlegge oppdrag, herunder hvor fartøyet skal samle informasjon.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør programmerer en rute gjennom de kjente, lett gjenkjennelige punktene A, B, C og D 3. Operatør instruerer systemet om å ta videoopptak mellom punkt A og B 4. Operatør instruerer systemet om å ta bilder i punkt C og D 5. Systemet utfører det planlagte oppdraget 6. Operatør inspiserer videoelementer 7. Operatør inspiserer bilder		
Utvidet Scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke Waypoints a. Test feilet 3. Systemet aksepterer ikke instruksjon om videoopptak a. Test feilet 4. Systemet aksepterer ikke instruksjon om å ta bilder a. Test feilet 5. Autonom takeoff/landing er ikke implementert a. Operatør tar av manuelt b. Operatør lander manuelt 6. Video er ikke av området mellom punkt A og punkt B a. Test feilet 7. Bilder er ikke av områdene C og D a. Test feilet		

Test-ID:	T005	*Relaterte krav:*	K004
Hva testes:	Systemet skal melde fra til systemoperatøren dersom et planlagt oppdrag ikke er mulig på grunn av systemets/fartøyets begrensninger.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og eventuelt flygende plattform 2. Operatør planlegger oppdrag med operasjonshøyde utenfor den flygende plattformens kapasitet 3. Operatør planlegger oppdrag med operasjonshøyde under lokalt minimum (waypoint befinner seg under bakken) 4. Operatør planlegger oppdrag med rekkevidde som overskrider kapasiteten til den flygende plattformen ved anslått batterikapasitet		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer oppdraget a. Test feilet 3. Systemet aksepterer oppdraget a. Test feilet 4. Systemet aksepterer oppdraget a. Test feilet		

Test-ID:	T006	*Relaterte krav:*	K005
Hva testes:	Fartøyet skal kunne fly i minst 15 minutter.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer et oppdrag uten videoopptak eller billedtakning 3. Systemet utfører oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke oppdrag a. Manuell kontroll aktiveres b. Testen gjennomføres ved manuell flygning 3a. Autonom takeoff/landing er ikke implementert a. Operatør tar av manuelt b. Operatør lander manuelt 3b. Autonom flygning er ikke implementert a. Operatør gjennomfører testen med manuell kontroll 3b. Oppdraget fullføres innen 15 minutter er passert a. Operatør initierer nytt oppdrag innen plattformen har landet 3c. Systemet går tom for strøm før 15 minutter er passert a. Test feilet		

Test-ID:	T007	Relaterte krav:	K006
Hva testes:	Video skal kunne lagres om bord på fartøyet.		
Optimalt scenario:	**Test i systemkonfigurasjon:** 1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer oppdrag med videoopptak, et enkelt scenario der plattformen hovrer er tilstrekkelig 3. Systemet utfører oppdraget 4. SD-kort inspiseres for lagret videoopptak **Forenklet test på komponentnivå:** 1. Koble til BeagleBoard 2. Aktiver protokoll for lagring av video 3. Komponentene filmer og lagrer video 4. SD-kort inspiseres for lagret videoopptak		
Utvidet scenario:	**Test i systemkonfigurasjon:** 1a. Bakkestasjon og/eller flygende plattform starter ikke a. Operatør kjører forenklet test på komponentnivå 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Operatør kjører forenklet test på komponentnivå 2a. Oppdraget lar seg ikke initiere a. Operatør kjører forenklet test på komponentnivå 3a. Autonom takeoff/flygning/landing er ikke implementert a. Operatør initierer manuell kontroll b. Operatør gjør manuell styring der det trengs 3b. Autonom flygning er ikke implementert a. Operatør gjennomfører testen med manuell kontroll 4. SD-kort inneholder korrupt eller ingen data a. Test feilet **Forenklet test på komponentnivå:** 1. Tilkobling til BeagleBoard feiler a. Test feilet 2a. Kobling mellom BeagleBoard og kamera kan ikke opprettes a. Test feilet 2b. Protokoll for lagring av video ikke tilgjengelig a. Test feilet 3. Komponentene filmer ikke a. Test feilet 4. SD-kort inneholder korrupt eller ingen data a. Test feilet		

Test-ID:	T008	Relaterte krav:	K007
Hva testes:	Fartøyet skal kunne ta av uten menneskelig interaksjon.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer et enkelt oppdrag (ta av og lande) 3. Systemet utfører oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke oppdrag a. Test feilet 3a. Systemet tar ikke av autonomt a. Test feilet 3b. Systemet lander ikke autonomt a. Operatør aktiverer manuell kontroll b. Operatør lander manuelt		

Test-ID:	T009	Relaterte krav:	K010
Hva testes:	Brukeren av systemet skal kunne endre den planlagte ruten under flygning.		
Optimalt scenario:	1. Operatør starer opp bakkestasjon og flygende plattform 2. Operatør planlegger og initierer et oppdrag 3. Systemet påbegynner utførelse av oppdraget 4. Operatør planlegger og initierer endringer i oppdraget 5. Systemet utfører oppdraget i henhold til de nye endringen		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke planlegging av oppdrag a. Test feilet 3a. Autonom takeoff/landing er ikke implementert a. Operatør tar av manuelt b. Operatør lander manuelt 3a. Autonom flygning etter waypoints er ikke implementert a. Test feilet 4a. Systemet aksepterer ikke endringer i plan a. Test feilet 4b. Systemet aksepterer ikke å initiere endringer i plan a. Test feilet 5. Systemet gjør ikke endringer i oppdraget a. Test feilet		

Test-ID:	T010	*Relaterte krav:*	K011
Hva testes:	Brukeren av systemet skal kunne laste ned video og bilder under flygning.		
Optimalt scenario:	1. SD-kort lastes med video og bilder 2. Operatør starer opp bakkestasjon og flygende plattform, inkludert Wi-Fi tilkobling mellom bakkestasjon og BeagleBoard 3. Operatør planlegger og initierer et oppdrag uten videoopptak eller billedtakning 4. Systemet starter oppdraget 5. Operatør initierer lasting av bilder fra plattformen til bakkestasjon 6. Operatør initierer lasting av video fra plattformen til bakkestasjon 7. Systemet avslutter oppdraget 8. Operatøren inspiserer nedlastede data		
Utvidet scenario:	2a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 2b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2c. Wi-Fi link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 3. Plattformen aksepterer ikke oppdrag a. Operatør aktiverer manuell kontroll b. Testen gjennomføres uten autonom flygning 4. Autonom takeoff er ikke implementert a. Operatør aktiverer manuell kontroll b. Takeoff utføres manuelt 5. Bilder kan ikke lastes ned a. Test feilet 6. Video kan ikke lastes ned a. Test feilet 7. Autonom landing er ikke implementert a. Operatør aktiverer manuell kontroll b. Landing utføres manuelt 8. Autonom flygning er ikke implementert a. Operatør gjennomfører testen med manuell kontroll 9. Data er ikke til stede eller er korrupte a. Test feilet		

Test-ID:	T011	*Relaterte krav:*	K012
Hva testes:	Video fra den flygende plattformen skal kunne streames til bakkestasjon i sanntid under flygning.		
Optimalt scenario:	1. Operatør starer opp bakkestasjon og flygende plattform, inkludert Wi-Fi tilkobling mellom bakkestasjon og BeagleBoard 2. Operatør planlegger og initierer et oppdrag uten videoopptak 3. Systemet iverksetter oppdraget 4. Operatør initierer streaming av video 5. Systemet streamer video fra flygende plattform til bakkestasjon 6. Operatør avslutter videostrøm 7. Systemet avslutter oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 1c. Wi-Fi link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke planlegging av oppdrag a. Operatør initierer manuell kontroll b. Resten av testen utføres med manuell kontroll 3. Autonom takeoff er ikke implementert a. Operatør initierer manuell kontroll b. Operatør tar av manuelt 4. Streaming av video ikke mulig å initiere a. Test feilet 5. Streaming avbrytes uanmeldt a. Test feilet 6. Streaming kan ikke avbrytes av operatøren a. Oppdrag avbrytes b. Test avbrytes med feilkode 7. Autonom landing er ikke implementert a. Operatør initierer manuell kontroll b. Operatør lander manuelt 8. Autonom flygning er ikke implementert a. Operatør gjennomfører testen med manuell kontroll		

Test-ID:	T012	*Relaterte krav:*	K013
Hva testes:	Det skal være mulig å lære seg systemet på under 60 minutter.		
Optimalt scenario:	1. Testperson er en person med den bakgrunnen man kan forvente at systemets fremtidige brukere vil ha, for eksempel yrkesmilitær med gode basiskunnskaper om blant annet militære kartsystemer 2. Instruktør er en person med god kjennskap til systemet 3. Testperson veiledes av instruktør. De følger en strukturert plan/opplæringsmanual for systemet.		
Utvidet scenario:	3a. Opplæringen tar mer enn en time a. Test feilet 3b. Testpersonen er ikke i stand til å benytte systemet selvstendig etter en times opplæring a. Test feilet		

Test-ID:	T015	**Relaterte krav:**	K010
Hva testes:	Brukeren av systemet skal kunne endre planlagt videoopptak og billedtakning under flygning.		
Optimalt scenario:	1. Operatør starer opp bakkestasjon og flygende plattform 2. Operatør planlegger og initierer et oppdrag med billedtakning og videoopptak 3. Systemet påbegynner utførelse av oppdraget 4. Operatør planlegger og initierer endringer i billedtakning og videoopptak 5. Systemet utfører oppdraget i henhold til de nye endringene 6. Operatør inspiserer bilder og video, kontrollerer at de er i henhold til siste oppdatering av oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke planlegging av oppdrag a. Test feilet 3a. Autonom takeoff/landing er ikke implementert a. Operatør tar av manuelt b. Operatør lander manuelt 3b. Autonom flygning etter waypoints er ikke implementert a. Test feilet 4a. Systemet aksepterer ikke endringer i plan a. Test feilet 4b. Systemet aksepterer ikke å initiere endringer i plan a. Test feilet 5. Systemet gjør ikke endringer i oppdraget a. Test feilet 6. Bilder og video er tatt som beskrevet i det originale oppdraget a. Test feilet		

Test-ID:	T016	Relaterte krav:	K007
Hva testes:	Fartøyet skal kunne lande uten menneskelig interaksjon.		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer et enkelt oppdrag (ta av og lande) 3. Systemet utfører oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke oppdrag a. Test feilet 3a. Systemet tar ikke av autonomt a. Operatør aktiverer manuell kontroll b. Operatør tar av manuelt 3b. Systemet lander ikke autonomt a. Test feilet		

Test-ID:	T017	Relaterte krav:	K006
Hva testes:	Bilder skal kunne lagres om bord på fartøyet.		
Optimalt scenario:	**Test i systemkonfigurasjon:** 1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør initierer oppdrag med billedtakning, et enkelt scenario der plattformen hovrer er tilstrekkelig 3. Systemet utfører oppdraget 4. SD-kort inspiseres for lagrede bilder **Forenklet test på komponentnivå:** 1. Koble til BeagleBoard 2. Aktiver protokoll for lagring av bilder 3. Komponentene tar en bildeserie og lagrer bildene 4. SD-kort inspiseres for lagret bilder		
Utvidet scenario:	**Test i systemkonfigurasjon:** 1a. Bakkestasjon og/eller flygende plattform starter ikke a. Operatør kjører forenklet test på komponentnivå 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Operatør kjører forenklet test på komponentnivå 2a. Oppdraget lar seg ikke initiere a. Operatør kjører forenklet test på komponentnivå 3. Autonom takeoff/flygning/landing er ikke implementert a. Operatør initierer manuell kontroll b. Operatør gjør manuell styring der det trengs 4. SD-kort inneholder korrupt eller ingen data a. Test feilet **Forenklet test på komponentnivå:** 1. Tilkobling til BeagleBoard feiler a. Test feilet 2a. Kobling mellom BeagleBoard og kamera kan ikke opprettes a. Test feilet 2b. Protokoll for lagring av bilder ikke tilgjengelig a. Test feilet 3. Komponentene tar ikke bilder a. Test feilet 4. SD-kort inneholder korrupt eller ingen data a. Test feilet		

Test-ID:	T018	Relaterte krav:	K014
Hva testes:	Systemoperatøren skal kunne planlegge oppdrag. Dette inkluderer: 1. Sette inn koordinater i både lat/long og MGRS		
Optimalt scenario:	1. Operatør starter opp bakkestasjon og flygende plattform 2. Operatør programmerer en rute gjennom de kjente, lett gjenkjennelige punktene A, B og C 3. Operatøren setter inn koordinater for de kjente punktene i både lat/long og MGRS formater 4. Systemet utfører det planlagte oppdraget 5. Operatøren observerer og vurderer utførelsen av oppdraget		
Utvidet scenario:	1a. Bakkestasjon og/eller flygende plattform starter ikke a. Test feilet 1b. Radiolink og/eller RC-link mellom bakkestasjon og den flygende plattformen kan ikke opprettes a. Test feilet 2. Systemet aksepterer ikke Waypoints a. Test feilet 3a. Systemet aksepterer ikke koordinater i lat/long a. Test feilet 3b. Systemet aksepterer ikke koordinater i MGRS a. Test feilet 4. Autonom takeoff og/eller landing er ikke implementert a. Operatøren tar av manuelt b. Operatøren lander manuelt 5. Systemet utfører ikke det planlagte/programmerte oppdraget a. Test feilet		

7.2 Prosjekttester

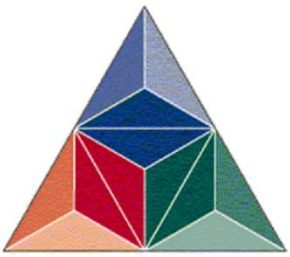
Test-ID:	T013	**Relaterte krav:**	K008
Hva testes:	Budsjettet skal ikke overstige 15 000 NOK.		
Optimalt scenario:	1. Et komplett regnskap over alle prosjektets godkjente utgifter settes opp og oppdateres underveis 2. Regnskapet sammenliknes med prosjektets budsjett Om prosjektet går over budsjett skal dette klareres med oppdragsgiver før det inntreffer.		
Utvidet scenario:	1a. Utlegg mangler kvittering a. Kostnaden dekkes av prosjektgruppa 1b. Utlegg er av den art at det ikke dekkes av oppdragsgiver a. Kostnaden dekkes av prosjektgruppa 2. Totale kostnader overstiger budsjett a. Test feilet		
Alternativt scenario:	I punkt 1 kan prosjektgruppa oppdage at visse kostnader vil bringe prosjektet over budsjett. Det kan da i dialog med oppdragsgiver vurderes å utvide budsjettet.		

Test-ID:	T014	**Relaterte krav:**	K009
Hva testes:	All programvare prosjektet produserer skal dokumenteres med doxygen, javadoc eller lignende.		
Optimalt scenario:	1. Automatisert dokumentasjon produseres på grunnlag av ferdig kode 2. Dokumentasjonen skal så gjennomgås av en testperson med bakgrunn innen det dokumenteringsverktøyet som brukes, med det formål å avdekke feil og mangler		
Utvidet scenario:	1. Automatisert dokumentasjon lar seg ikke produsere a. Test feilet 2. Testperson avdekker grove feil og mangler a. Test feilet		
Alternativt scenario:	I punkt 2 kan man gå gjennom koden og kontrollere kommentering fremfor å gå gjennom selve dokumentasjonen. I punkt 2 kan man også vurdere å skrive / benytte et program som gjør en systematisk kontroll.		

8 Referanser

1. Wikipedia. *TDD*. 28.11.2012]; Available from: http://en.wikipedia.org/wiki/Test-driven_development.
2. Oracle. *Programming with Assertions*. 06.01.2013]; Available from: http://docs.oracle.com/javase/1.4.2/docs/guide/lang/assert.html.
3. Codeproject. *Design by contract*. 11.01.2013]; Available from: http://www.codeproject.com/Articles/1863/Design-by-Contract-Framework.
4. Wikipedia. *Code Coverage*. 30.12.2012]; Available from: http://en.wikipedia.org/wiki/Code_coverage.
5. Wikipedia. *xUnit*. 19.12.12]; Available from: http://en.wikipedia.org/wiki/XUnit.
6. Sourceforge. *JUnit 4*. 06.01.2013]; Available from: http://junit.sourceforge.net/.

KONGSBERG



HØGSKOLEN
i Buskerud

S.T.A.R.

System for Tactical Aerial Reconnaissance

Prosjektgruppe:	S.T.A.R. – System for Tactical Aerial Reconnaissance
Dokumentnavn:	Testrapport
Dokumenttype:	Testdokument
Dokument ID:	TR001
Versjonsnummer:	V1.0
Versjonsdato:	23.05.2013
Graderingsnivå:	Ugradert

1 Innhold

2 Versjonshistorie

Versjonsnummer	Versjonsdato	Endringer
V1.0	23.05.13	Første versjon

3 Introduksjon

Dette dokumentet vil gå igjennom planlagte tester og dokumentere resultatet av gjennomførte tester. Testene beskrevet i *P004 – Testspesifikasjon* skal kunne verifisere at kravene i *P003 – Kravspesifikasjon* er oppfylt. Dokumentet vil inneholde siste utførte test. Hvis en test utføres flere ganger vil tidligere resultater vær tilgjengelige i tidligere versjoner av dokumentet.

3.1 Eksempelskjema

Test-ID:	*T001*	**Relaterte krav:**	*K001*
Testdato:	*Dato*	**Testperson:**	*Navn på person(er) som utfører testen.*
Hva testes:	Beskriver hva som skal testes		
Utført Scenario	*1. Første steg* *2. Videre instruksjoner* *3. Avslutning*		
Resultat:	*1. Hva var vellykket* *2. Hva feilet* *3. Test bestått/ikke bestått*		

4 Resultater

4.1 Systemtester

Test-ID:	T001	**Relaterte krav*:***	K001
Testdato:	23.05.13	**Testperson:**	Jørgen Markussen
Hva testes:	Bilder skal inneholde geografiske referanser og tidsmerking		
Utført Scenario	1 til 5		
Resultat:	Korrekt merking ikke på plass. **Test feilet.**		

Test-ID:	T002	**Relaterte krav*:***	K002
Testdato:	21.05.13	**Testperson:**	Liam Jensrud
Hva testes:	Video skal inneholde geografiske referanser og tidsmerking.		
Utført Scenario	1 til 5		
Resultat:	NATO-standard for merking til video er ikke implementert Filer er ikke tilgjengelig. Korrekt merking ikke på plass. **Test feilet**		

Test-ID:	T003	**Relaterte krav*:***	K003
Testdato:	19.05.13	**Testperson:**	Gunnar Sandaker, Jørgen Markussen
Hva testes:	Systemoperatøren skal kunne planlegge oppdrag. Dette inkluderer: 1. Opprette rute der fartøyet skal fly i form av veipunkter 2. Definere operasjonshøyde for veipunkter		
Utført Scenario	1 til 5		
Resultat:	Bakkestasjonen aksepterer oppdrag, oppdrag overføres til flygende plattform. Flygende plattform ikke i stand til å utføre oppdrag, pga manglende egenskaper i autonom tilstand. **Test feilet.**		

Test-ID:	T004	**Relaterte krav*:***	K003
Testdato:	22.05.13	**Testperson:**	Liam Jensrud
Hva testes:	Systemoperatøren skal kunne planlegge oppdrag, herunder hvor fartøyet skal samle informasjon.		
Utført Scenario	1 til 3		
Resultat:	Instruks om videoopptak ved veipunkter er ikke implementert **Test feilet**		

Test-ID:	T005	**Relaterte krav*:***	K004
Testdato:	19.05.13	**Testperson:**	*Gunnar Sandaker*
Hva testes:	Systemet skal melde fra til systemoperatøren dersom et planlagt oppdrag ikke er mulig på grunn av systemets/fartøyets begrensninger.		
Utført Scenario	1 til 4.		
Resultat:	**Test bestått.** **Merknad1:** Fungerer kun i NED koordinater. **Merknad2:** Tar ikke hensyn til batterinivå. Operatør velger kapasitet i minutter.		

Test-ID:	T006	**Relaterte krav*:***	K005
Testdato:	20.05.13	**Testperson:**	Gunnar Sandaker
Hva testes:	Fartøyet skal kunne fly i minst 15 minutter.		
Utført Scenario	Fløy i ca. 10,5 minutter, med ny quad-kropp.		
Resultat:	**Test Feilet**		

Test-ID:	T007	**Relaterte krav*:***	K006
Testdato:	23.05.13	**Testperson:**	Jørgen Markussen
Hva testes:	Video skal kunne lagres om bord på fartøyet.		
Utført Scenario	Forenklet test, 1 til 4.		
Resultat:	Lagringsenhet inneholder korrupt eller ingen data. **Test feilet.**		

Test-ID:	T008	**Relaterte krav*:***	K007
Testdato:	22.05.13	**Testperson:**	Liam Jensrud
Hva testes:	Fartøyet skal kunne ta av uten menneskelig interaksjon.		
Utført Scenario	1 til 3		
Resultat:	Automatisk take off er ikke implementert. **Test feilet**		

Test-ID:	T009	**Relaterte krav*:***	K010
Testdato:	22.05.13	**Testperson*:***	Liam Jensrud
Hva testes:	Brukeren av systemet skal kunne endre den planlagte ruten under flygning.		
Utført Scenario	1 til 3a		
Resultat:	Autonom flygning etter veipunkter er ikke implementert **Test feilet**		

Test-ID:	T010	**Relaterte krav*:***	K011
Testdato:	23.05.13	**Testperson*:***	Jørgen Markussen
Hva testes:	Brukeren av systemet skal kunne laste ned video og bilder under flygning.		
Utført Scenario	1 til 5		
Resultat:	Bilder kan ikke lastes ned. **Test feilet.**		

Test-ID:	T011	**Relaterte krav*:***	K012
Testdato:	23.05.13	**Testperson*:***	Jørgen Markussen
Hva testes:	Video fra den flygende plattformen skal kunne streames til bakkestasjon i sanntid under flygning.		
Utført Scenario	1 til 4		
Resultat:	Streaming av video ikke mulig å initiere. **Test feilet.**		

Test-ID:	T012	**Relaterte krav*:***	K013
Testdato:	23.05.13	**Testperson*:***	
Hva testes:	Det skal være mulig å lære seg systemet på under 60 minutter.		
Utført Scenario	Ikke utført.		
Resultat:			

Test-ID:	T015	**Relaterte krav*:***	K010
Testdato:	21.05	**Testperson*:***	Liam Jensrud
Hva testes:	Brukeren av systemet skal kunne endre planlagt videoopptak og billedtakning under flygning.		
Utført Scenario	1 til 3a		
Resultat:	Autonom flygning etter veipunkter er ikke implementert. **Test feilet**		

Test-ID:	T016	**Relaterte krav*:***	K007
Testdato:	21.05	**Testperson**	Liam Jensrud
Hva testes:	Fartøyet skal kunne lande uten menneskelig interaksjon.		
Utført Scenario	1 til 3b		
Resultat:	Automatisk landing er ikke implementert. **Test feilet**		

Test-ID:	T017	**Relaterte krav*:***	K006
Testdato:	19.05	**Testperson:**	Liam
Hva testes:	Bilder skal kunne lagres om bord på fartøyet.		
Utført Scenario	1 til 2a, 3a, 4		
Resultat:	Bilder fra oppdrag ble tatt og lagret om bord på fartøyet. Eksponeringen til bildene var for høy. **Test bestått.**		

Test-ID:	T018	Relaterte krav:	K006
Testdato:	22.05	Testperson:	Liam Jensrud
Hva testes:	Systemoperatøren skal kunne planlegge oppdrag. Dette inkluderer: Sette inn koordinater i både lat/long og MGRS		
Utført Scenario	1 til 3b		
Resultat:	Systemet aksepterer ikke koordinater i MGRS. **Test feilet**		

4.2 Prosjekttester

Test-ID:	T013	Relaterte krav:	K008
Testdato:	22.05.13	Testperson:	Liam Jensrud
Hva testes:	Budsjettet skal ikke overstige 15 000 NOK.		
Utført Scenario	1 til 2a		
Resultat:	Det totale budsjettet oversteg opprinnelige budsjettet på 15 000 NOK **Test feilet**		

Test-ID:	T014	Relaterte krav:	K009
Testdato:	23.05.13	Testperson:	Jørgen Markussen
Hva testes:	All programvare prosjektet produserer skal dokumenteres med doxygen, javadoc eller lignende.		
Utført Scenario	1 til 2		
Resultat:	Kildekode er dokumenter på doxygenformat. **Test bestått.**		