

AARHUS UNIVERSITY SCHOOL OF ENGINEERING

AUTONOM FORFØLGELSESDRONE MED SMARTPHONE APPLIKATION

BACHELORPROJEKT
ELEKTROINGENIØR

Projektrapport

Projekt nr: 16114

Jonas Risager Nielsen - 201270337

Benedikt Wiese - 201310362

Mathias Poulsen - 201370945

Vejleder

Torben Gregersen

Aarhus University School of Engineering

15. december 2016

Versionshistorie for Projektrapporten

Version	Dato	Beskrivelse
0.0	28.11.2016	Opsætning af dokumentet.
0.1	28.11.2016	Tilføjet Forord, Indledning, Krav, Afgrænsning, Metode.
0.2	29.11.2016	Tilføjet Arkitektur, Design, Implementering, Test.
0.3	07.12.2016	Ordforklaring tilføjet.
0.4	08.12.2016	Tilføjet Resume, Abstract.
0.4.1	09.12.2016	Tilføjet Resultater, Diskussion af resultater, Konklusion.
0.5	12.12.2016	Rettet efter vejleder møde.
0.5.1	14.12.2016	Revideret.
1.0	15.12.2016	Aflevering.

Ordforklaring

Forkortelse	Forklaring
3G, ISP, Internet	Disse har samme betydning i dette dokument. De bliver alle brugt til at beskrive internetforbindelsen.
Ω	Ohm.
A	Ampere.
API	Application Programming Interface.
ARM	Advanced RISC Machine.
ASE	Aarhus University School of Engineering.
BDD	Block Definition Diagram.
CD	Class Diagram.
CPU	Central Processing Unit.
DM	Domain Model.
FURPS-analyse	Functionality, Usability, Reliability, Performance og Supportability.
GPIO	General Purpose Input/Output.
GPS	Global Positioning System.
HTTP	HyperText Transfer Protocol.
IBD	Internal Block Diagram.
ID	Identifikation.
IDE	Integrated Development Environment.
iOS	Iphone Operating System.
JSON	JavaScript Object Notation.
LED	Light Emitting Diode.
MoSCoW-analyse	Must, Should, Could og Would.
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor.
PWM	Pulse Width Modulation.
RGB	Red, Green, Blue.

RPi	Raspberry Pi 3 model B.
Shadow BAC-X1	Navnet på dronen, der udvikles i dette projekt.
SysML	System Modeling Language.
TCP	Transmission Control Protocol.
UART	Universal Asynchronous Receiver/Transmitter.
UC	Use Case.
UI	User Interface.
UML	Unified Modeling Language.
USB	Universal Serial Bus.
XML	Extensible Markup Language.

Indholdsfortegnelse

Kapitel 1 Indledning	3
Kapitel 2 Krav	5
2.1 Aktør kontekst diagram	5
2.2 Use Case's	6
2.3 Ikke-funktionelle krav	8
2.4 Prioritering	8
2.5 Afgrænsning	9
Kapitel 3 Realisering	10
3.1 Metode	10
3.1.1 Domænemodel	10
3.1.2 SysML	10
3.1.3 UML	10
3.1.4 N+1	11
3.2 Analyse	12
3.2.1 FlightController	12
3.2.2 DroneControllerUnit	12
3.2.3 Trådsløs Kommunikationen	12
3.2.4 Styresystem smartphone	13
3.3 Arkitektur	14
3.3.1 Hardware	15
3.3.2 Software	16
3.4 Design	23
3.4.1 Hardware	23
3.4.2 Software	24
3.5 Implementering	29
3.5.1 Hardware	29
3.5.2 Software	30
3.6 Test	31
3.6.1 Modultest	31
3.6.2 Integrationstest	32
3.6.3 Acceptttest	32
Kapitel 4 Resultater	34
Kapitel 5 Diskussion af resultater	35
Kapitel 6 Fremtidigt Arbejde	37
Kapitel 7 Konklusion	39

Abstract

The project was carried out in autumn 2016. The aim has been to develop an autonomous drone that could follow and film a user. The drone was controlled directly from the associated smartphone application. The drone developed in this project is named Shadow BAC-X1.

The system was limited to function only with a single drone and a single user. Communication happened either through the Internet or Bluetooth. The security of the communication was down prioritized in the project. Therefore the communication was not protected against unauthorized use.

The results produced in this project, was a drone that communicates with the smartphone application through Internet or Bluetooth. The drone was controlled through the smartphone application and maneuvered according to GPS coordinates.

Based on the project requirements a functionel prototype had succesfully been developed. The prototype had been developed and tested in a controlled enviroment and has not been tested in the air.

Resume

Projektet blev udarbejdet i efteråret 2016. Formålet med projektet har været at udvikle en autonom drone, som kunne følge og filme en bruger. Dronen blev styret direkte fra den tilhørende smartphone applikation. Dronen, der blev udviklet i dette projekt, havde navnet Shadow BAC-X1.

Systemet blev afgrænset til udelukkende at fungere med én drone og én bruger. Kommunikationen foregik enten via internettet eller Bluetooth. Sikkerheden i kommunikationen var nedprioriteret i projektet. Derfor var kommunikation ikke beskyttet mod uautoriseret brug.

Der var igennem projektet udviklet en drone, der kommunikerede med applikationen via internettet eller Bluetooth. Dronen blev styret gennem smartphone applikationen og manøvrerede efter GPS koordinater.

Der blev udviklet en fungerende prototype jævnfør projektets kravspecifikation. Prototypen blev udviklet og testet i et kontrolleret miljø og har ikke været afprøvet i luften.

Forord

Denne rapport omhandler bachelorprojektet *Autonom forfølgelses drone med smartphone applikation*, udarbejdet på Aarhus University School of Engineering. Rapporten blev udarbejdet af Mathias Poulsen, Benedikt Wiese og Jonas R. Nielsen. Under projektforløbet var alle tre studerende på diplomingeniøruddannelsen: Elektronikingeniør. Projektoplægget blev udarbejdet af gruppens medlemmer.

Tilhørende til denne projektrapport er en procesrapport samt bilag. Procesrapporten ligger umiddelbart efter projektrapporten og omhandler de arbejdsprocesser, der blev brugt i løbet af projektet. Derudover er der vedlagt en række bilag, der indeholder blandt andet kravspecifikationen, analyse, systemarkitektur og -design, samt de tests der er udført på systemet. Disse giver en dybere forståelse for systemet og bliver refereret til i dette dokument.

Lektor Torben Gregersen var vejleder gennem projektet. Projektet blev afleveret den 16. december 2016, og bliver bedømt ved en mundtlig eksamen den 13. januar 2017.

Stor tak til Torben Gregersen for vejledning gennem projektforløbet og tak til familiemedlemmer for korrekturlæsning.

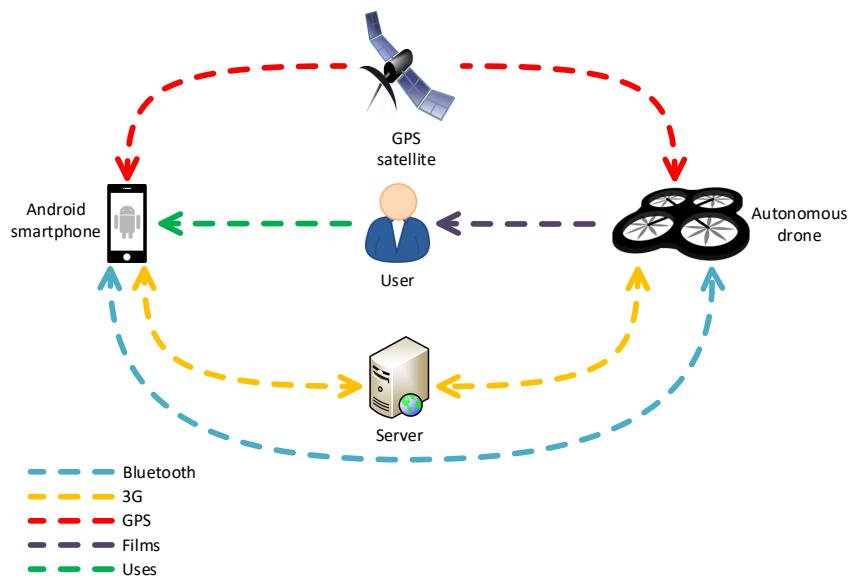
Indledning 1

I løbet af de seneste år er der sket en stor udvikling indenfor robotter og droner. Det er specielt enheder, der fungerer autonomt, altså uden menneskelig interaktion, der er under udvikling. Dette ses blandt andet hos Amazon. Amazon er igang med at udvikle et autonomt pakkeudleveringssystem baseret på droner [1].

Der er et stort potentiale i droner, der kan flyve autonomt [2]. Det er på baggrund af netop dette, at vi har valgt at udvikle en autonom drone i projektet.

Gruppen overvejede dronens formål inden projektets start. Det blev valgt at udvikle en prototype af en drone, der er udstyret med et kamera, som kan følge og filme en bruger. Dronen skal følge brugeren autonomt baseret på brugerens GPS-koordinat. Hele systemet skal bestå af en applikation til en Android mobiltelefon, en server og dronen selv. Ved hjælp af smartphone applikationen kan brugeren angive en relativ position for dronen, og dronen vil så holde denne position i forhold til brugeren gennem hele flyvesessionen.

Systemet benytter GPS til positionering og både Bluetooth og 3G til den trådløse kommunikation. En oversigt over systemet ses på figur 1.1. Kommunikationen vil hovedsageligt foregå over 3G, såfremt dette er tilgængeligt. Hvis 3G ikke er tilgængeligt, vil Bluetooth blive benyttet.



Figur 1.1: Systemets oversigt

Dette er ikke et nyt koncept, da et stort antal droner allerede har "follow-funktionaliteten". Et eksempel på en drone, der har en lignende funktionalitet, er dronen Hexo+. Denne drone kan autonomt følge og filme brugeren. Dronen styres vha. af en smartphone applikation [3].

Ansvarsområder

Tabel 1.1 viser en overordnet opdeling af ansvarsområderne i projektet i forhold til produktet. Kravspecifikationen og systemarkitekturen for hele projektet er lavet i fællesskab, og opdelingen er derfor først sket under design- og implementeringenfasen. Tabel 1.2 viser en mere detaljeret opdeling af ansvarsområderne for dronen.

Område	Navne
Applikation til smartphone	Benedikt
Serveropsætning	Jonas
Drone	Jonas & Mathias

Tabel 1.1: Produktmæssigt hovedansvarsområde

Klasse	Navne
BtHandler	Mathias
Communicator	Mathias
DataHandler	Mathias & Jonas
DroneController	Mathias
GPSHandler	Jonas
Interface	Mathias
ISPHandler	Mathias
SerialArduinoHandler	Jonas
SonarHandler	Mathias
StepMotorHandler	Mathias
DroneClassTester	Jonas
AeroQuad opsætning	Jonas
Raspberry Pi opsætning	Mathias

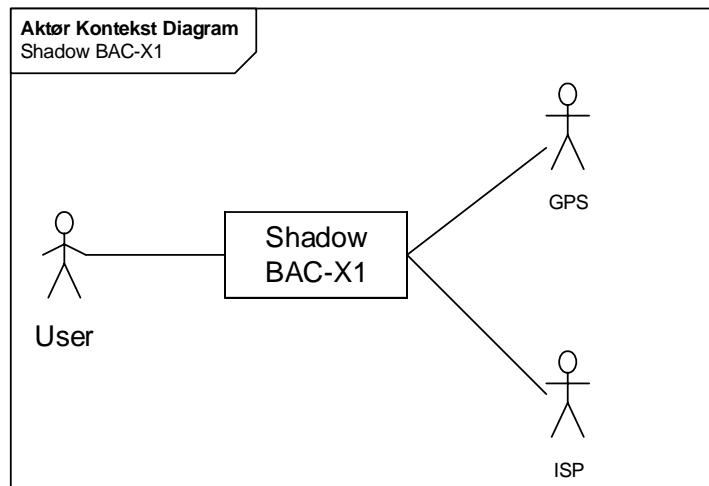
Tabel 1.2: Ansvarsområder for dronen

Krav 2

Dette afsnit beskriver systemets krav baseret på et aktør kontekst diagram, Use Cases og ikke-funktionelle krav.

2.1 Aktør kontekst diagram

For at identificere hvilke primære og sekundære aktører, der har indflydelse på systemet, blev et aktør kontekst diagram opstillet. Dette ses på figur 2.1.



Figur 2.1: Aktør kontekst diagram

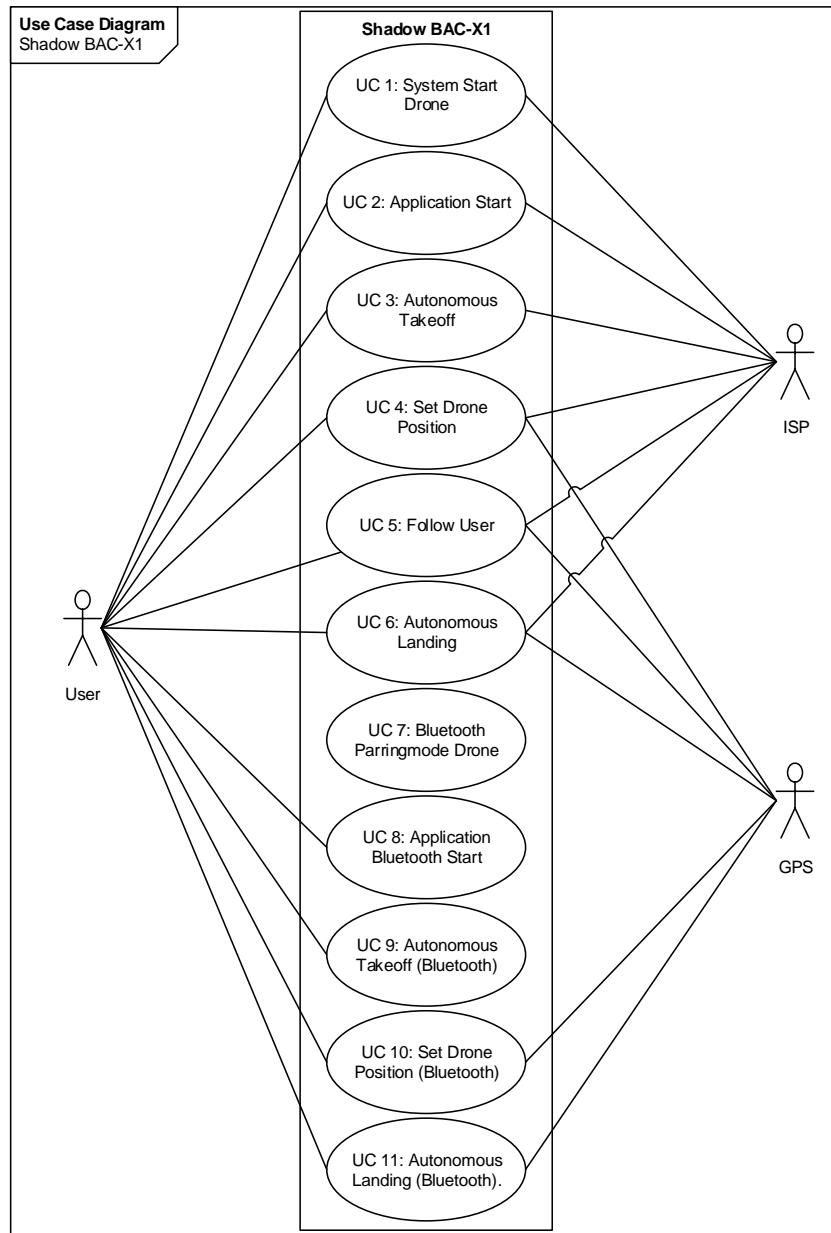
På figur 2.1 ses brugeren til venstre for det samlede system som den primære aktør. Brugeren benytter systemet, der bliver vist på diagrammet som en blackbox. På højre side, ses de sekundære aktører. Disse aktører bliver brugt af systemet. GPS og ISP bliver defineret som sekundære aktører, da disse er teknologier, der benyttes men fysisk ligger uden for systemet Shadow BAC-X1.

2.2 Use Case's

I følgende afsnit vil Use Case diagrammet, samt de enkelte Use Case's kort blive beskrevet. En fuldstændig beskrivelse kan findes i afsnit 1.7 i Kravspecifikationen.

Use Case diagram

Figur 2.2 viser systemets Use Case diagram. I dette diagram er systemet lukket op, og funktionalitet beskrevet gennem en række Use Case's.



Figur 2.2: Use Case diagram

Systemet har stadig de samme aktører, som på figur 2.1. Det kan dog ses, hvilke Use Case's

der bliver startet af brugeren, og hvilke der benytter ISP og GPS. Det ses, at Use Case 7 ikke tilgås udefra. Dette er fordi, Use Case 7 initialiseres af dronen selv og ikke af brugeren.

Når brugeren ikke har forbindelse til internettet eller ikke ønsker at bruge data, har brugeren mulighed for at forbinde til dronen via Bluetooth. Systemets funktionalitet er ikke den samme, når der bruges Bluetooth i forhold til ISP. Dronen kan kun følge brugeren, når applikationen og dronen er forbundet via ISP.

Use Case - 1: System Start Drone

Denne Use Case håndterer dronens opstartsprocedure. Her initialiserer dronen de nødvendige moduler og venter på et Start-signal fra applikationen. Når dette er modtaget, svarer dronen med en acknowledgement og begynder herefter at sende status beskeder til applikationen.

Use Case - 2: Application Start

Use Case 2 er applikationens opstart. Denne Use Case forudsætter, at applikationen forbindes til dronen gennem 3G. Applikationen sender et Start-signal til serveren og venter på dronens acknowledgement. Når dette er modtaget, opdaterer applikationen sit interface med data fra status beskederne.

Use Case - 3: Autonomous Takeoff

Dronen modtager i denne Use Case et TakeOff-signal fra applikationen. Herefter letter dronen autonomt fra jorden og holder højden, som brugeren har indstillet gennem smartphone applikationens interface, inden brugeren har trykket på TakeOff knappen.

Use Case - 4: Set Drone Position

Use Case 4 benyttes, når brugeren ønsker dronen i en specifik position. Brugeren vælger værdier gennem applikationens interface og sender disse til dronen. Dronen udregner sin nye position og nавигerer til denne autonomt. Kameraet indstilles således, dette er rettet mod brugeren.

Use Case - 5: Follow User

Når dronen modtager et Follow-signal, begynder denne at følge applikationens GPS koordinat baseret på Follow-signalerne. Når dronen ikke skal følge brugeren mere, sendes et Unfollow-signal til dronen.

Use Case - 6: Autonomous Landing

Dronen modtager et Land-signal fra applikationen. Den begynder herefter autonomt at lande.

Use Case - 7: Bluetooth Parringmode Drone

Denne Use Case benyttes af dronen, når den ikke kan få forbindelse til applikationen over internettet. Dronen starter selv denne Use Case, for at benytte Bluetooth i stedet for 3G.

Når dronen har accepteret en forbindelsesanmodning fra applikationen, begynder denne at sende status beskeder retur.

Use Case - 8: Application Bluetooth Start

Når applikationen skal skifte til Bluetooth, er det et aktiv valg fra brugerens side. Denne Use Case håndterer applikationens opsætning af Bluetooth, og at denne forbinder til dronen.

Use Case - 9: Autonomous Takeoff (Bluetooth)

Denne Use Case fungerer på samme måde, som Use Case 3. Her benyttes blot Bluetooth frem for 3G.

Use Case - 10: Set Drone Position (Bluetooth)

Denne Use Case fungerer på samme måde, som Use Case 4. Her benyttes blot Bluetooth frem for 3G.

Use Case - 11: Autonomous Landing (Bluetooth)

Denne Use Case fungerer på samme måde, som Use Case 6. Her benyttes blot Bluetooth frem for 3G.

2.3 Ikke-funktionelle krav

For at definere de tekniske krav, der stilles til systemet, blev der opstillet en række ikke-funktionelle krav. Disse krav beskriver egenskaber, der ikke har indflydelse på systemets Use Case's. Dette kunne være dronens hastighed eller præcisionen af de enkelte komponenter. Projektets ikke-funktionelle krav kan ses i afsnit 1.8 i Kravspecifikation i bilag.

2.4 Prioritering

For at vurdere, hvor vigtigt kravene indenfor de enkelte områder er, blev der lavet en FURPS-analyse[4] af systemet. I denne analyse blev systemet inddelt i fem punkter: Functionality, Usability, Reliability, Performance og Supportability.

For at skabe et overblik over, hvad der fokuseres på i projektet, blev der tildelt points fra 1-5 til de enkelte punkter. 1 er mindst- og 5 er mest relevant. Det blev valgt, at lægge stort fokus på at implementere funktionaliteterne beskrevet i Use Case'ene, og at udvikle et system, der var nemt at bruge. Hvorimod det blev nedprioriteret at fokusere på punkter som lavt strømforbrug og systemets skalerbarhed.

FURPS-analysen kan ses i afsnit 1.12 i Kravspecifikation i bilag.

2.5 Afgrænsning

For at afgrænse projektets funktionalitet og for at finde frem til Use Case'ene, blev en MoSCoW-analyse udført [5]. I denne analyse blev alle funktionaliteter, der ønskes i systemet, opstillet og dernæst prioriteret i de fire kategorier *must*, *should*, *could* og *would*. Det blev valgt, at definere systemets Use Case's ud fra analysens *must* krav. Derved blev disse krav implementeret i dette projekt.

Et punkt, der blev nedprioriteret i projektet, er sikkerhed mod uautoriseret brug fra andre personer. Det er på nuværende tidspunkt muligt, at styre systemets drone, hvis man kender til dronens unikke ID.

Der er en række love omkring brugen af droner, som projektet skal overholde. Det er brugerens eget ansvar udelukkende at benytte dronen lovligt. Det blev dog valgt, at implementere en maksimal højde på 50 meter, således at dronen med garanti overholder lovgivningens maksimale højde på 100 meter [6].

Systemet benyttede et GoPro kamera monteret på dronen. Dette kamera fungerede selvstændigt. I projektet var det kun rotationen af kameramodulet der blev implementeret. Kameraet kunne ikke styres fra systemets applikation.

Realisering 3

Afsnittet beskriver projektets udviklingsforløb. Derunder indgår de brugte metoder, samt vigtige dele af den udførte analyse og systemets arkitektur og -design. Afsluttende gøres der rede for implementeringen og de tests der blev udført på systemet.

3.1 Metode

Dette afsnit vil kort beskrive nogle af de forskellige modeller brugt igennem projektet.

3.1.1 Domænemodel

Til arkitekturen af softwaren blev der lavet en domænemodel over det samlede system og dronen. Ud fra disse domænemodeller og systemets Use Case's kunne systemets konceptuelle klasser findes. Disse konceptuelle klasser blev brugt videre i design processen i forbindelse med N+1 modellen.

3.1.2 SysML

Under arkitektur- og designprocesserne blev sproget SysML [7] benyttet. Dette kan især bruges til en visuel beskrivelse af systemer, der indeholder både hardware og software. I dette projekt blev SysML benyttet dog kun til hardwaredelen.

Anvendelsen af SysML-diagrammer er et godt værktøj til at visualisere systemets overordnede opbygning i form af f.eks. Use Case diagrammet. Derudover blev Use Case'ene benyttet til at skabe både et overblik samt et dybere indblik i systemets hardware i form af BDD og IBD.

3.1.3 UML

UML er en standard for diagrammer til beskrivelse af software i objektorienterede projekter [8].

Gennem dette projekt blev klassediagrammer og sekvensdiagrammer brugt. Klassediagrammet er et af de mest brugte UML diagrammer. Her vises systemets softwareklasser, og hvilken relation disse har til hinanden. Systemets klassediagrammer udarbejdes på baggrund af de funktioner, der fremgår af sekvensdiagrammerne.

Sekvensdiagrammer hører ind under begrebet adfærdsdiagrammer, som benyttes til at vise, hvordan systemets aktører og objekter arbejder sammen om at gennemfører et brugsmønster. I dette projekt blev sekvensdiagrammer benyttet til at forklare kommunikationen mellem de forskellige softwareblokke i en given Use Case.

Det blev valgt, at afvige fra UML standarden i klassediagrammerne. For at vise de nyfundne metoder igennem systemets Logical View, blev disse skrevet med sort skrift på de statiske klassediagrammer. Metoder, der er relevante for de tilsvarende Use Case's, men allerede er blevet identificeret tidligere, blev vist med grå skrift.

Et eksempel på brugen af UML fremgår i afsnit 3.3 Arkitektur.

3.1.4 N+1

N+1 er en model til udvikling af software. Denne model deler arkitektur- og designprocessen op i N antal View's [9]. Oprindeligt var dette en 4+1 View model. +1 delen står for systemets Use Case's, der betragtes som et View. Tallet 4 står for de resterende View's: Logical, Process, Deployment og Data. Modellen er efterhånden blevet videreført til en mere generel model ved navn N+1. N+1 muliggør anvendelsen af det antal View's, der er nødvendige for at beskrive arkitektur- og designdelen af softwaren i et projekt.

I dette projekt blev en 6+1 model benyttet. Systemet blev opstillet gennem Use Case's, og herefter blev følgende View's belyst i arkitektur- og designprocesserne: Logical, Process, Data, Deployment, Security og Implementation.

I Logical View opdeles systemet i CPU'er. Systemets Use Case's gennemgås for hver CPU, og de relevante konceptuelle klasser findes. På baggrund af disse udarbejdes et sekvensdiagram til hver Use Case. Fra hvert sekvensdiagram udledes et statisk klassediagram. Når denne proces er udført for hver Use Case, samles klassediagrammerne fra de enkelte Use Case's til ét klassediagram for hver CPU.

Proces View omhandler de forskellige processer i de enkelte CPU'er. Der gøres rede for interaktionen mellem processer på samme CPU. Her kan det også blyses nærmere, hvilke processer der er tråde, og hvilke der ikke er.

Data View belyser data strukturer og de persistente data i systemet. Dette kan være, hvordan en given CPU håndterer data omkring f.eks. GPS koordinater. Her kan ligeledes være forklaringer om databaser, logs og hvilken form for hukommelse dataet bliver gemt i.

Deployment View omhandler protokollerne mellem de forskellige CPU'er. Her gøres rede for de protokoller, som er udviklet eller anvendt i forbindelse med kommunikation imellem systemets CPU'er.

Security View omhandler den sikkerhed, som ligger i systemet. Dette kan være verificering af brugere eller kryptering af fortrolig data.

I Implementation View skal det forklares for læseren, hvordan systemet skal bygges og opsættes, såfremt kildekoden er tilgængelig. Dette afsnit beskriver derudover særlige tiltag i systemet.

Denne model blev benyttet til at opstille projektets softwarearkitektur, -design og -implementering. Modellen blev forbundet med den valgte iterative udviklingsproces, således at de enkelte View's løbende blev udvidet.

3.2 Analyse

I projektets start blev en analyse af systemets dele udarbejdet. I dette afsnit vil de vigtigste valg blive gennemgået. Projektets analyse kan findes i en mere detaljeret udgave i Analyse i bilag.

3.2.1 FlightController

Dronen, som benyttes i systemet, er produceret af AeroQuad [10]. AeroQuad har udviklet softwaren til dennes FlightController. FlightController'en har ansvaret for at holde dronen stabil i luften ved hjælp af sensorer, og at navigere dronen baseret på input fra fjernbetjeningen.

Fra AeroQuad er to mulige FlightController'e tilgængelige. En dedikeret FlightController baseret på en ARM processor, med tilhørende sensorer og nødvendige udgange eller en Arduino Mega 2560 med et tilhørende shield. I dette projekt blev det valgt at benytte Arduino Mega'en, da AeroQuad har nedlagt supporten til den dedikerede FlightController og derfor anbefaler at benytte en Arduino med det tilhørende shield.

3.2.2 DroneControllerUnit

For at dronen kan flyve autonomt skal fjernbetjeningens modtager erstattes med en Single Board Computer. Denne enhed kaldes DroneControllerUnit og skal kunne kommunikere gennem både 3G og Bluetooth. Denne skal ligeledes kunne generere et input til FlightController'en og tilgå GPS.

Til denne opgave blev følgende Single Board Computere analyseret:

- Beaglebone Black
- Arduino Mega 2560
- Raspberry Pi 3 Model B

Der blev valgt en Raspberry Pi 3. Fordelene ved denne er at det er en Linux maskine med en høj processerkraft. Raspberry Pi'en er multikernet og giver derved også muligheden for et trådet system, der kan køre parallelt. Ulempen ved Raspberry Pi'en i forhold til Arduino'en er, at den ikke har PWM udgange.

Dette er dog løst ved, at benytte en Arduino Nano i samarbejde med Raspberry Pi'en. En anden stor fordel ved at vælge Raspberry Pi'en er, at denne har et integreret Bluetooth modul, hvilket skal benyttes i projektet.

3.2.3 Trådløs Kommunikationen

Den trådløse kommunikation i systemet udgør grundstenen for hele projektet. I projektet blev Bluetooth, 3G og 433MHz overvejet. Projektet bygger på, at brugeren udelukkende benytter en mobiltelefon til at kunne styre systemet. Derved er 433MHz udelukket, da dette vil kræve ekstra hardware for at kunne fungere. En hybrid løsning blev valgt, for at sikre mod eventuelle situationer uden 3G dækning.

Dronen skal altså kunne benytte både 3G og Bluetooth. Det samme skal brugerens mobiltelefon.

For begge kommunikationensformer blev det valgt at understøtte retransmission for at sikre pålidelig kommunikation.

Da projektets enheder blev udviklet i tre forskellige programmeringssprog, var det vigtigt at pakkedata ikke gik tabt, grundet forskellig pakkehåndtering. Det blev derfor valgt, at benytte JSON-formatet [11] til at indkapsle pakkerne. Dette gav en ens håndtering af pakkerne gennem hele systemet.

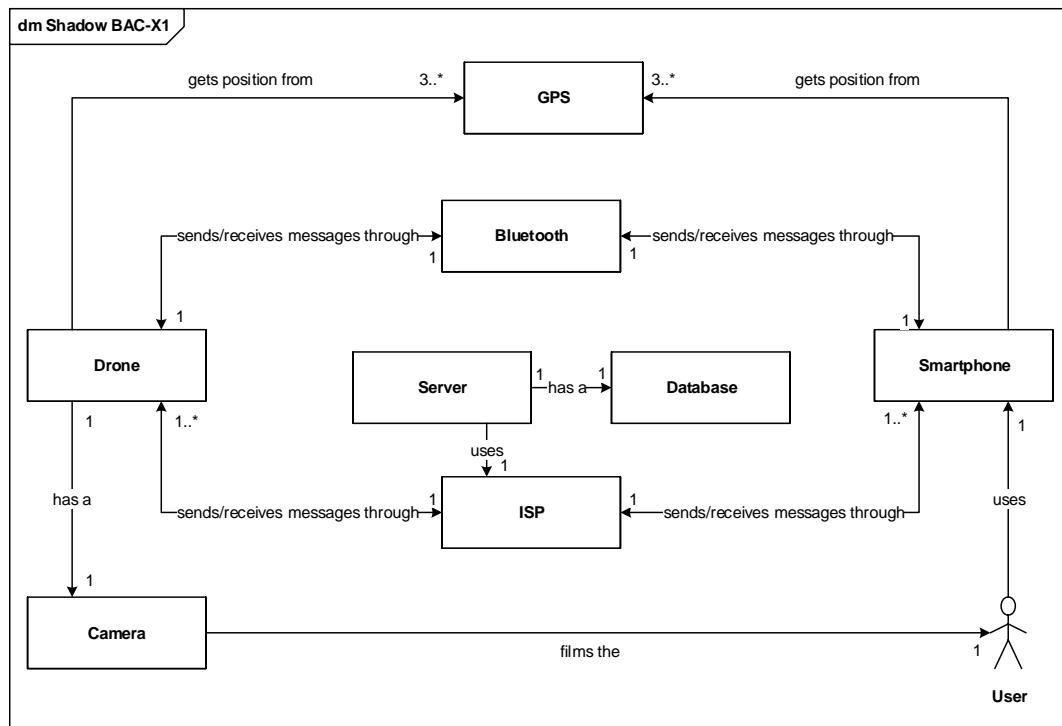
3.2.4 Styresystem smartphone

Udviklingen af applikationer til smartphones tilhører primært de to mest populære styresystemer iOS og Android. Grundet markedsandelen og udviklingsmiljøer blev det valgt, at implementere en Android applikation i projektet. iOS har en betydeligt mindre markedsandel og kræver en MAC computer for at udvikle applikationer.

3.3 Arkitektur

I dette afsnit beskrives, hvordan arkitekturen er udarbejdet gennem projektet. Arkitekturen fokuserer på systemets blokke og kommunikationen mellem disse. Dette afsnit er opdelt i hardware- og softwarearkitektur.

På figur 3.1 ses domænemodellen for det samlede system. Domænemodellen bruges til at skabe et overblik over systemet, og til at finde konceptuelle klasser senere i arkitekturen.



Figur 3.1: Domænemodel for det samlede system

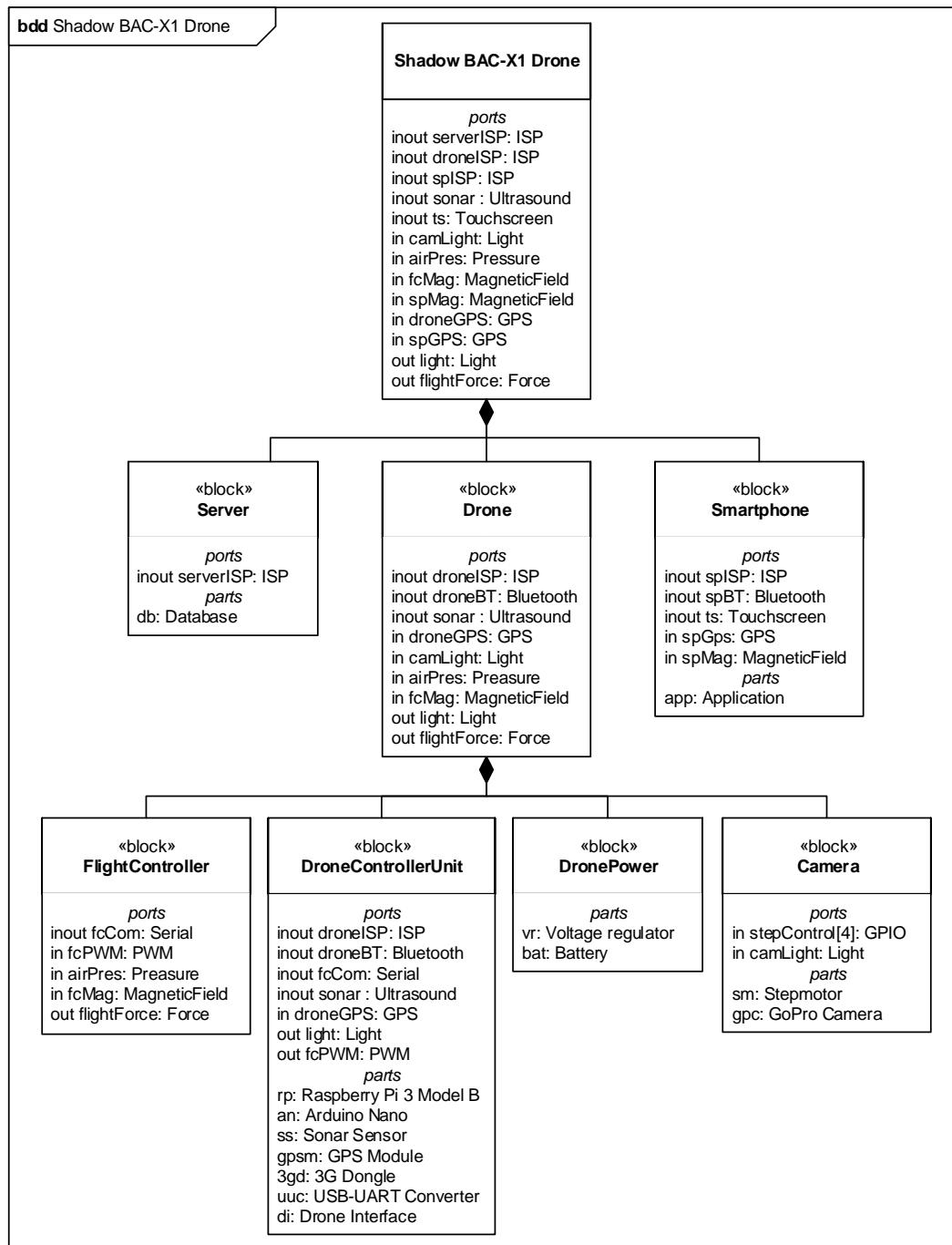
Figur 3.1 viser, at brugeren benytter systemets applikation på en smartphone. Via applikationen kan brugeren sende beskeder til dronen gennem enten ISP eller Bluetooth. Både dronen og applikationen skal tilgå GPS for at finde deres position. Serveren, som benyttes over ISP, har tilgang til en database, hvor denne gemmer beskederne. Dronen filmer brugeren via et GoPro kamera.

En domænemodel blev ligeledes udarbejdet for dronens interne blokke, denne kan findes i afsnit 1.1.2 i Arkitektur & Design i bilag.

3.3.1 Hardware

Dette afsnit vil kort beskrive projektets hardware arkitektur.

På figur 3.2 ses systemets BDD. Dette viser systemets blokke og hvad disse indeholder af komponenter. Diagrammet viser også inputs og outputs til hver enkelt blok.



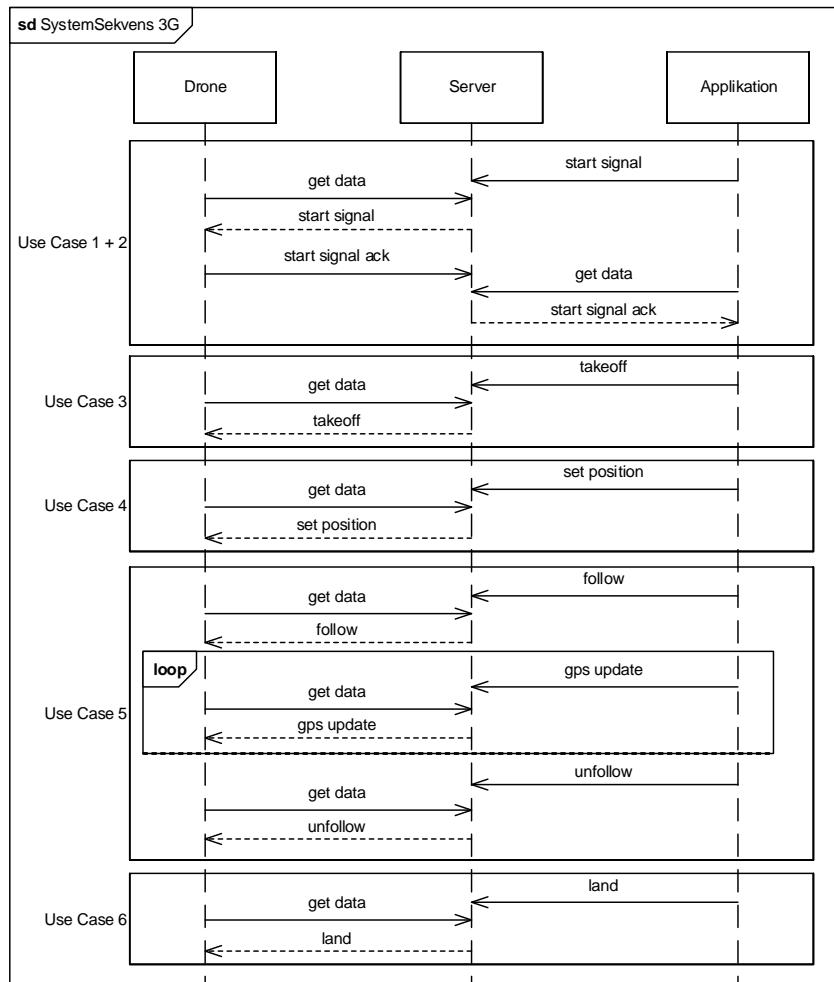
Figur 3.2: Block definition diagram

For en mere detaljeret forklaring af figur 3.2 se afsnit 2.1 i Arkitektur & Design.

3.3.2 Software

Arkitekturen og design til systemets software blev udviklet gennem N+1 modellen, som også er beskrevet i afsnit 3.1.4. Arkitekturen blev udviklet gennem Logical View.

Kommunikationen mellem systemets blokke er beskrevet i to systemsekvensdiagrammer, der viser kommunikationen i relation til systemets Use Case's. Figur 3.3 giver et overblik over Use Case 1-7, som er de Use Case's hvor ISP benyttes.

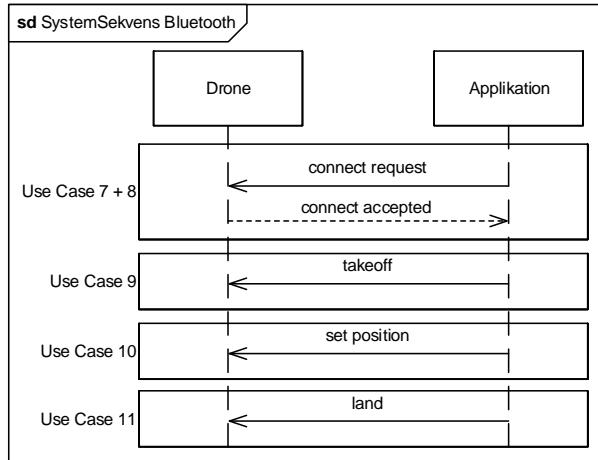


Figur 3.3: Sekvensdiagram for systemet - 3G

Det fremgår af figur 3.3, at når systemet benytter ISP, går alt kommunikationen mellem dronen og applikationen igennem serveren. Dette bevirket at dronen og applikationen aldrig har direkte kommunikation med hinanden. Ulempen ved dette er, at der skal arbejdes med timeouts, for at detektere om dronen eller applikationen har mistet forbindelsen til resten af systemet ved kommunikationsfejl.

På figur 3.3 ses det, at kommunikationen i de enkelte Use Case's foregår på samme måde, der sendes blot forskellige kommandoer.

Figur 3.4 omhandler de resterende Use Case's, hvor kommunikationen forgår via Bluetooth.



Figur 3.4: Sekvensdiagram for systemet - Bluetooth

På figur 3.4 ses det, at serveren er taget ud af systemet. Kommunikationen foregår nu direkte imellem dronen og applikationen. Når de to enheder opretter forbindelse, er der derfor ikke behov for at sende et Start-signal.

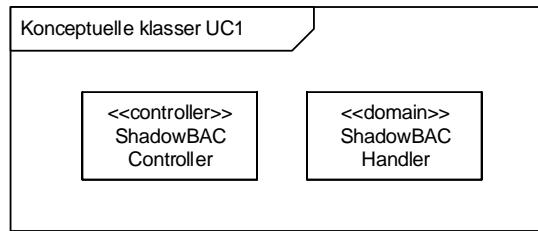
Bluetooth-kommunikationen har en kortere rækkevidde end kommunikationen via 3G igennem serveren. Grundet dette, er der ikke en Use Case i systemet, hvor dronen følger brugeren ved hjælp af Bluetooth, men udelukkende Autonomous Takeoff, Autonomous Landing og Set Drone Position.

Logical View fungerer ved at systemets Use Case's gennemgås for hver enkel CPU. I dette system blev der fundet tre CPU'er (drone, server og applikation), hvor Logical View blev udarbejdet for hver af disse. Først blev de konceptuelle klasser baseret på den specifikke Use Case og systemets tilhørende domænemodel fundet. Herefter blev et sekvensdiagram opstillet, hvorefter et statisk klasse-diagram for den enkelte Use Case kunne udledes. De enkelte statiske klasse-diagrammer for hver Use Case kunne til sidst samles til ét, som gav et overblik over al funktionaliteten på CPU'en.

Konceptuelle klasser kan indeles i tre katogorier: Controller-, Domæne- og Boundaryklasser.

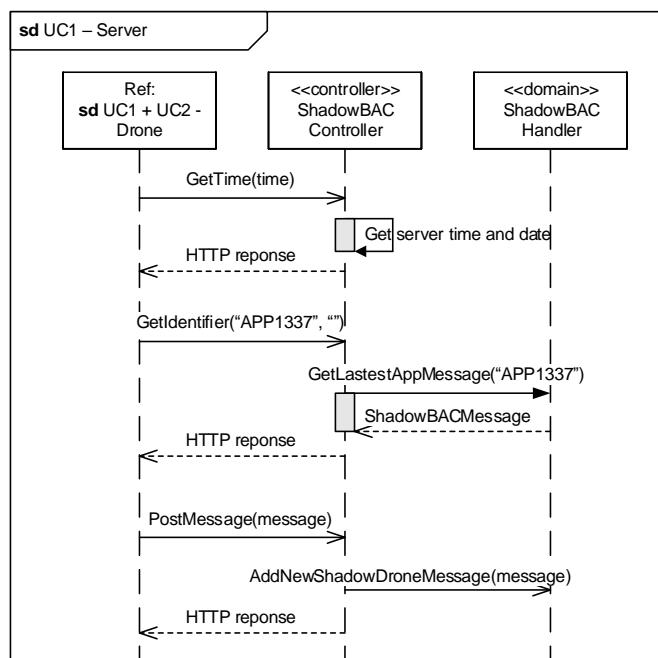
- Controllerklasser har hovedansvaret for den pågældende Use Case og er bindeleddet mellem Domæne- og Boundaryklasserne.
- Domæneklasser indeholder funktionalitet, som kun benyttes på den enkelte CPU og som ikke kan tilgås udefra.
- Boundaryklasser er CPU'ens interface ud og ind af systemet.

For at visualisere processen i Logical View gennemgås Use Case 1 for serveren. Først findes de konceptuelle klasser. Figur 3.5 viser de konceptuelle klasser for serveren, baseret på domænemodellen og Use Case 1.



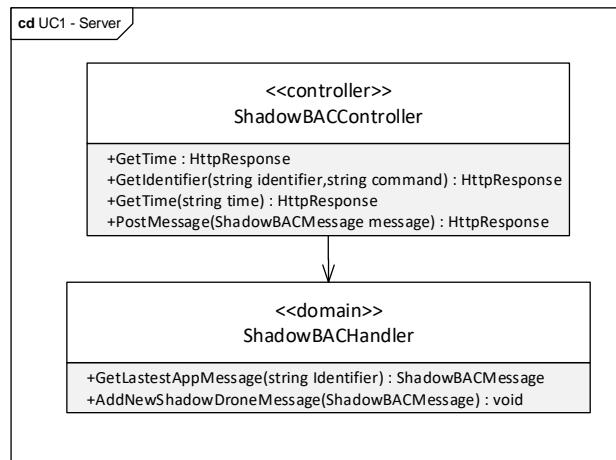
Figur 3.5: Server konceptuelle klasser - Use Case 1

Et sekvensdiagram kan laves ved at gennemgå Use Case'en og opstille kommunikationen mellem de konceptuelle klasser. Figur 3.6 viser serverens sekvensdiagram for Use Case 1.



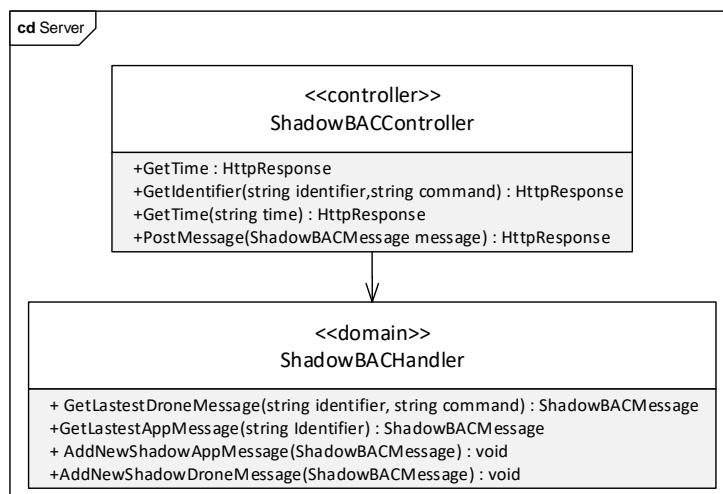
Figur 3.6: Server sekvensdiagram - Use Case 1

Baseret på sekvensdiagrammet kan funktionerne trækkes ud i et statisk klassediagram. Derudover defineres essentielle attributter ligeledes i diagrammet. Figur 3.7 viser det statiske klassediagram baseret på serverens Use Case 1.



Figur 3.7: Server klassediagram - Use Case 1

På denne måde gennemgås alle Use Case's for hver enkel CPU og et samlet statisk klassediagram kan opstilles for hver CPU. Figur 3.8 viser serverens samlede klassediagram fundet gennem Use Case's, hvor der er tilføjet yderligere metoder til klasserne.



Figur 3.8: Samlet klassediagram for serveren

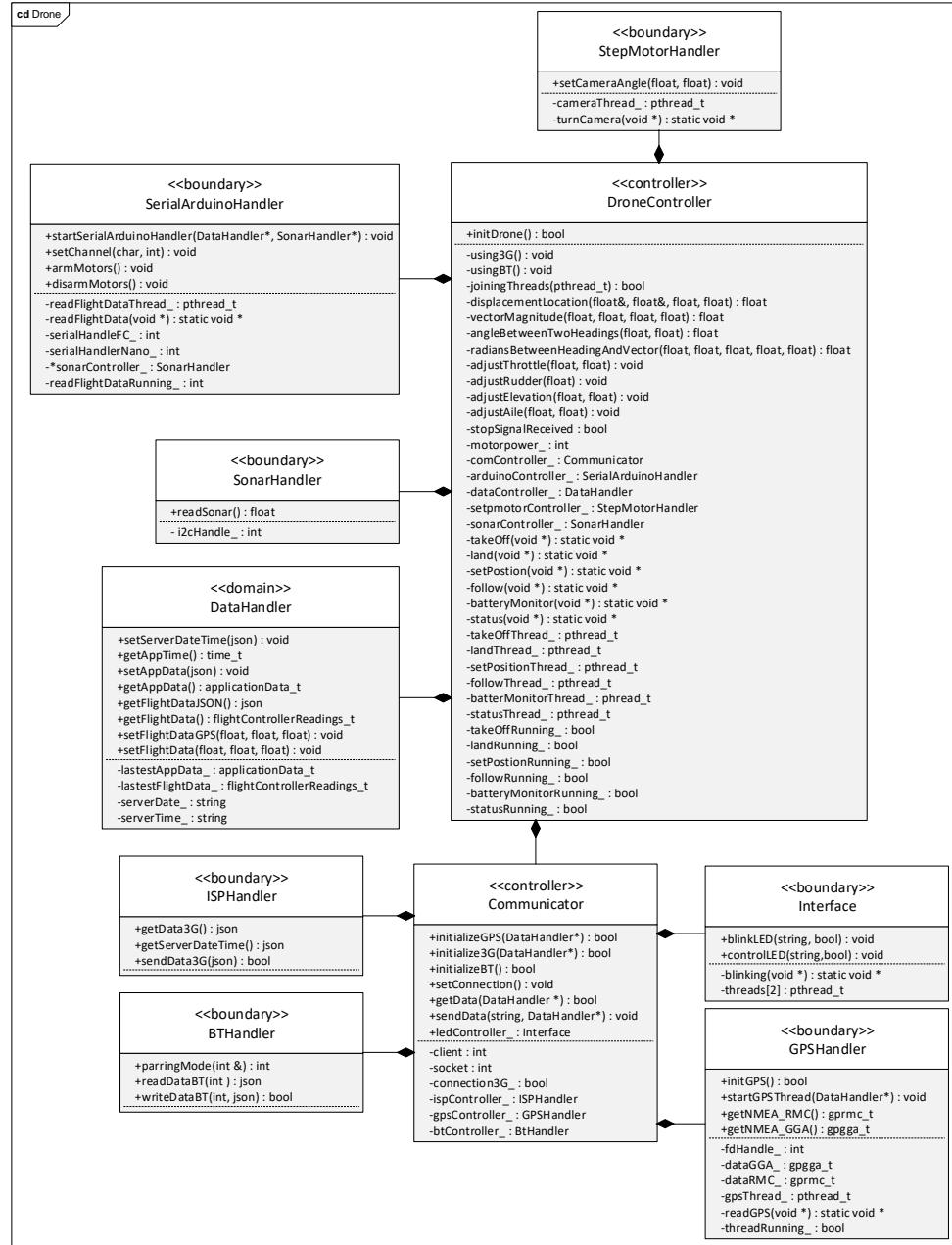
Tabel 3.1 beskriver ansvarsområderne for klasserne i figur 3.8.

Navn	Beskrivelsen
ShadowBACController	Controlleren, som står for at håndtere de forskellige HTTP beskeder fra og til dronen samt smartphone applikationen.
ShadowBACHandler	Denne klasse står for at håndtere serverens database. Dette indebærer at hente og gemme beskeder i databasen.

Tabel 3.1: Beskrivelser af klasserne i figur 3.8

Drone

Dronens samlede klassediagram er ligeledes fremkommet gennem Logical View for dronens CPU. Her er alle Use Case's gennemgået og funktionaliteten er vist gennem klassediagrammet på figur 3.9.



Figur 3.9: Samlet klassediagram for dronen

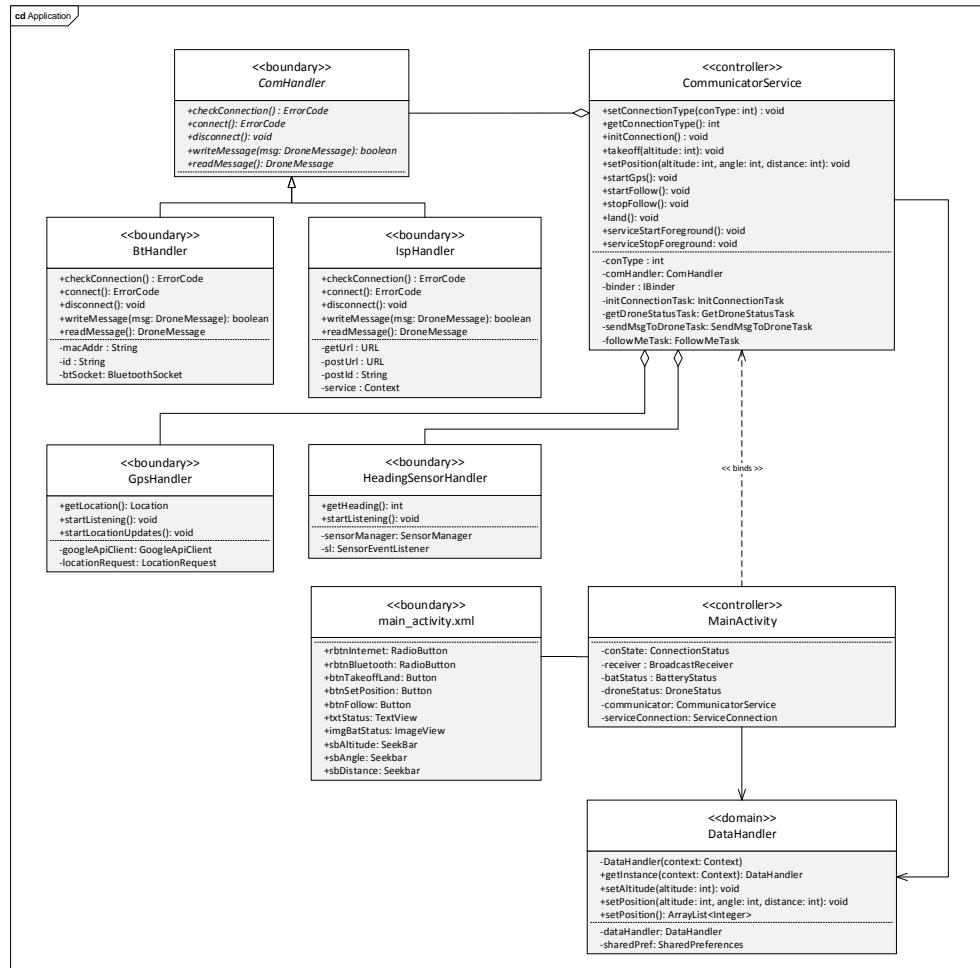
Tabel 3.2 beskriver ansvarsområderne for klasserne i figur 3.9.

Navn	Beskrivelsen
DroneController	Dette er main-controlleren på dronen. Denne står for at holde dronen i luften og styrer denne.
DataHandler	Denne klasse står for al lagring og håndtering af data på dronen.
Communicator	Denne klasse står for at håndtere al ekstern kommunikation på dronen. Dette indebærer kommunikationen over Bluetooth, 3G og GPS.
ISPHandler	Denne klasse står for kommunikationen over 3G-modemet.
GPSHandler	Denne klasse står for at kommunikere med og konfigurere GPS-modulet.
Interface	Denne klasse står for at styre interfacet på dronen.
SerialArduinoHandler	Denne klasse står for at håndtere den serielle kommunikation fra Raspberry Pi'en ud til både FlightController'en og Arduino Nano'en.
SonarHandler	Denne klasse står for at håndtere udlæsningen af data fra sonarsensoren samt konverteringen.
BtHandler	Denne klasse står for kommunikationen over Bluetooth samt etableringen af denne forbindelse.

Tabel 3.2: Beskrivelser af klasserne i figur 3.9

Applikation

For applikationen er alle Use Case's gennemgået og følgende klassediagram på figur 3.10 er opstillet på baggrund af denne gennemgang.



Figur 3.10: Samlet klassediagram for applikationen

Tabel 3.3 beskriver ansvarsområderne for klasserne i figur 3.10.

Navn	Beskrivelsen
main_activity.xml	MainActivities interface beskrevet i en xml fil.
MainActivity	Controllerklassen, der håndterer input fra brugeren og kommunikerer med servicen.
CommunicatorService	Styrer alt kommunikation med dronen i applikationens baggrund og sender beskeder tilbage til MainActivity.
ComHandler	En abstrakt klasse de andre kommunikationshandler arver fra. Dette fører til et fælles interface og mere genanvendelig funktionalitet.
BtHandler	Ansvar for at kommunikere over Bluetooth.
IspHandler	Ansvar for at kommunikere over internettet.
DataHandler	DataHandler'en er en Singletonklasse, der står for at skrive og læse fra mobiltelefonens interne hukommelse.
GpsHandler	Modtager GPS koordinater ved hjælp af Google Play services location API.
HeadingSensorHandler	Beregner smartphonens retning i forhold til magnetisk nord baseret på accelerometer- og magnetometerværdier.

Tabel 3.3: Beskrivelser af klasserne i figur 3.10

For en mere detaljeret forklaring af arkitekturen henvises til afsnit 3.2 i Arkitektur & Design i bilag.

3.4 Design

I dette afsnit er udarbejdelsen af systemets design beskrevet. Først vil designet af systemets hardware blive gennemgået dernæst systemets software.

3.4.1 Hardware

Ud fra forudsætningerne til hardwaren, som er visualiseret i systemets IBD, blev følgende tre print designet i Multisim.

- Spændingsregulator-print
- Stepmotor-print
- Droneinterface-print

Spændingsregulator-printet blev designet med en spændingsregulator LM75S05 [12], der kan leve en strøm op til 2A. Dette print blev dog fejldimensioneret, da Raspberry Pi'en og de tilsluttede USB enheders totale strømforbrug var højere end forventet. Dette ledte til høj varmeudvikling i spændingsregulatoren, som gjorde at udgangsspændingen faldt. Dette førte til, at Raspberry Pi'en genstartede. Selv med en køleplade på spændingsregulatoren var udgangsspændingen for lav.

Det blev derfor valgt at udskifte printet med en DC-DC konverter med regulerbar spændingsoutput [13]. Dette komponent kan levere en strøm på op til 3A, som er tilstrækkelig for at systemet kan fungere korrekt.

Da Raspberry Pi'ens GPIO'er ikke kan levere nok strøm til at forsyne stepmotoren direkte, blev et stepmotor-print designet. Printets vigtigste komponenter er fire MOSFET transistorer, der fungerer som en digital switch og styres fra Raspberry Pi'en.

Droneinterfacet-printet består af fire LED dioder i forskellige farver og en RGB diode. Disse benyttes til at informere brugeren om dronens tilstand se afsnit 1.10 i Kravspecifikation i bilag. For at begrænse strømmen igennem dioderne, blev det valgt at indsætte en formodstand på 82Ω foran hver diode. Dette fører til en strøm igennem dioden på cirka 15mA se afsnit 2.4.1 i Arkitektur & Design i bilag for yderligere detaljer omkring droneinterface-printet.

Dronens mekaniske design er beskrevet i afsnit 2.6 Mekanisk Design i Arkitektur & Design i bilag. Dette afsnit beskriver, hvordan de forskellige lag på dronen er bygget op.

3.4.2 Software

Dette afsnit omhandler udviklingen af systemets softwaredesign. Softwarens design er opdelt i de fem N+1 View's: Process-, Data-, Deployment-, Security- og Implementation View. Se afsnit 3.1 i Arkitektur & Design i bilag for en uddybende forklaring af de enkelte View's.

3.4.2.1 Process View

Her redegøres for de forskellige processer og tråde i systemet. Det beskrives, hvad der blev gjort, for at undgå fejl i forbindelse med data, som tilgås fra flere tråde samtidig.

Drone

Dronen benytter en Raspberry Pi 3 med en QuadCore processor. Det blev derfor valgt at udvikle et multitrådet system. Under dronens opstart startes tråde til at udlæse data fra relevante sensorer eller subsystemer. Main-tråden håndterer kommunikationen med enten serveren over 3G eller direkte med applikationen gennem Bluetooth. Baseret på kommandoerne, som dronen modtager, starter denne tilhørende tråde.

På dronen er alt data samlet i en klasse, som er implementeret trådsikkert gennem brugen af mutex's. Dette sikrer, at trådene på dronen ikke tilgår den samme hukommelse på samme tid.

Applikation

I applikationen kører en stor del af funktionaliteten på main-tråden, som også bliver kaldt for UI-tråden. På denne tråd er det ikke tilladt at udføre blokerende kald, der tager lang tid, da Android lægger stor vægt på at UI'en altid er åben for brugerinput [14]. Derfor udføres al kommunikation med serveren og dronen i baggrundstasks. For at disse tasks kan køre parallelt, udføres de på hver sin tråd. For at returnere resultatet af arbejdet, der blev udført i baggrunden, sendes en broadcast til *MainActivity*, der giver informationen videre til brugeren.

Der kan være flere tasks, der vil kommunikere med dronen eller serveren på samme tid. Det er derfor sikret, at både *BtHandler*- og *IspHandler*-klassen er implementeret således, at dette er muligt.

Derudover er *DataHandler*-klassen lavet trådsikker, ved hjælp af Java's *synchronize*-funktionalitet [15]. Dette fører til, at kun én metode i denne klasse kan blive kaldt ad gangen. Andre tråde, der prøver at kalde en metode, bliver sat i kø.

3.4.2.2 Data View

I dette afsnit dannes et overblik over det persistente data i systemet. Her er det især serveren, og opbygningen af den tilhørende database, der beskrives.

Server

Serveren har en database tilknyttet igennem Cloud Hosting servicen GearHost [16]. Når serveren modtager en besked fra enten dronen eller applikationen gemmes den i databasen, således modparten kan tilgå den senere. Databasen indeholder to tabeller, én til beskeder fra applikationen og én til beskeder fra dronen.

Databasen indeholder fem procedurer til at håndtere tabellerne:

- To procedurer til at returnere det nyeste element fra hver tabel.
- En procedure til at returnere det nyeste element i tabellen med den specifikke kommando. (Kun dronens tabel)
- To procedurer til at tilføje et nyt element i hver af tabellerne.

På denne måde kan der gemmes eller hentes beskeder i databasen.

Drone

Dronen benytter ingen persistent data. Der gemmes dog log-filer på dronen, som er blevet brugt under udviklingen af systemet.

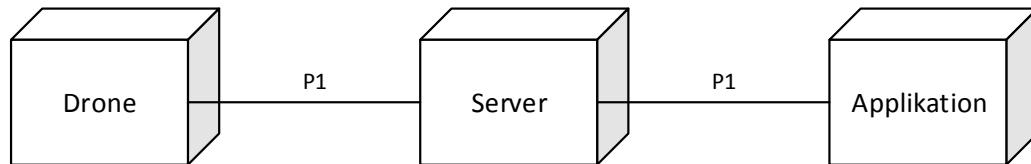
Applikation

Applikationen har enkelte værdier, der gemmes i mobiltelefonens interne hukommelse. Disse værdier gemmes hovedsageligt for at kunne genskabe applikationens brugergrænseflade, efter den er blevet lukket ned. Derudover gemmes dronens ID, så brugeren ikke er nødt til at indtaste denne på ny, hver gang applikationen startes.

3.4.2.3 Deployment View

Dette afsnit beskriver hvilke protokoller, der er brugt i forbindelse med kommunikationen mellem systemets overordnede blokke.

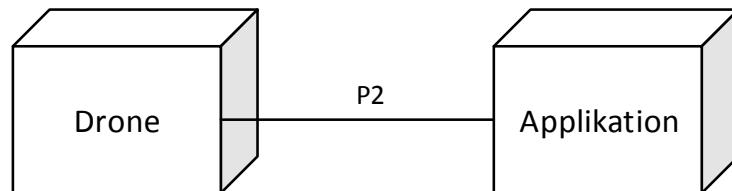
På figur 3.11 ses, kommunikationen mellem systemets blokke, når datapakkerne sendes over internettet.



Figur 3.11: Kommunikation mellem systemets blokke over internettet

Protokollen P1, der er vist på figur 3.11 består af en JSON streng. Denne streng bliver pakket i en HTTP body [17], som sendes fra én blok til en anden. For at sikre pålidelig kommunikation, benyttes TCP [18], som anvender retransmission.

På figur 3.12 ses kommunikation i systemet over Bluetooth. Det ses, at serveren ikke længere fungerer som b林deled, men at kommunikationen nu er direkte mellem dronen og applikationen.



Figur 3.12: Kommunikation mellem systemets blokke over Bluetooth

På figur 3.12 bliver protokollen P2 anvendt. Denne protokol, består af de samme JSON strenge som P1, dog bliver disse pakket i en Bluetooth besked [19].

Da der i systemet bruges både ISP og Bluetooth som kommunikationsformer, bliver der brugt to forskellige protokoller til dette.

Da beskederne gemmes i databasen når ISP benyttes, er det valgt at bruge den samme beskedstruktur for at minimere antallet af tables. Beskedstrukturen kan ses i tabel 3.4

Beskedstruktur		
Type	Navn	Beskrivelse
String	Identifier	Beskriver, hvem der sender beskeden
String	Command	Beskedens kommando
String	Date	Datoen, da beskeden modtages af serveren
String	Time	Tiden, da beskeden modtages af serveren
Float	GPSLong	Senderens GPS longitude
Float	GPSLat	Senderens GPS latitude
Float	Altitude	Dronens højde / Dronens ønskede højde
Float	Angle	Dronens ønskede vinkel til brugeren
Float	Distance	Dronens ønskede afstand til brugeren
Float	Heading	Retningen, senderen bevæger sig i

Tabel 3.4: Systemets besked struktur

En pakket JSON streng kan have følgende udseende:

```
{
    "Identifier": "App1337",
    "Command": "FOLLOW_SIGNAL",
    "Date": "08-11-16",
    "Time": "12:48:49",
    "GPSLong": 10.1914854,
    "GPSLat": 56.1714973,
    "Altitude": 15.0,
    "Angle": 90.0,
    "Distance": 20.0,
    "Heading": 45.0
}
```

Listing 3.1: Besked pakke i JSON format

Listing 3.1 viser en besked, sendt fra applikationen, der beder dronen om at følge brugeren. En liste over alle kommandoer, der kan sendes, vises i afsnit 3.5.1.1 i Arkitektur & Design i bilag.

3.4.2.4 Security View

Som beskrevet i afsnit 2.4 blev det valgt at nedprioritere internetsikkerheden. Det blev derfor fravalgt at bruge HTTPS protokollen eller en anden form for kryptering. Det blev besluttet at bruge en sikker forbindelse, når applikationen forbinder til dronen over Bluetooth. Dette beskytter mod "Man-In-The-Middle" angreb udefra [20].

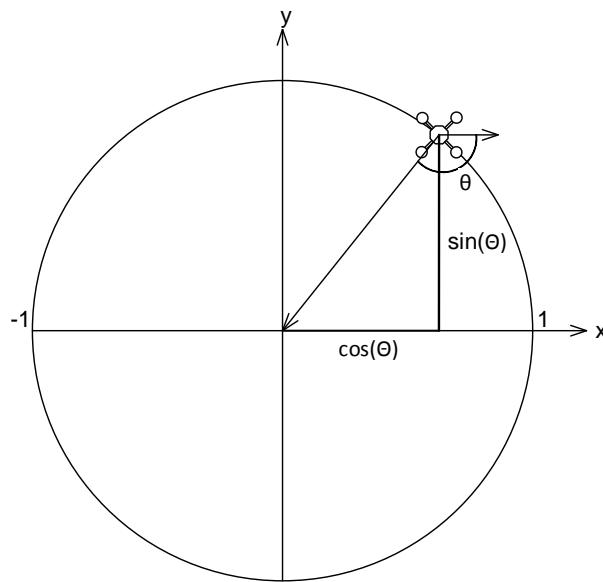
Det blev derimod prioriteret højt, at dronen udelukkende fungerer, når brugeren ønsker det. For at sikre dette blev der implementeret et Stop-signal. Når dronen modtager dette signal, lander dronen med det samme. Når dronen er landet eller allerede er på jorden i forvejen, slukker Arduino Nano'en for motorerne, og Raspberry Pi'en lukker ned. Dette sikrer at motorerne ikke kan starte op, mens brugeren er i nærheden.

3.4.2.5 Implementation View

Dette afsnit vil indeholde en kort forklaring af, hvordan flyvealgoritmerne på dronen er designet. Dette View omfatter yderligere i bilag opsætningen af AeroQuad-koden på flightcontrolleren, opsætningen af serveren, de vigtigste matematiske funktioner samt reguleringsfunktioner på dronen. Derudover er de vigtigste funktionaliteter i applikationen beskrevet. Dette kan findes i afsnit 3.7 i Arkitektur & Design i bilag.

I dronens software blev flyvealgoritmerne designet ud fra enhedscirklen. Dette betyder, at når antallet af radianer fra dronens retning og om til retningen mod punktet er beregnet, kan flyvealgoritmerne beregne, om der skal flyves frem eller tilbage og til hvilken side.

Flyvealgoritmen til at flyve frem og tilbage benytter cosinus, og flyvealgoritmen til at flyve til siden benytter sinus. En skitse af dette ses på figur 3.13.



Figur 3.13: Forklaring af dronens flyvealgoritmer

For at forklarere figur 3.13 yderligere, kommer der et eksempel på figur 3.14. Det antages, at der er 120° fra dronens retning til det ønskede punkt i origo.

```

angle := 120:
angleRadians :=  $\frac{angle \cdot \pi}{180}$ :
evalf(cos(angleRadians))
-0.5000000000
evalf(sin(angleRadians))
0.8660254040

```

Figur 3.14: Udregning af cosinus og sinus af dronen på enhedscirklen

På figur 3.14 ses det, at $\cos(\text{angleRadians})$ giver et negativt tal. Det betyder, at dronen skal flyve bagud. $\sin(\text{angleRadians})$ giver et positivt tal, hvilket betyder at dronen skal flyve til højre.

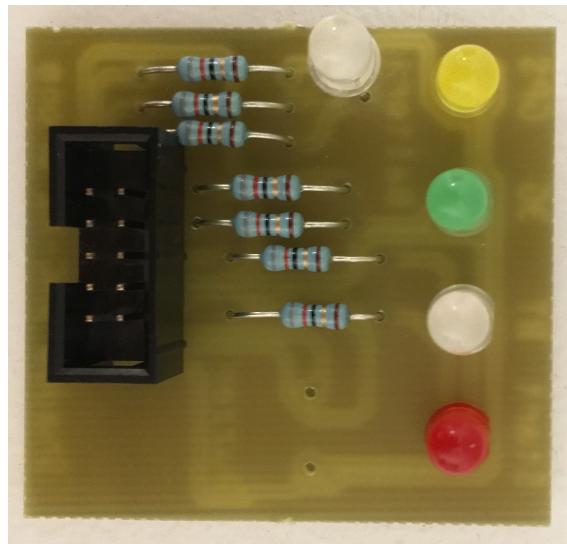
3.5 Implementering

I dette afsnit vil der være en beskrivelse af implementeringen af hardwaren og softwaren i projektet.

3.5.1 Hardware

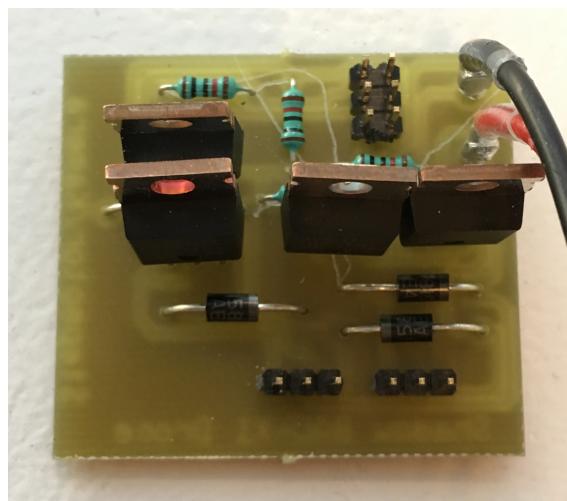
Der er produceret to print, et Droneinterface-print og et stepmotor-print.

For en dybere forståelse af de to print, henvises til design af printene i afsnit 2.4 i Arkitektur & Design i bilag. Droneinterface-printet ses på figur 3.15.



Figur 3.15: Droneinterface-printet

Stepmotor-printet ses på figur 3.16.



Figur 3.16: Stepmotor-printet

3.5.2 Software

Softwareen er opdelt i tre dele: server, drone og applikation. De vigtigste elementer fra source-koden kan ses i Implementation View afsnit 3.7 i Arkitektur & Design i bilag. Source-koden er vedlagt i bilag.

Server

Serveren blev udviklet i C# 5.0, som er en del af .NET frameworket version 4.5.2. Koden blev udviklet i Microsoft Visual Studio 2015 Update 3. EntityFramework version 6.1.3 blev benyttet i projektet til håndteringen af databasen. For at redigere i databasen, der blev oprettet gennem GearHost [16], blev Microsoft Server Management Studio 2016 benyttet. Gennem dette program blev databasens tabeller og procedurer oprettet.

Drone

Dronens software blev udviklet i C++. For at compilere projektet blev g++ compileren version 4.9.2 benyttet. For at udvikle softwaren blev IDE'et Netbeans 8.1 anvendt. Softwaren, der kører på Raspberry Pi'en, var Jessie Lite. Jessie Lite bygger på Debian Linux [21]. Da Raspberry Pi'en er multikernet, blev det valgt at lave softwaren multitrådet. Dette blev gjort gennem API'et POSIX Threads. [22]

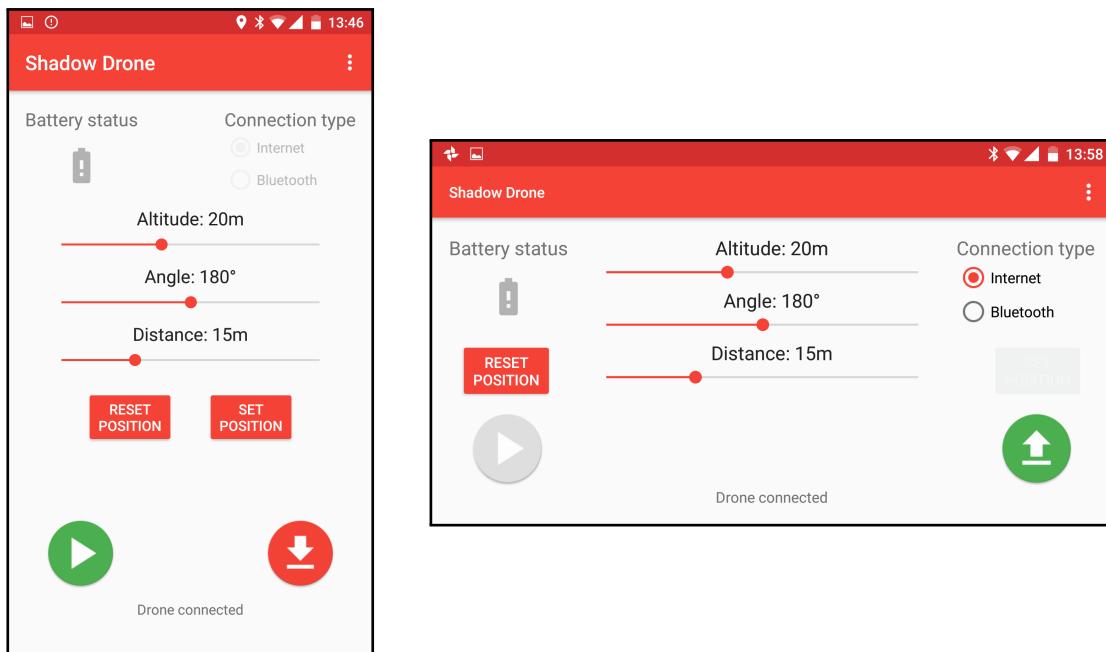
Det blev valgt at bruge C++, da dette giver mulighed for objektorienteret programmering og C++ er et cross platform programmeringssprog.

Applikation

Systemets applikation blev udviklet i Android Studio 2.2.2 [23]. Det er det officielt anbefalede IDE til udvikling af Android applikationer. IDE'et indeholder funktionaliteter som eksempelvis debugging på både en simuleret enhed og en eksisterende enhed, der forbindes til computeren via USB. Det anvendte programmeringssprog er Java 8, som blev valgt, da Android's standard klasser er skrevet i Java. Projektet bliver compileret med Gradle 2.14.1.

For at kunne få en mere præcis lokation af brugeren blev API'et Google Play service location 10.0.1 anvendt.

Applikationens interface blev designet i XML filer. Hovedskærmens design kan ses på figur 3.17.



Figur 3.17: Applikationens interface

Applikationen er designet i både en horisontal og vertikal orientering. Derudover er figur 3.17 et eksempel på, hvordan interfacets knapper bliver aktiveret og deaktiveret, alt efter hvad dronens status er. Der kan findes yderlige information om applikationens interface i afsnit 1.9 i Kravspecifikation i bilag.

3.6 Test

Dette afsnit indeholder en beskrivelse af, hvordan systemet blev testet. Dette omhandler modultest, integrationstest og accepttest. For en detaljeret beskrivelse af testene henvises der til Modul- & Integrationstest og Accepttestspezifikation i bilag.

3.6.1 Modultest

Der blev løbende igennem projektforløbet udført modultests af de forskellige dele i systemet. Der blev derved udført modultest af både serveren, dronen og applikationen.

3.6.1.1 Server

Modultesten af serveren blev brugt til at sikre, at serverens funktionalitet fungerede efter hensigten. Der blev brugt et tredjepartsprogram Postman [24] til modultesten.

3.6.1.2 Drone

Modultesten af dronen blev brugt til at sikre at både de to hardwareprint, der blev produceret, og de klasser, der bruges i koden, virker efter hensigten. På hardwareprintene blev det testet, at printenes output stemmer overens med det forventede.

Klasserne i koden blev testet ved brug af en testklasse og log-beskeder i koden. Derved kunne det aflæses i log-filen, hvorvidt klassen fungerede efter hensigten. Nogle af klasserne

benyttede hardware til visuelt at verificere, at klassen virkede efter hensigten. Dette var f.eks. gældende for klassen *StepMotorHandler*, som styrer stepmotoren. Derved blev enkelte klasser også integrationstestet med deres pågældende hardware.

3.6.1.3 Applikation

Modultesten af applikationen blev brugt til at sikre, at funktionaliteten i applikationen fungerede efter hensigten. For at teste funktionaliteten blev der oprettet en *TestActivity*. Denne *TestActivity* kunne teste funktionaliteten i applikationen, og kunne derfor sikre, at denne virker efter hensigten. *MainActivity* blev testet ved at udkommentere dele af *CommunicatorService*-koden og simulere feedback fra denne, for at se om brugergrænsefladen agerede som forventet. Testenes resultater blev delvist udskrevet på skærmen i *TestActivity* eller i Android Studio's Log.

3.6.2 Integrationstest

Der blev lavet en integrationstest af det samlede system. Denne integrationstest sikrede, at hele systemets samspil fungerede korrekt og efter hensigten.

Integrationstesten blev opdelt i tre faser. En integrationstest blev udført efter hvert Sprint for at sikre funktionaliteten, som var udviklet i det pågældende Sprint. Hver test testede sammenspillet imellem serveren, dronen og applikationens funktionalitet.

Der henvises til Procesrapportens afsnit 1.3 Udviklingsforløb, for en detaljeret forklaring af Sprint's i projektforløbet.

3.6.3 Accepttest

Sidst i projektforløbet blev en accepttest af hele systemet udført. Accepttesten testede alle Use Case's og deres undtagelser. Igennem denne test blev det sikret, at hele systemet opfyldte de angivne krav. Ikke-funktionelle krav blev ligeledes testet i accepttesten.

Da accepttesten blev udført i et laboratorium, har dronen været fastgjort i et teststativ. Dette kan ses på figur 3.18.



Figur 3.18: Testopstilling for accepttesten.

I teststativet, kan dronen dreje om sin egen vertikale akse, samt hælde i alle retninger. Den kan dog ikke ændre højde.

Resultater 4

Dette afsnit præsenterer resultaterne fra projektets gennemførte accepttest. For en yderligere og mere detaljeret oversigt se Accepttestspecifikation i bilag.

Use Case 1 og 2 behandler initialiseringen af internetforbindelsen mellem applikationen og dronen igennem serveren. Accepttesten af disse forløb problemfrit for både hovedscenarierne og undtagelserne. Opstarten af Bluetoothforbindelsen i Use Case 7 og 8 kunne godkendes uden anmærkninger.

Use Case 3, 4, 5 og 6 samt 9, 10 og 11 omhandler styringen af dronen igennem henholdsvis internet og Bluetooth. Kommunikationen mellem systemets blokke samt sekundære aktører fungerede fejlfrit. Applikationen opførte sig igennem hele testforløbet som beskrevet i kravene. Da testene blev udført i testopstillingen vist på figur 3.18 i afsnit 3.6.3 Accepttest, har det ikke været muligt at verificere dronens fulde flyveegenskaber. Dronen samt dens kamera bevægede sig dog i testopstillingen som forventet i forhold til de anførte krav. Dette gjorde, at testene til de nævnte Use Case's blev godkendt.

Både applikationen og dronen kunne modtage GPS koordinater og aflæse retningen i forhold til magnetisk nord fra deres sensorer. Applikationen opfyldte de stillede krav. Dronens sensorer opfyldte dog ikke kravenes præcision. Dronens GPS-præcision blev kun testet i ét punkt, hvor præcisionen ikke var tilstrækkelig. Derfor kunne dronens GPS-præcision ikke opfylde kravet.

Applikationen blev testet på en smartphone med Android version 6.0.1. I Android Studio blev det defineret, at applikationen udelukkende kunne indeholde funktionalitet, der er tilgængelig i version 4.1.x. Kravet blev opfyldt, baseret på Android Studio's garanti, dog ikke testet.

Serverens krav om uptime er garanteret igennem serverens leverandør Gearhost. Der garanteres en uptime på 99.999% [16]. Serverens responstid blev godkendt.

Dronens flyvehastighed blev ikke godkendt, da denne ikke kunne testes i testopstillingen. Batteriets amperetimer var ikke tilstrækkelige til at opfylde kravet om en flyvetid på 30 minutter.

Opstartstiden af systemet bestod accepttesten delvist, da denne afhænger af AeroQuads flightcontroller, hvis opstartstid varierer kraftigt.

Dronen har på intet tidspunkt igennem projektforløbet været testet udenfor testopstillingen.

Diskussion af resultater

5

Dette afsnit indeholder en diskussion af resultaterne fra kapitel 4.

Det er i accepttesten verificeret, at dronen prøver på at udføre det forventede under testene. Det har dog ikke kunne testes yderligere, da det af sikkerhedsmæssige årsager blev valgt at beholde dronen i testopstillingen. Dronen er låst i en position, således denne ikke kan hæve sig over jorden. Dette har givet en stærk begrænset testmulighed af dronens flyveegenskaber. Dronens flyvealgoritmer er derfor ikke testet til fulde. En anden testopstilling skulle have været udviklet for at teste dronens funktionalitet under en flyvesession.

Værdierne, som dronen regulerer efter i flyvealgoritmerne, bygger på antagelser og ikke på reelle tests med dronen. Algoritmerne skal derfor verificeres og modificeres efter gennemførte tests med dronen. Implementeringen af dronens landingsalgoritme kan f.eks. lede til at dronen aldrig vil lande, da reguleringen af motorkraften til sidst vil være så lille, at ændringen ingen reel effekt har. Tilsvarende problemer kan opstå med dronens andre algoritmer. Dette kan kun verificeres ved at teste dette med dronen i flyvende tilstand.

Systemets kommunikation virker både via internettet og Bluetooth. Det viste sig dog under implementering og testforløbet, at rækkevidden med Bluetooth var længere end forventet i analysen. Derfor burde Follow-funktionaliteten også være implementeret, når brugeren benytter Bluetooth. Derved kunne Bluetooth benyttes som standard kommunikationstype i stedet for internet. Dette vil betyde at en algoritme skal implementeres på både dronen og applikationen, som sørger for at der skiftes dynamisk mellem de to kommunikationsformer, alt efter om Bluetooth signalet er tilstrækkeligt eller ej.

Der skal udføres yderligere tests for at bedømme, om Bluetooth forbindelsen er stabil nok til at internetforbindelsen kan fjernes helt fra systemet. Dette vil føre til, at der ikke skal bruges en server og at brugeren ikke har brug for et SIM-kort til dronen. Hvis internetforbindelsen skal beholdes i systemet, skal den udvides med en mere pålidelig protokol. På nuværende tidspunkt sendes kun et acknowledgement fra dronen, på Start-signalet. Protokollen skal ændres, således at der sendes et acknowledgement som svar på alle modtagne beskeder.

Både i applikationen og på dronen udlæses retningen fra GPS, når dronen følger brugeren. Denne retning antages at være mere præcis end sensorerne, der bestemmer enhedens retning i forhold til magnetisk nord. Især på dronen er magnetometeret upræcist, da de store strømme til motorerne danner magnetfelter omkring ledningerne, som giver fejl i sensorværdierne. Da retningen fra GPS'en benyttes i Follow-funktionaliteten kan det give problemer, da brugeren og dronen skal være i bevægelse, før der kan modtages en korrekt retning fra GPS'en. Det er undladt at benytte smartphonens sensorer, når dronen

følger brugeren, da det i applikationen antages, at brugeren lægger smartphonen i lommen, hvormed det ikke er sikkert, hvordan denne vender. I opstarten af Follow-funktionaliteten skal der ske en flydende overgang fra sensorernes retning til GPS'ens retning. Der kunne eventuelt tages hensyn til om applikationen er synlig for brugeren eller ej, da dette kan være en indikation for, om han ser på smartphonen i øjeblikket. For at øge magnetometerets præcision på dronen kan det mekaniske design revurderes, således at kabler, der leder en stor strøm, er så langt fra magnetometeret så muligt.

Igennem projektforløbet er der, især under dronens opstart, opstået problemer med AeroQuad's flightcontroller. Tiden det tager FlightController'en at forbinde til motorenes ESC'er varierer kraftigt og tog i projektforløbet op til fem minutter. Der skal enten ændres i koden til FlightController'en eller benyttes en anden, for at løse dette problem. Det er vurderet, at det ikke er koden, som er udviklet i projektforløbet, der er årsagen til problemet, da dette også opstår, når FlightController'en tilgås med AeroQuad's egen PC-software.

Fremtidigt Arbejde 6

Dette afsnit beskriver eventuelle mangler i projektet og hvilke udvidelses- og anvendelsesmuligheder, der er i projektet.

Flere enheder

Systemet skulle videreudvikles således, at dette kan håndtere flere droner og brugere på samme tid. Dette vil indebære en udvikling på både serveren og applikationen. For netop at kunne have flere brugere i systemet, ville et loginsystem være nødvendig. Det ville ligeledes give systemet et ekstra sikkerhedsniveau.

Flyvetid

I projektet har der ikke været fokus på at maksimere flyvetiden. Det vil være et stort fokusområde i videreudviklingen af dronen. Dronen skulle ændres således, at denne kan holde sig i luften i længere tid på et fuldt opladet batteri.

Sikkerhed i form af kryptering

Sikkerheden i systemets trådløse kommunikation er lav. Hvis systemet skulle fortsætte med at benytte HTTP, skulle dette ændres til HTTPS for en mere sikker kommunikation.

Prædefineret flyvepositioner og flyvelog

I applikationen skulle brugeren have mulighed for at vælge prædefinerede positioner til dronen for at gøre den mere brugervenlig. Applikationen eller serveren i systemet skulle gemme tidligere flyvninger i form af logs. På denne måde vil brugeren kunne få en visuel gennemgang af tidligere flyveruter.

Undvigelse af forhindringer

Da dronen flyver autonomt, ville det være mere optimalt, hvis dronen kunne undvige eventuelle forhindringer. Dette kræver dog en del mere udvikling, da undvigelse af forhindringer er en krævende feature.

Kamera

Systemet skulle udvikles således, at applikationen kunne benytte GoPro kameraets funktionalitet direkte. En anden løsning kunne være at sætte et andet kamera på dronen og selv udvikle funktionaliteten til denne gennem dronens Raspberry Pi. Raspberry Pi's eget kamera kunne med fordel benyttes, da denne kan tilsluttes direkte til dronens Raspberry Pi.

Manuel styring af dronen gennem applikationen

Systemet, som det er udviklet i projektet, har kun autonome flyveegenskaber. I videreudviklingen af systemet ville det give flere anvendelsesmuligheder, hvis dronen kunne styres direkte fra applikationen. Eventuelt ved brug af smartphonens sensorer.

Google Maps integration

Da applikationen i systemet løbende modtager status beskeder fra dronen, som indeholder dronens GPS koordinater, kunne dette vises på et kort, således at brugeren kunne se dronens sidst kendte position. Dette ville være relevant, når dronen nødlander på baggrund af lavt batteri eller mistet forbindelse.

Automatiseret modultest

I videreudviklingen af projektet skulle systemets modultests automatiseres. Dette kan gøres gennem et unit test framework. Både dronen, serveren og applikationen har mulighed for at implementere et unit test system. Dette ville gøre systemets blokke nemmere at teste og vedligeholde.

Batterispænding

Batterispændingen monitoreres gennem en spændingsdeler på FlightController'en. Spændingen falder dog drastisk, når dronen starter sine motorer og trækker en stor strøm. Dette giver en misvisende måling af batteriets spænding. Der vil yderligere kunne tilføjes en strømmåler. På baggrund af begge disse målinger, kunne en algoritme laves til at udregne batteriets egentlige spænding.

GPS filtrering

På applikationen sørger API'et Google Play service location for at denne modtager det mest præcise GPS-koordinat, baseret på en algoritme der vægter GPS-koordinaternes præcision og alder. En lignende algoritme skulle implementeres på dronen for at øge præcision af flyvningen.

Konklusion 7

Det har i projektet været formålet, at udvikle et system bestående af en drone og en tilhørende applikation. Dronen skal autonomt kunne følge og filme applikationens bruger.

Gennem forløbet er N+1 benyttet til at udvikle arkitekturen og designet i systemet. Dette har især fungeret godt på dronen og serveren. Logical View har dog givet visse udfordringer for applikationen, da meget af funktionaliteten bliver startet af Android styresystemet ved hjælp af callbacks. Dette er svært at illustre i sekvensdiagrammer. N+1 har været let at kombinere med den iterative arbejdsproces gennem projektforløbet. De forskellige View's har ført til en klar opdeling i de enkelte faser i projektets systemarkitektur og -design.

Projektets applikation er implementeret som en Android applikation. Denne kan benytte både internet og Bluetooth som kommunikationsform. Applikationen viser dronens aktuelle batterispænding i den grafiske brugergrænseflade. Dronens højde samt vinkel og afstand til brugeren kan styres direkte gennem applikationens brugergrænseflade. Derudover er applikationens pålidelighed testet og det kan konkluderes, at den kan skifte orientering samt lukkes ned, når dronen er i luften uden at miste forbindelsen eller terminere grundet fejl. Dermed er kravene stillet til applikationen opfyldt.

Serveren i systemet kan modtage beskeder over internettet fra både dronen og applikationen. Denne kan gemme og hente beskeder fra databasen. Dermed opfylder serveren alle krav, der er stillet til denne i kravspecifikationen.

Dronen har to muligheder for kommunikation. Den kan kommunikere igennem internettet med systemets server eller direkte med applikationen via Bluetooth. Hvis dronen ikke kan oprette forbindelse til internettet, skifter denne automatisk til Bluetooth. Dronen kan autonomt lette og lande, når applikationen sender de tilhørende kommandoer. Dronen kan autonomt flyve til et givent GPS koordinat, denne har modtaget fra applikationen. Den vil forsøge at opretholde samme position til brugeren, når Follow-funktionaliteten er aktiveret. Såfremt dronen mister forbindelsen til serveren over internettet eller applikationen gennem Bluetooth, lander denne autonomt på sin nuværende position. Hvis dronens batterispænding falder under et specifikt niveau, lander dronen ligeledes autonomt. Et GoPro kamera kan monteres på dronen og dette kan roteres, omkring dronens vertikale akse. Dronen opfylder dermed de stillede krav. I projektet er det dog ikke lykkedes at teste dronens flyveegenskaber til fulde. Dronen har under hele udviklingsforløbet, været monteret i en testopstilling, som gør, at dronen ikke kan forlade jorden. Derfor er flyveegenskaberne ikke testet tilstrækkeligt.

Dronens manglende flyvetests begrænsrer dog gruppens tillid til prototypens pålidelighed. Det kan konkluderes, at der skal afsættes mere tid til både opstillingen og udførelsen

af tests i et droneprojekt end, hvad projektgruppen oprindeligt forventede. På baggrund af det udviklede produkt, kan det konkluderes, at projektet er resulteret i en prototype med fungerende og testet kommunikationsformer. Systemets yderligere funktionalitet, er implementeret jævnfør i kravspecifikationen.

References

- [1] Amazon. *Amazon Prime Air*. Nov. 2016. URL: <https://www.amazon.com/b?node=8037720011>.
- [2] Dario Floreano & Robert J. Wood. "Science, technology and the future of small autonomous drones". I: *REVIEW* 521 (maj 2015).
- [3] Hexo+. *Hexo+*. Nov. 2016. URL: <https://hexoplus.com/>.
- [4] Wikipedia. *FURPS*. Nov. 2016. URL: <https://en.wikipedia.org/wiki/FURPS>.
- [5] Wikipedia. *MoSCoW method*. Nov. 2016. URL: https://en.wikipedia.org/wiki/MoSCoW_method.
- [6] Trafik- og byggestyrelsen. *Droneregler*. Sep. 2016. URL: <http://www.trafikstyrelsen.dk/da/droneregler>.
- [7] SysML.org. *SysML Open Source Specification Project*. Nov. 2016. URL: <http://sysml.org/>.
- [8] UML.org. *Unified Modeling Language*. Nov. 2016. URL: <http://www.uml.org/>.
- [9] Craig Larman. *Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development*. third. Chapter 39 regarding N+1 View Model. Pearson Education (Us), okt. 2004.
- [10] AeroQuad. *AeroQuad*. Nov. 2016. URL: <http://aeroquad.com/content.php>.
- [11] Wikipedia. *JSON*. Nov. 2016. URL: <https://en.wikipedia.org/wiki/JSON>.
- [12] ST Microelectronics. *L78S05CV Datasheet*. 1. udg. ST Microelectronics. <http://www.st.com/content/ccc/resource/technical/document/datasheet/e9/be/53/a3/1f/6f/4f/75/CD00000449.pdf/files/CD00000449.pdf/jcr:content/translations/en.CD00000449.pdf>, mar. 2014.
- [13] Banggood.com. *Mini DC-DC Converter Step Down Module Adjustable Power Supply*. Nov. 2016. URL: <http://www.banggood.com/Mini-DC-DC-Converter-Step-Down-Module-Adjustable-Power-Supply-p-920327.html?rmmds=search>.
- [14] developer.android.com. *Processes and Threads*. Nov. 2016. URL: <https://developer.android.com/guide/components/processes-and-threads.html>.
- [15] Oracle. *Synchronized Methods*. Nov. 2015. URL: <https://docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html>.
- [16] GearHost. *GearHost*. Nov. 2016. URL: <https://www.gearhost.com/>.
- [17] Wikipedia. *Hypertext Transfer Protocol*. Nov. 2016. URL: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.
- [18] Margaret Rouse. *TCP (Transmission Control Protocol)*. Aug. 2014. URL: <http://searchnetworking.techtarget.com/definition/TCP>.
- [19] Wikipedia. *Bluetooth*. Nov. 2016. URL: <https://en.wikipedia.org/wiki/Bluetooth>.

- [20] developer.android.com. *AndroidStudio*. Nov. 2016. URL: <https://developer.android.com/reference/android/bluetooth/BluetoothDevice.html>.
- [21] Simon Long. *Jessie is here*. Sep. 2015. URL: <https://www.raspberrypi.org/blog/raspbian-jessie-is-here/>.
- [22] Lawrence Livermore National Laboratory Blaise Barney. *POSIX Threads Programming*. Jun. 2016. URL: <https://computing.llnl.gov/tutorials/pthreads/>.
- [23] developer.android.com. *AndroidStudio*. Nov. 2016. URL: <https://developer.android.com/studio/index.html>.
- [24] Postman. *Postman*. Sep. 2016. URL: <https://www.getpostman.com/>.

AARHUS UNIVERSITY SCHOOL OF ENGINEERING

AUTONOM FORFØLGELSESDRONE MED SMARTPHONE APPLIKATION

BACHELORPROJEKT
ELEKTROINGENIØR

Procesrapport

Projekt nr: 16114

Jonas Risager Nielsen - 201270337

Benedikt Wiese - 201310362

Mathias Poulsen - 201370945

Vejleder

Torben Gregersen

Aarhus University School of Engineering

15. december 2016

Versionshistorie for Procesrapporten

Version	Dato	Beskrivelse
0.0	28.11.2016	Opsætning af dokumentet.
0.1	29.11.2016	Tilføjelse af Møder.
0.2	30.11.2016	Tilføjelse af Gruppeddannelse, Samarbejds aftale, Udviklingsforløb, Projektledelse, Arbejdesfordeling, Planlægning, Projekt administration, Opnåede erfaringer, Fremtidigt arbejde.
0.3	31.11.2016	Tilføjelse af Konklusion på udviklingsprocessen, og Konklusion fra projektgruppen.
0.4.0	01.12.2016	Revideret af gruppen.
0.4.1	05.12.2016	Rettelser efter vejledermøde.
0.5	07.12.2016	Ordforklaring tilføjet.
0.6	14.12.2016	Revideret.
1.0	15.12.2016	Aflevering.

Ordforklaring

Forkortelse	Forklaring
3G	3. Generations mobilnetværk.
ASE	Aarhus University School of Engineering.
Shadow BAC-X1	Navnet på dronen, der udvikles i dette projekt.
SQL	Structured Query Language.

Indholdsfortegnelse

Kapitel 1	Indledning	2
Kapitel 2	Projektgennemførelse	3
2.1	Gruppedannelse	3
2.2	Samarbejdsaftale	3
2.3	Udviklingsforløb	4
2.3.1	ASE-modellen	4
2.3.2	Scrum	4
2.4	Projektledelse	5
2.5	Arbejdsfordeling	5
2.6	Planlægning	6
2.7	Projekt administration	7
2.8	Møder	7
2.9	Opnåede erfaringer	8
2.10	Fremtidigt arbejde	8
Kapitel 3	Konklusion	9
3.1	Konklusion på udviklingsprocessen	9
3.2	Konklusion fra projektgruppen	9

Forord

Denne rapport omhandler udviklingsprocessen af bachelorprojektet *Autonom forfølgelses drone med smartphone applikation* udarbejdet på Aarhus University School of Engineering. Rapporten blev udarbejdet af Mathias Poulsen, Benedikt Wiese og Jonas R. Nielsen. Under projektforløbet var alle tre studerende på diplomingeniøruddannelsen; Elektronikingeniør. Projektoplægget blev udarbejdet af gruppens medlemmer.

Tilhørende til denne procesrapport er en projektrapport og bilag. Projektrapporten ligger umiddelbart før procesrapporten og omhandler selve projektets forløb. Derudover er der vedlagt en række dokumenter i bilag, der indeholder blandt andet mødeindkaldelser, møderefereater og tidsplan.

Lektor Torben Gregersen var vejleder gennem projektet. Projektet blev afleveret den 16. december 2016, og bliver bedømt ved en mundtlig eksamen den 13. januar 2017.

Indledning 1

Procesrapporten handler om udviklingsprocessen i projektet. Gennem projektet blev forskellige metoder benyttet for at strukturere arbejdet. Dette førte til en mere målrettet udvikling i projektet.

ASE-modellen [1] blev benyttet iterativt. Elementer af Scrum [2] blev brugt til at strukturere projektets iterationer og opbygning. I projektet blev der lagt vægt på at køre en agil proces, fælles beslutninger og en struktureret arbejdsdag.

Denne procesrapport beskriver, hvordan de enkelte metoder blev benyttet gennem projektet. Derudover beskrives eventuelle afvigelser fra metoderne. Procesrapporten afsluttes med en konklusion på metoderne brugt i projektet og en konklusion fra de enkelte gruppemedlemmer.

Tabel 1.1 viser ansvarsfordelingen i projektets proces.

Ansvar	Jonas	Mathias	Benedikt
Scrum Master			X
Udviklingsteam	X	X	X
Product Owner	X	X	X
BurndownChart	X		
Referant		X	

Tabel 1.1: Opdeling af procesmæssige ansvarsområder.

Projektgennemførelse 2

I dette kapitel vil hele projektgennemførelsen blive beskrevet. Derunder indgår hvordan gruppen blev sammensat, hvilke metoder blev benyttet i projektet og effekten af disse.

2.1 Gruppedannelse

Til bachelorprojektet var det de studerendes eget ansvar at danne en projektgruppe. Ingeniørhøjskolen anbefalede grupper i størrelsesordenen 2-3 personer. Denne projektgruppe blev dannet på baggrund af tidligere projektsamarbejde udført gennem studiet. Gruppen havde tidligere udarbejdet tre projekter fordelt på tre semestre.

Dette bevirkede, at gruppen allerede inden projektets start havde en fælles forventning og indstilling til projektet. Dette medførte, at gruppen arbejdede godt sammen. Selvom gruppens medlemmer alle var elektroingeniørstuderende, var specialområderne forskellige. Dette førte til at gruppen samlet set havde en bred forståelse for både hardware og software.

2.2 Samarbejdsaftale

I projektet blev en samarbejdsaftale underskrevet af alle gruppens medlemmer. Ingen af medlemmerne havde før benyttet en samarbejdsaftale. Denne aftale sikrede, at alle var indforstået med forventningerne til arbejdsbyrden og projektets gennemførelse.

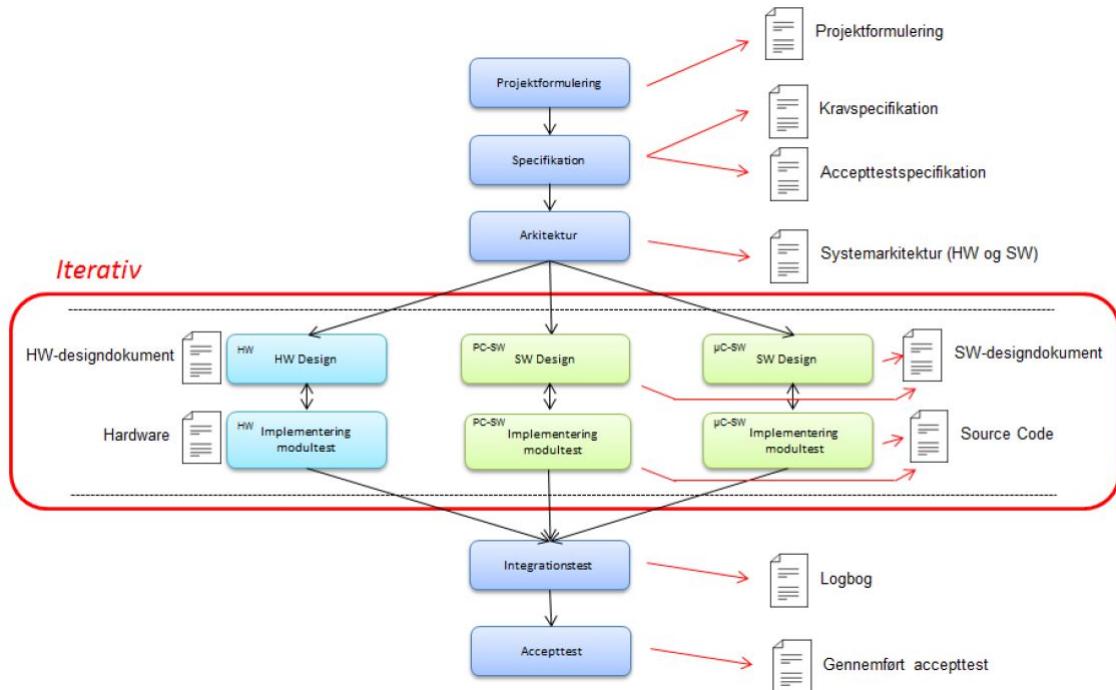
Samarbejdsaftale er vedlagt i bilag.

2.3 Udviklingsforløb

I dette afsnit vil metoderne i projektet blive forklaret. Hvordan disse blev brugt og hvordan de bidragede til projektet.

2.3.1 ASE-modellen

Projekter på Ingeniørhøjskolen udvikles baseret på ASE-modellen [1]. Figur 2.1 viser modellens opbygning.



Figur 2.1: ASE-modellen

Det blev valgt, at tilføje en analyse til modellen, for at belyse forskellig valg taget i projektet. Analysen blev udarbejdet sideløbende med kravspecifikationen, men blev anbragt efter kravspecifikationen i bilag. I design- og implementeringsfaserne blev en iterativ proces anvendt. Dette gjorde udviklingen mere fleksibel og sikrede en bedre kvalitet af det endelige produkt, da eventuelle fejl kunne rettes i en senere iteration. Modellen kan benyttes til både udvikling af hardware- og softwareprojekter.

2.3.2 Scrum

Scrum er en agil udviklingsproces med fokus på projektledelse, som har eksisteret siden 1990'erne [2]. Forskellen på Scrum og andre metoder som f.eks vandfaldsmodellen [3] er, at Scrum ikke er en liniær men en iterativ proces bestående af Sprints.

Scrum bygger på en Scrum Master, Product Owner og et Development Team. Product Owner'en er ansvarlig for produktet og leverancer til kunden. Han er derudover ansvarlig for Product Backlog'en. I Product Backlog'en inddeltes arbejdsopgaverne således, at Development Team'et har den bedste chance for at nå målet. Scrum Master'en er

ansvarlig for at sikre, at alle har forstået Scrum og at intet unødvendigt forstyrrer udviklingsprocessen. Development Team'et er udviklingsteamet, der står for, at udføre opgaverne, defineret til det enkelte Sprint.

I dette projekt agerede hele gruppen som både Development Team og Product Owner, da gruppen selv definerede projektet. Benedikt agerede som Scrum Master gennem projektet.

2.4 Projektledelse

Projektets ledelse var et samarbejde mellem gruppens medlemmer. Både krav og vigtige beslutninger blev truffet i fællesskab. Det var derfor også gruppens samlede ansvar, at definere kravene til systemet.

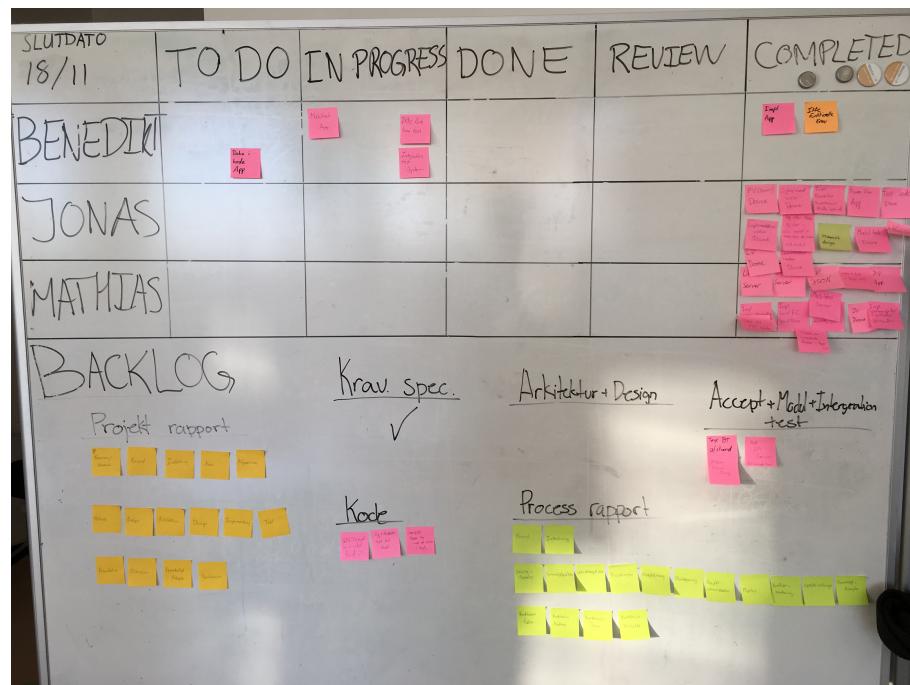
For at skabe en bedre struktur på udviklingsprocessen, blev Benedikt udnævnt som Scrum Master. Han fik til opgave at sikre, at de daglige møder blev afholdt, samt at Scrum Board'et blev opdateret. Det var derudover hans ansvar at sikre, at Sprint's og de dermed forbundne review- og planlægningsmøder blev over-/afholdt.

2.5 Arbejdsfordeling

Projektet blev opdelt i tre store blokke: dronen, applikationen og serveren. Alle gruppens medlemmer valgte bevidst, at arbejde med teknologier som de ikke havde erfaringer med i forvejen, for at kunne udvikle sig. Jonas arbejdede med serveropsætningen og udviklingen af systemets database i C# og SQL. Efterfølgende arbejdede han med på dronens kode. Mathias arbejdede på dronens Raspberry Pi. Han havde før arbejdet i C++, dog aldrig på en Raspberry Pi. Benedikt udviklede applikationen, som han tog et kursus i sideløbende med projektforløbet. Inden da havde han ikke arbejdet med et højniveau programmeringssprog som Java. Ingen af gruppemedlemmerne havde før arbejdet med droner eller autonome systemer. Selv om hovedansvarsområderne blev fordelt mellem gruppens medlemmer blev de vigtige beslutninger taget i fællesskab. Dette førte til, at alle medlemmer fik et overordnet overblik over alle systemets blokke.

Ud over den faglige arbejdsfordeling, havde hver gruppemedlem et processrelateret ansvar. Jonas var ansvarligt for Burndowncharts og Mathias var referent. Som nævnt tidligere agerede Benedikt som Scrum Master.

For at holde et overblik over Sprint'ets opgaver, blev et Scrum Board benyttet. Derudover kunne dette værktøj bruges til at se, hvad de enkelte medlemmer arbejdede på i øjeblikket og hvad der eventuelt skulle reviewes. Et billede af Scrum Board'et kan ses på figur 2.2.



Figur 2.2: Scrum Board

Projektgruppen var meget tilfreds med arbejdsfordeling. Opgavemængden var passede på de forskellige blokke til et halvt års arbejde.

2.6 Planlægning

Projektforløbet blev opdelt i fem Sprint's. Skriveperioderne i starten og slutningen af projektet udgjorde to Sprint's. Varigheden af de to Sprint's var henholdsvis to og fire uger. Udviklingsforløbet i midten af perioden blev inddelt i tre større Sprint's med en varighed på tre uger hver. Disse Sprint's blev opsat med baggrund i systemets Use Case's. Projektets Sprint's fremgår af tabel 2.1.

Sprint	Varighed	Målsætning
1	2 uger	Færdig kravspecifikation. Opsætning af drone. Færdig hardwarearkitektur og -design. Færdige domænemodeller.
2	3 uger	Udvikle funktionaliteten samt dokumentation til Use Case 1, 2, 7 og 8. Dette indebærer 3G og Bluetooth kommunikation mellem dronen, serveren og applikationen.
3	3 uger	Udvikle funktionaliteten samt dokumentation til Use Case 3, 6, 9 og 11. Dette indebærer TakeOff- og Land-kommandoerne både gennem 3G og Bluetooth.
4	3 uger	Udvikle funktionaliteten samt dokumentation til Use Case 4, 5 og 10. Dette indebærer SetPosition- og Follow-kommandoerne både gennem 3G og Bluetooth.
5	4 uger	Udførelse af accepttest med vejleder. Færdig projekt- og procesrapport. Aflevering af projektet.

Tabel 2.1: Projektets Sprints

Hvert Sprint i udviklingsforløbet endte ud i et produkt, som var muligt at teste for at sikre funktionaliteten i projektet.

2.7 Projekt administration

I projektets Sprint's blev en Sprint Backlog benyttet til, at holde styr på opgaverne tilknyttet Sprintet. Et Scrum Board var tilknyttet til at monitorere processen af de enkelte opgaver i Sprint Backlog'en. Et Burndownchart blev igennem processen brugt til at vurdere, hvorvidt tidsestimeringen af opgaverne var realistiske. Denne information kunne bruges i tidsestimeringen af de efterfølgende Sprint's.

Sprint Review's blev brugt til at evaluere resultaterne efter et Sprint i udviklingsperioden. Dette blev gjort med henblik på at sikre funktionaliteten inden næste Sprint.

Igennem projektet blev mange aspekter af Scrum implementeret dog ikke dem alle. Gennem forløbet blev der ikke udført Sprint Retrospective. Daily Scrum samt Sprint Review blev udført i tilpassede udgaver. Derfor ville det ikke være korrekt at sige, at der blev benyttet Scrum, dog blev elementer af Scrum implementeret og brugt i projektets gennemførelse.

2.8 Møder

Projektgruppen afholdt, så vidt muligt, møder med vejlederen hver uge. Disse møder blev brugt til at drøfte eventuelle spørgsmål, der opstod i løbet af ugen med projektarbejdet.

Forinden hvert møde blev en mødeindkaldelse sendt til vejlederen. Mødeindkaldelsen indeholdt en agenda for, hvad der skulle drøftes på mødet. Derudover var de dokumenter, som vejlederen ønskes at kigge igennem, vedhæftet.

Da projektet blev gennemført med brugen af elementer fra Scrum, blev der indlagt et Daily Scrum hver morgen. Disse møder blev brugt til, at hvert gruppemedlem kunne fortælle, hvad vedkommende havde været i gang med, og hvor langt han var nået. Projektgruppen sad i det samme lokale gennem projektforløbet. Derfor var der en løbende dialog imellem gruppemedlemmerne.

I starten af hvert Sprint, blev der afholdt et længere møde kaldet Sprint Planning. De opgaver, der skulle klares for at opnå Sprint Goal, blev opstillet og estimeret her.

Efter hvert Sprint blev der afholdt et Sprint Review, hvor resultatet af Sprint'et blev gennemgået. Ikke løste- og nyfundne opgaver blev tilføjet til Product Backlog'en. Derved kunne disse opgaver løses i et efterfølgende Sprint.

For et mere detaljeret overblik over de enkelte møder, henvises til mødeindkaldelserne og referaterne vedlagt i bilag.

2.9 Opnåede erfaringer

Igennem dette projekt udvidede alle gruppens medlemmer deres viden omkring brugen af Scrum i et større projekt. Samarbejdet var i fokus i gruppen. Dette gav en del erfaringer i forhold til, at man var en gruppe og ikke en individuel person med ansvaret for projektet alene.

Projektet var en erfaringsrig proces og alle medlemmerne mener, at de havde fået erfaringer inden for både produktudvikling og projektstyring. Gruppen erfarede værdien af en struktureret arbejdsproces gennem brugen af de beskrevne metoder. Opdelingen af projektet i iterationer gjorde tilgangen til opgaverne nemmere. Scrum elementerne var med til at gøre processen lærerig og effektiv.

2.10 Fremtidigt arbejde

Udviklingsprocessen i fremtiden skulle indeholde de resterende elementer fra Scrum. Scrum er først implementeret, når alle delene af Scrum benyttes. Dette kunne trække udviklingsprocessen til et højere niveau og dermed gøre denne mere effektiv.

Tidsestimeringen i udviklingen skulle forbedres, da tidsforbruget på en opgave i visse tilfælde endte med at være langt fra den estimerede tid.

Konklusion 3

Dette afsnit indeholder en fælles konklusion, samt individuelle konklusioner fra gruppe-medlemmerne.

3.1 Konklusion på udviklingsprocessen

Generelt var udviklingsprocessen god, og interne deadlines blev overholdt. Gruppen arbejdede godt og effektivt sammen, hvilket resulterede i, at alle mål blev nået.

Udviklingsprocessen med Scrum elementer var med til, at give gruppen en struktureret arbejdesmetode igennem hele projektets forløb. Dette havde en positiv indvirkning på resultatet af projektet. Opsætningen af Sprint's i projektet fungerede efter hensigten. I det primære udviklingforløb ledte dette hen til et færdigt produkt, som kunne testes.

ASE-modellen var en metode gruppemedlemmerne kunne støtte sig til, da denne var kendt gennem skolen. Dette var med til, at give processen en god struktur. Modellen var med til at give overblik over processen og sikre en veludført dokumentation af projektet.

Samarbejdet i gruppen var fuldt tilfredsstillende, da gruppemedlemmernes forventninger var ens. Alle i gruppen leverede en eksemplarisk arbejdsindsats.

3.2 Konklusion fra projektgruppen

Jonas

Jeg synes, at udviklingsprocessen fungerede utrolig godt gennem projektforløbet. Scrum og ASE-modellen gav os tilsammen de værktøjer, vi skulle bruge, for at strukturere vores projektforløb fornuftigt.

Udarbejdelsen af analysen var med til at give projektet en god start, da der blev gjort mange gode tanker omkring projektet. Det var en fornøjelse, at benytte Scrum til at estimere opgaver og samle funktionalitet i Sprint's. Brugen af et Scrum Board gav et godt overblik over de enkelte Sprint's gennem forløbet.

Alt i alt var jeg utrolig tilfreds med udviklingsforløbet.

Mathias

Projektforløbet blev gennemført ved brug af elementer af Scrum og ASE-modellen. Brugen af Scrum gav mig rigtig stor erfaring i forhold til, at styre et projekt med en iterativ proces. Det fungerede særdeles godt sammen med den arbejdsindsats, som gruppen lagde i projektet.

Alt i alt fungerede hele udviklingsprocessen igennem projektforløbet meget godt, hvilket medførte at alle deadlines blev nået. Dette var jeg yderst tilfreds med.

Benedikt

Projektets udviklingsprocess med Scrum og ASE-modellen gjorde, at der igennem hele projektforløbet blev arbejdet meget struktureret og målrettet. Især Scrum var med til at øge arbejdslysten igennem sin iterative proces. Den iterative process førte til en god afveksling mellem implementering og dokumentation.

At dele projektet op i Sprint's førte til, at der var mange deadlines undervejs, der skulle overholdes. Det var nemmere at overholde tidsplanen og lave integrationstests af de nye funktionaliteter, da alle blokke blev færdig på samme tid. Derudover sorgede Scrum Board'et og de daglige møder for, at alle gruppens medlemmer på ethvert tidspunkt vidste, hvad projektets overordnede status var.

References

- [1] Ingeniørhøjskolen Aarhus Universitet. “Vejledning til dokumentation af semesterprojekter”. Version 1.1. Nov. 2016.
- [2] Jeff Sutherland og Ken Schwaber. “The Scrum Guide”. Developed and sustained by Ken Schwaber and Jeff Sutherland. Jul. 2016.
- [3] LederIndsigt. *Vandfaldsmodellen*. Dec. 2016. URL: <http://lederindsigt.dk/begreber/ledelsesbegreber/v/vandfaldsmodellen-waterfall-model/>.