Ingeniørhøjskolen Aarhus Universitet 2014

Projektrapport

Autonom overvågningsdrone

Rasmus Lydiksen	Anders Opstrup	Kevin Grooters
11647	11726	11655

Ingeniørhøjskolen
Aarhus Universitet
Finlandsgade 22
8200 Aarhus N

Projekt titel:

Autonom overvågningsdrone

Projekt:

Bachelorprojekt

Projektperiode:

August 2014 - December 2014

Projektgruppe:

14123

Gruppemedlemmer:

11647 – Rasmus T. Lydiksen
11762 – Anders H. Opstrup
11655 – Kevin Grooters

Vejleder:

Torben Gregersen

Samlet sidetal: 39
Projekt afsluttet: 17-12-2014

Resumé og Abstract

Abstract

The purpose of this bachelor project is to develop an autonomous surveillance drone which at low cost can monitor a given area. The idea is to develop an alternative to expensive and resource demanding surveillance performed by humans. This project is designed and developed on Aarhus school of engineering.

A quadrocopter has been built into a surveillance drone. The user will be able to create a set of coordinates on a webapplication, the drone will then autonomously fly to the given coordinates and monitor that area. The drone consists of an Aeroquad ARF quadrocopter, a GPS, height sensors and a mobile communication module. The user can view previous flights and create a new flight setup through the webapplication.

During the project, a full functioning server, the communication link and exchange of data between server and the drone was successfully implemented. In addition, data from the GPS, compass and height sensors was collected and processed. Furthermore, the main part of the webapplication and the autonomous flight system was implemented.

Resumé

Formålet med dette bachelorprojekt er at udvikle en overvågningsdrone, der nemt og omkostningsfrit kan overvåge et område. Dronen udvikles for at skabe et alternativ til dyr og ressourcekrævende overvågning udført af mennesker. Projektet er designet og udviklet på Ingeniørhøjskolen Aarhus Universitet.

I projektet omdannes en fjernstyret quadrocopter til en drone, som skal bruges til overvågning. Ud fra brugers anvisninger skal dronen autonomt kunne flyve til GPS positioner og tage overvågningsbilleder.

Den udviklede drone består af en Aeroquad ARF quadrocopter med påmonteret GPS, højdemåler og mobilt kommunikationsmodul. Systemets bruger indstiller droneflyvning samt monitorerer data fra tidligere flyvninger via en webapplikation.

I projektforløbet lykkedes det at implementere en fuldt fungerende server, samt kommunikationslink og udveksling af data mellem server og drone. Desuden blev opsamling af data fra GPS, kompas og højdemåler fuldt ud implementeret. Det lykkedes endvidere at implementere hovedparten af webapplikationen og dronens autonome flyvning.

Indholdsfortegnelse

Intro 1

1.1 Revisionshistorik

Rev. Nr	Dato	Initialer	Ændring
1.0	1-12-2014	AO, KG, RL	Oprettet dokument.
1.1	1-12-2014	RL	Tilføjet resume, indledning, forord og systembeskrivelse.
1.2	1-12-2014	KG	Tilføjet, opgaveformulering, krav og projektafgrænsning.
1.3	1-12-2014	AO	Tilføjet udviklingsproces. udviklingsproces.
1.4	8-12-2014	KG	Tilføjet systemarkitektur i projektbeskrivelse..
1.5	8-12-2014	RL	Tilføjet design i projektbeskrivelse.
1.6	8-12-2014	AO	Tilføjet implementering i projektbeskrivelse.
1.7	10-12-2014	AO, KG, RL	Tilføjet test, resultater og diskussion af resultater.
1.8	12-12-2014	AO, KG, RL	Tilføjet konklusion.

Tabel 1.1: Revisionshistorik

1.2 Ordforklaring

Forkortelse	Betydning	Forklaring
Drone	Drone	Aeroquad ARF quadrocopter og påmonteret hardware.
Webapp	Webapplikation	Applikation som bruges til opsætning af nye flyvninger og monitorering af tidlige flyvninger.
Server	Server	Server dækker over systemets database og tilhørende logik.
Main controller	Main controller	Main controller er en arduino mega2560. Main controlleren styrer dronen ud fra flyveopsætning og data fra diverse sensorer.
Flight control board	Flight control board	Boardet er købt sammen med quadrocopter hos aeroquad. Boardet bruges som mellemled mellem main controller og dronens motorer.
3G/GPS	3G/GPS modul	Dette modul er et 3G/GPS shield der påmonteres main controller.
bdd	Block definition diagram	bdd bruges til at vise systemets overordnede hardware blokke.
ibd	Internal block diagram	ibd bruges til at vise ekstern og intern kommunikation tilhørende en eller flere blokke.
sd	Sekvens diagram	Diagrammet bruges til at vise afvikling og timing af forskellige processer.

Tabel 1.2: Ordforklaring

1.3 Arbejdsfordeling

Projektgruppen har i fællesskab udarbejdet kravspecifikation, systemanalyse, systemdesign og accepttest. Nedenfor specificeres hvilke arbejdsområder og arbejdsopgaver projektgruppens medlemmer særligt har beskæftiget sig med.

Drone

Gruppemedlem:

- Rasmus Lydiksen
- Kevin Grooters

Arbejdsopgaver:

- GPS (Kevin)
- 3G kommunikation (Kevin)
- Switch board (Kevin)
- Motor styring (Rasmus)
- Ultralyds sensor (Rasmus)
- Kompas (Rasmus)
- Styrings regulering (Rasmus)

Server og webapplikation

Gruppemedlem:

- Anders H. Opstrup

Arbejdsopgaver:

- Opsætning server
- SQLite database
- Webapplikation

Forord 2

Dette projekt blev tilbudt som et bachelorprojekt af Ingeniørhøjskolen Aarhus Universitet. Grundet tidens stigende fokus på droner ønskede Ingeniørhøjskolen det undersøgt hvorvidt det var muligt at udvikle en drone.

Der var i udgangspunktet ingen begrænsninger for hvad bachelorprojektet kunne eller skulle indeholde. Derfor har gruppen på egen hånd opsat krav, designet og udformet bachelorprojektet *Autonom overvågningsdrone*.

Projekt, projektrapport og tilhørende dokumentationen er udarbejdet af projektgruppe 14123, bestående af tre ingeniørstuderende. Gruppen er blevet bevilliget økonomisk støtte til indkøb af drone, sensorer og andet hardware. Ydermere har gruppen fået vejledning og rådgivning af en vejleder tilknyttet fra Ingeniørhøjskolen Aarhus Universitet.

Referencer

Betegnelsen [x] hentyder til en reference, som står i referencelisten sidst i rapporten.

Indledning 3

I den moderne verden findes der mange forskellige former for overvågning, og overvågning foregår overalt. Primært bruges overvågning til at skabe tryghed og forebygge kriminalitet, hvilket de fleste mennesker er positivt stemt overfor. Men pga. en stigning i brugen af overvågning, bruges der gradvist flere og flere ressourcer på området. I projektet undersøges det, hvorvidt det er muligt at udvikle en autonom overvågningsdrone med dertilhørende webapplikation og server. Det er gruppens mål at udvikle en drone, der ud fra brugers anvisninger kan overvåge og tage billeder af et defineret område. Dronen udvikles for at mindske mængden af menneskelige ressourcer der bruges på overvågning.

Dronen gøres i stand til at orientere sig om egen GPS position, flyvehøjde og flyveretning samt at kommunikere med en server. Gennem kommunikation med server henter dronen flyveopsætning, som systemets bruger har oprettet via webapplikationen. Efter at hentet flyveopsætning, flyver dronen på egen hånd ud og overvåger et defineret område. Projektet er planlagt, designet og konstrueret hen over en periode på 4 måneder.

Projektrapporten er sat op efter en standard, hvor systemet indledningsvis beskrives på et abstrakt niveau. Herefter beskrives krav til produktet, hvor der gås mere analytisk og teknisk til værks. Rapporten afsluttes af med resultater, diskussion af resultater samt en konklusion.

Opgaveformulering 4

Hovedformålet med projektet er at udvikle en drone, der autonomt kan overvåge et område. Overvågningen skal foregå fra luften, hvor dronen flyver mellem forskellige GPS positioner som den overvåger og tager billeder af. Området som dronen skal overvåge indstilles af systemets bruger via en webapplikation.

Systembeskrivelse 5

På Ingeniørhøjskolen Aarhus Universitet forefindes en AeroQuad ARF Quadrocopter. Målet med projektet er at omdanne quadrocopteren til en autonom overvågningsdrone.

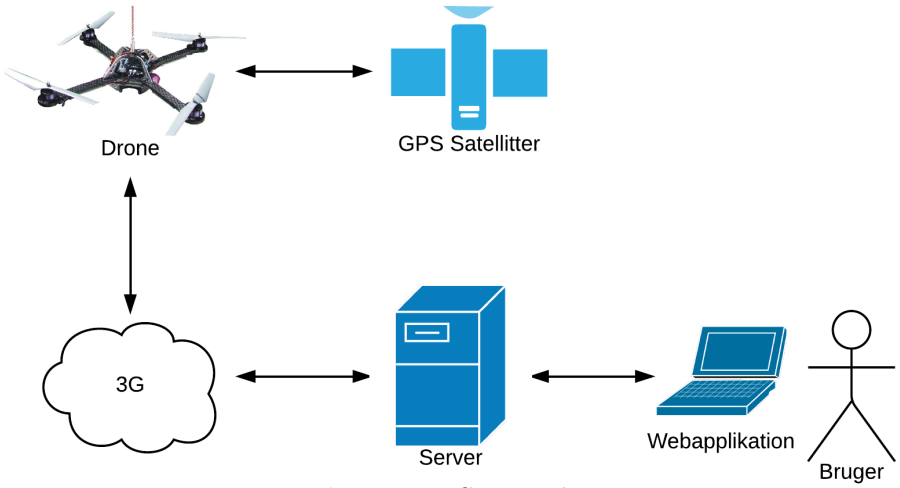
Dronen skal ud fra brugers anvisninger overvåge og tage billeder af et defineret område. Dronen tilgås via en webapplikation, der fungerer som en grafisk brugerflade mellem bruger og systemet. Via webapplikationen kan bruger oprette flyveopsætninger til drone samt se billeder og flyverute fra tidligere flyvninger. Når der laves ny flyveopsætning vælger bruger en række GPS positioner som dronen skal flyve til, flyvehøjde og hvorvidt der skal tages billeder ved de valgte GPS positioner. Når bruger har lavet en ny flyveopsætning, stilles flyveopsætningen tilgængelig for dronen på server.

Da dronen skal flyve autonomt, er det vigtigt at den kan orientere sig på egen hånd. Derfor er dronen udstyret med GPS, afstandssensorer og kompas. Til enhver tid skal kommunikation mellem drone og server foregå via mobilt netværk. Der gøres hovedsageligt brug af 3G netværket, men i områder med dårlig forbindelse vil der blive gjort brug af 2G som fall-back netværk.

Systemskitse

Bruger benytter en computer til at tilgå webapplikation og lave en ny flyveopsætning. Når bruger har lavet en ny flyveopsætning, overføres flyveopsætningen via Internettet til server, hvor den gøres tilgængelig for dronen.

Via det mobile 3G netværk kommunikerer drone med server. Inden flyvning påbegyndes henter drone flyveopsætning fra server og under flyvning sender dronen information om nuværende GPS position og overfører billeder til server. Under flyvning kontrollerer drone løbende egen GPS position via kommunikation med GPS satellitter. Dette gør den for at opdatere egen position og for efterfølgende at kunne beregne den korrekte flyveorientering.



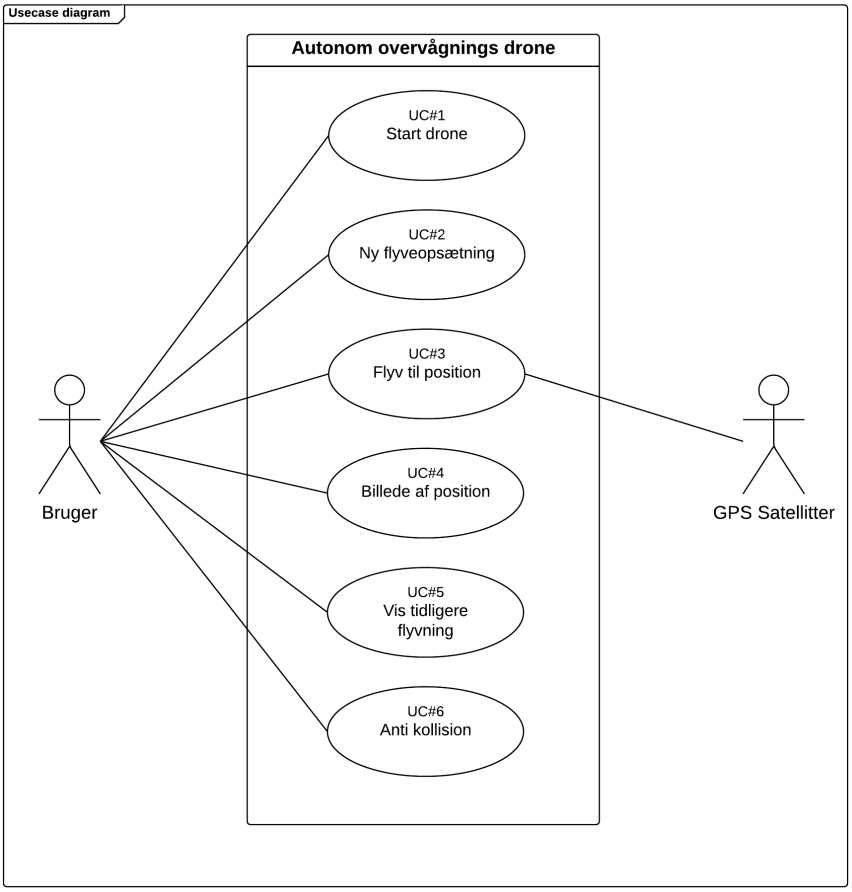
Figur 5.1: Systemskitse

Krav 6

I dette afsnit beskrives kravene til projektet [1]. Indledningsvis præsenteres funktionelle krav ved brug af use cases. På figur 6.1 vises et use diagram, der viser aktører og use cases tilhørende systemet. Efter use case diagrammet følger en kort beskrivelse af de seks use cases og deres funktion. Afsnittet afsluttets med ikke-funktionelle krav, der både er krav til systemet som helhed og til systemets blokke.

6.1 Funktionelle krav

Til systemet er der identificeret de 2 aktører: Bruger & GPS-satellitter. Bruger er primær aktør der ønsker at initialisere og styre systemet. Bruger er ansvarlig for at oprette flyveopsætninger på webapplikationen og tilslutte batteri til dronen. GPS-satellitter er sekundær aktør, der gør det muligt for dronen at finde egen position. På figur 6.1 vises et use case diagram, der viser systemets aktører og i hvilke use cases aktørerne optræder.



Figur 6.1: Use case diagram

I projektets kravspecifikation er alle use cases beskrevet som fully dressed use cases. I fully dressed use cases beskrives hvordan hovedforløb ser ud i en given use case, hvis den succesfuldt gennemføres. Desuden beskrives startbetingelse, aktører og tilføjelser.

Nedenfor beskrives hovedforløb og funktion af de seks use cases:

Use case 1: Start drone
Bruger tilslutter batteri til drone og dronen initialiseres. Drone sender information om nuværende position til server.

Use case 2: Ny flyveopsætning
Bruger logger på webapplikation og opretter en ny flyveopsætning. Flyveopsætningen sættes tilgængelig for drone på server.

Use case 3: Flyv til position
Drone henter flyveopsætning fra server og påbegynder flyvning. Under flyvning tilpasser drone løbende flyvehøjde og flyveorientering og fortsætter flyvning mod ønsket position.

Use case 4: Billede af position
Når drone er ankommet til ønsket GPS position tages et billede. Hvis billedet accepteres, flyver dronen videre mod næste GPS koordinat eller til udgangsposition.

Use case 5: Vis tidligere flyvninger
Bruger tilgår webapplikation for at se flyveruter og billeder fra tidligere flyvninger.

Use case 6: Antikollision
Drones antikollisionssensorer bruges til at detektere forhindringer. I tilfælde af forhindringer ændrer dronen enten flyveretning eller flyvehøjde for at undgå kollision.

6.2 Ikke-funktionelle krav

De ikke-funktionelle krav opstilles ud fra systemspecifikationerne og er krav som ikke kan defineres i use cases. Disse krav har ingen indvirkning på systemets endelige funktion, kun systemspecifikationer.

De ikke-funktionelle krav opstilles i fem grupper. Nogle krav er generelle krav til systemet, mens andre krav mere specifikt omhandler drone, server og webapplikation eller dataopsamling. De ikke-funktionelle krav bruges til at sikre system performance, eksempelvis stilles krav til de hardwaremoduler og sensorer der er monteret på dronen.

Projektafgrænsning 7

Ud fra kravspecifikationen og systemskitsen figur 5.1 var der dannet en grundlæggende ide om hvordan systemet skulle se ud og fungere. Men for at blive i stand til at sætte realistiske mål for projektet, blev det besluttet at lave en afgrænsning af systemets funktionalitet. I dette afsnit beskrives nogle af de afgrænsninger gruppen har valgt at lave.

Video live feed og billedekvalitet
En af de første afgrænsninger der blev lavet, var afgrænsningen af kamera med høj billedkvalitet som kunne optage video. Planen om højkvalitets billeder og live stream blev ændret, fordi upload-hastigheden på på 3G / 2G netværket er begrænset, og gruppen var opmærksomme på det kunne blive en flaskehals. Idéen om live feed blev droppet og i stedet blev det besluttet at gøre brug af et kamera med lav-middel opløsning, som løbende skulle tage almindelige billeder.

Tilgang til webapplikation
I kravspecifikationen beskrives det, at ethvert device med Internet adgang skal kunne benytte webapplikationen. Men afhængigt af hvilket device der benytter webapplikationen, kræves en ny opsætning. Det blev besluttet kun at designe og implementere webapplikationen til brug via computere.

Flyvehøjde og tilhørende sensorer
Hvis dronen skulle flyve over længere distancer ville flyvehøjde over 5 meter helt klart være fordelagtigt. Primært ville det være en fordel at kunne flyve højere, fordi der normalt er færre forhindringer jo højere der flyves. Men der er bevidst valgt en teknologi, der maksimalt kan bruges til måling af afstande på 4-5 meter. Teknologien er valgt af to årsager. For det første fordi de anvendte ultralydssensorer er nemmere at håndtere og billigere. For det andet fordi gruppen ikke ønskede at flyve højere end 2-3 meter. Skulle noget gå galt, er det langt farligere hvis dronen falder fra 20 end 2-3 meters højde.

Projektbeskrivelse 8

I det følgende beskrives projektforløbet. Først beskrives hvilke processer og værktøjer der er anvendt. Derefter beskrives hvordan værktøjerne er anvendt. Til sidst beskrives hvilke resultater gruppen har opnået, og der laves en diskussion på baggrund af resultaterne. Afsnittet afsluttes med en konklusion.

8.1 Udviklingsproces

Projektforløbet er gennemført over en periode på 4 måneder, hvor der er blevet gjort brug af forskellige arbejdsmetoder og værktøjer. I dette afsnit beskrives hvorledes projektet er gennemført, hvilke metoder der er anvendt samt hvilke udviklingsværktøjer der er benyttet.

8.1.1 Gennemførelse

Under hele projektforløbet har alle gruppens medlemmer arbejdet tæt sammen. Specielt i starten af projektet hvor foranalyse, krav og indledende systemarkitektur blev udarbejdet, arbejdede gruppens medlemmer i tæt fællesskab. Da projektet bevægede sig over i detaljeret design og implementering, blev der gjort brug af iterativ udvikling, og gruppens medlemmer blev ansvarlige for forskellige dele. Foruden iterativ udvikling blev der også gjort brug af agil udvikling.

Iterativ udvikling

Stort set alle udviklingsprojekter forløber via vandfaldsmodel eller iterativ udvikling. Hvis vandfaldsmodellen benyttes gennemløber projektet en række faser, der hver for sig afsluttes, inden næste fase påbegyndes. Opbygning af vandfaldsmodellen kunne eksempelvis være: Krav, analyse, design, implementering og test. I et iterativt projektforløb gennemløber projektet i stedet en række iterationer, der hver for sig kan ses som miniudgaver af vandfaldsmodellen. Iterationer, der ligger tidligt i et projektforløb, vil ofte omhandle grundfunktionalitet, mens iterationer senere i projektforløbet bruges til at tilføje funktionalitet til systemet.

Da krav til systemets funktionalitet var beskrevet med use cases, var det oplagt at opdele detaljeret design og implementering i iterationer. Derfor blev detaljeret design og implementering opdelt i de fire iterationer, som beskrives nedenfor.

Iteration 1
I første iteration er fokus på systemets mest grundlæggende funktionalitet. Drone gøres i stand til at oprette forbindelse til server via 3G/GPS-modulet. Desuden tilsluttes batteri, ESC'er, motorer og ultralydssensor til drone. Målet med iterationen er at kunne gennemføre use case 1.

Iteration 2
I iteration 2 er hovedformålet at få kommunikation mellem server og drone til at fungere. Bruger skal kunne oprette nye flyveopsætninger og gøre dem tilgængelige på server for dronen. Ydermere skal drone kunne finde egen GPS position, flyvehøjde og orientering. Ud fra viden om egen position, flyvehøjde og orientering skulle dronen kunne flyve til lokationer som er forudbestemt af bruger. Efter denne iteration skal use case 2 og 3 kunne gennemføres.

Iteration 3
I iteration 3 er hovedformålet at tilføje billedehåndtering. Der monteres kamera på drone, så der kan tages billeder under flyvning. Billeder taget under flyvning sendes via mobilnet fra drone til server og gøres tilgængelige for bruger. Målet med iteration 3 er at kunne gennemføre use case 4 og 5.

Iteration 4
I iteration 4 er hovedformålet at udvikle antikollision til dronen. Inden tilføjelsen af antikollision kunne dronen udelukkende flyve i lukkede områder uden forhindringer. Tilføjelsen af antikollision skal muliggøre flyvning i normale områder med forhindringer. Efter denne iteration skal alle use cases kunne gennemføres.

Agil udvikling

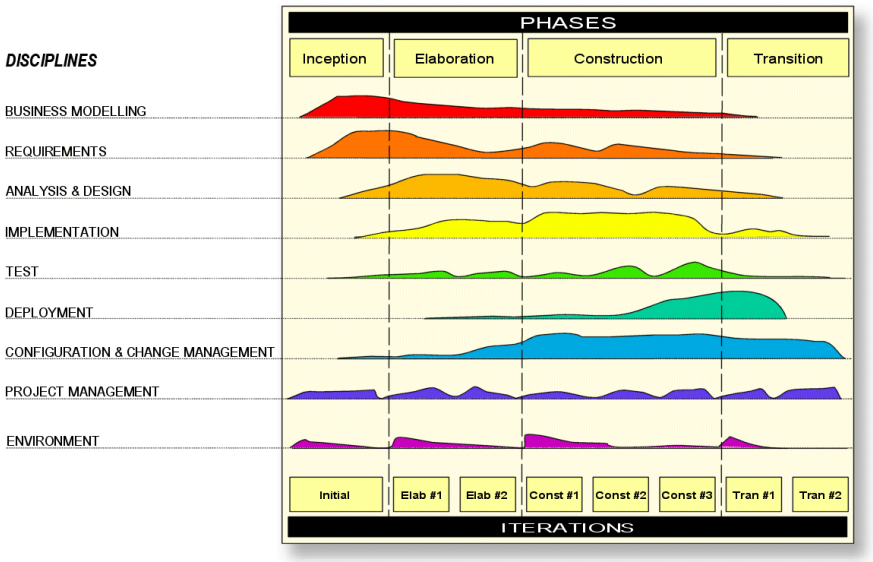
Agil systemudvikling har fokus på løbende at levere værdi til kunden gennem en fleksibel og omskiftelig udviklingsproces. Gennem brug af metoder som scrum light er udvikling og dokumentationen af projektetforløbet meget agilt. Under construction fasen blev der gjort brug af backlog og scrum meetings til at bevare overblik over projektets fremgang.

8.1.2 Metoder

I dette afsnit beskrives de metoder der er blevet benyttet i projektet.

RUP model

Rational **U**nified **P**rocess er en systemudviklings metode der er blevet anvendt til at definere den overordnede ramme for projektet. RUP er opbygget af de fire faser som beskrives nedenfor.



Figur 8.1: RUP model

Inception
Inception fasen er projektets indledende fase, og bruges til at lave forundersøgelse, valg af hardware/software samt opsætning af krav. I løbet af inception fasen udarbejdes foranalyse, indkøbsliste, krav til systemet og overordnet systemskitse.

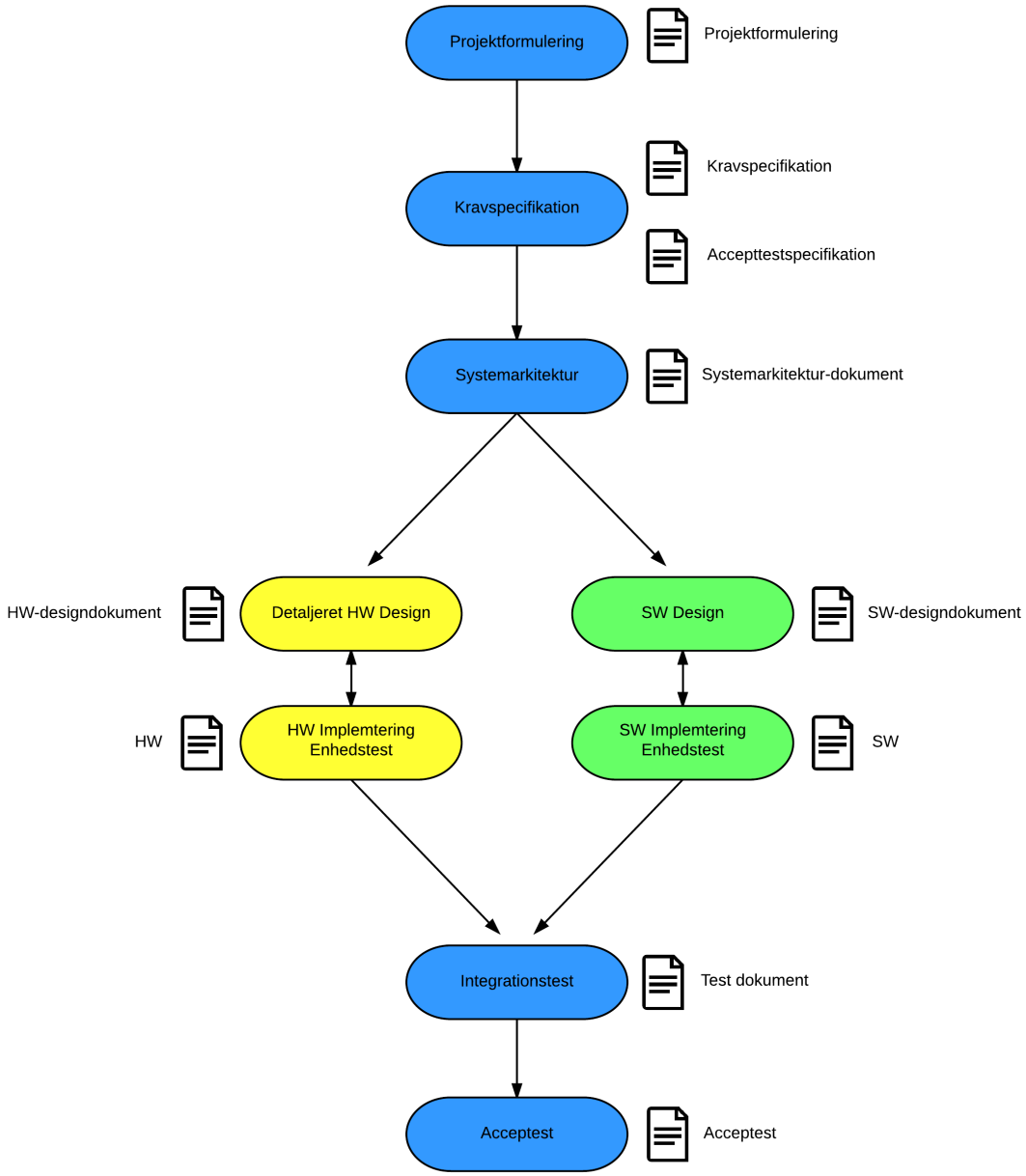
Elaboration
Elaboration fasen tager hånd om systemarkitektur og design. I denne fase udarbejdes der indledningsvis en domain model og derefter systemarkitektur af hardware og software.

Construction
I construction fasen er produktet i fokus, og der skiftes løbende mellem implementering af ny funktionalitet, test og dokumentation af nytilføjet funktionalitet.

Transition
Transition er den afsluttende fase, og tager hånd om færdiggørelse af projekt og overdragelse af produkt. Fasen bruges til lave accepttest, færdiggøre dokumentation og udarbejdelse af rapport.

ASE modellen

ASE modellen som ses på figur 8.2 er brugt i kombination med RUP. RUP bruges til at danne overordnede rammer for projektet, mens ASE modellen bruges til definere hvilke dokumenter der skal udarbejdes og hvornår. På figur 8.2 vises hvilke dokumenter der oprettes som produkt af ASE modellen.

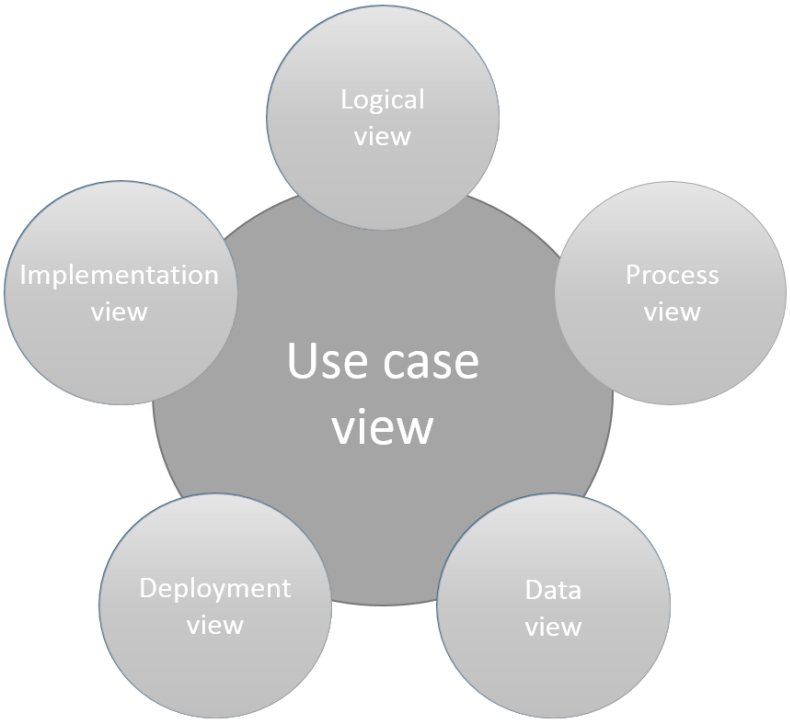


Figur 8.2: ASE modellen

N + 1 model

Alt software design i projektet er lavet ud fra systemudviklings modellen N + 1 [10]. Modellen er en udviklingsmodel, der beskriver software fra flere forskellige view's. Med N + 1 modellen tages der altid udgangspunkt i et use case view. Derefter er det op til udviklerholdet at bestemme hvor mange views der skal bruges for at beskrive systemet på tilstrækkelig vis.

Til beskrivelse af systemets software anvendes en 5 + 1 view model, som indeholder use case view, logical view, process view, data view, deployment view og implementation view. På figur 8.3 ses en illustration der viser 5 + 1 modellen og på den følgende side beskrives 5 + 1 modellens view's uddybende.



Figur 8.3: 5 + 1 model

Use case view
Use case viewet består af use case beskrivelser, der er udviklet ud fra brugers synspunkt og som bruges til at beskrive systemets funktionalitet og forskellige brugsscenarier. Alle udarbejdede use case beskrivelser forefindes i kravspecifikationen.

Logical View
Logical view bruges til at beskrive de logiske blokke i systemet. På baggrund af hver iteration, er der udarbejdet tilhørende designoverview-, pakke-, klasse-, sekvens- og state machine diagrammer. Diagrammerne bruges til at give et overblik over systemets funktionalitet på et mere detaljeret niveau.

Process View
Process view bruges til at beskrive sideløbende processer/tråde i systemet og hvordan samspillet imellem disse er. I view'et beskrives også krav til timing i kommunikation mellem drone og server.

Data View
I data viewet beskrives layout af data, der gemmes i systemet, og hvordan det lagres. Desuden beskrives hvordan data sendes rundt i systemet og hvordan servers database tilgås.

Deployment View
Deployment view beskriver systemets grundlæggende fysiske elementer og sammenspillet mellem dem. I view'et beskrives også hvilke software pakker der bruges i systemet og hvor de bruges. Desuden beskrives hvilke protokoller der er anvendt, fx. layout af meddelelser med header/start/stop.

Implementation View
Implementation view beskriver vigtige elementer i systemet som ikke er blevet beskrevet i andre views. Bla. beskrives hvilke værktøjer der er benyttet til projektet og hvordan disse værktøjer er sat op. Det beskrives også hvilke filer systemet er bygget af og hvordan disse filer skal linkes sammen.

8.1.3 Udviklingsværktøjer

I dette afsnit beskrives de udviklingsværktøjer [2] der er benyttet i løbet af projektet.

Latex
Latex er benyttet til udarbejdelse af dokumentation og rapport.

Git
Git er versionsstyring og fildelings værktøj.

Lucidchart
Lucidchart er benyttet til udarbejdelse af software diagrammer.

Visio
Visio er benyttet til udarbejdelse af hardware diagrammer.

Atmel studio 6.2
Atmel studio er det IDE hvor kode til main controller skrives.

Arduino IDE
Arduino IDE er brugt til at teste kode til main controller.

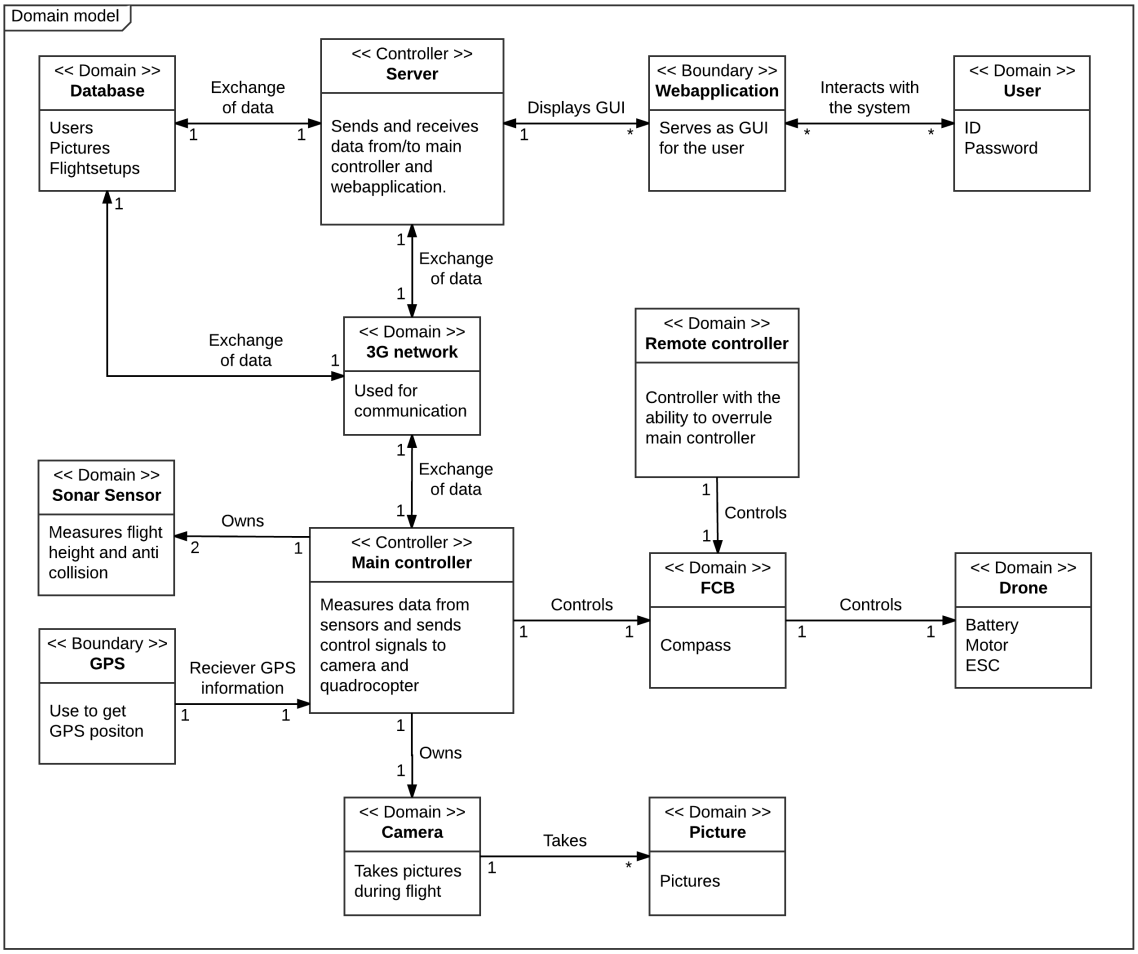
Atom
Atom er et IDE / tekstbehandlings program til håndtering kode.

OSX ternimal
Opsætning og programmering af server og webapplikation.

SQLite database
SQLite er benyttet til systemets database.

8.2 Systemarkitektur

Domain modellen anvendes som en overgang mellem kravspecifikation og systemarkitektur. I kravspecifikation beskrives hvad der sker ved interaktion med systemet. Mens systemarkitekturen bruges til at beskrive systemet i blokke og til at skitsere både interne og eksterne forbindelser. Domain modellen bruges til at beskrive hele systemets domæne. Der kigges ikke på hardware vs. software, der kigges i stedet på "enheder"og deres ansvarsområder.



Figur 8.4: Domain model

Efter at have udarbejdet kravspecifikation, systemskitse og domain model, var der formet en ide om hvordan systemet skulle fungere. I det følgende beskrives hvordan systemarkitekturen er udformet.

Systemarkitekturen er udført ved brug af SysML og UML diagrammer. SysML bruges til beskrivelse af systemets hardware i blokke og som samlet system. Ud fra N + 1 og applikations modellen udformes UML diagrammer, der bruges til beskrivelse af systemets software. Foruden SysML og UML diagrammer benyttes en række andre diagrammer, i det følgende beskrives nogle af de mest centrale diagrammer.

Use case
Use cases og tilhørende diagrammer er benyttet i projektforløbets indledende faser. De bruges til at definere systemets kunnen og opdele systemet i mindre dele. Use cases har i høj grad fungeret som et omdrejningspunkt, hvorfra alt funktionalitet udspringer.

SysML
Der er benyttet to slags SysML diagrammer til dokumentation af systemets hardware. Block definition diagrammer (bdd'er) er brugt til at identificere og beskrive systemets hardware blokke og deres indbyrdes forhold. Internal block diagrammer (ibd'er) er brugt til at vise de identificerede blokkes interne og eksterne forbindelse, hvordan blokkene kommunikerer og hvilke signaler der flyder imellem dem.

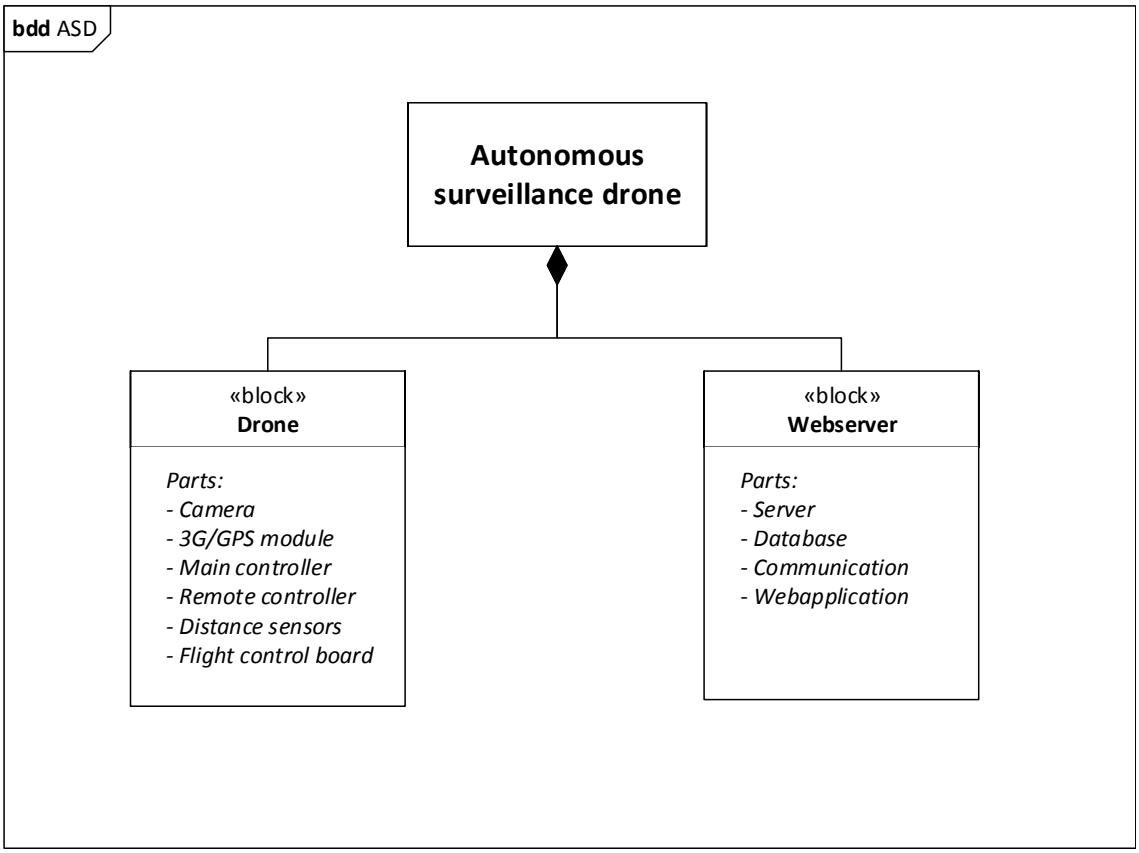
UML
Der er benyttet fire slags UML diagrammer til dokumentation af systemets software. På baggrund af use cases og domain modellen udformes pakkediagrammer, der bruges til at beskrive ansvarsområder i systemet. Sekvensdiagrammer bruges til at identificere systemets klasser, klassernes metoder samt timing i systemet. Klasser samt tilhørende metoder og attributter beskrives ved hjælp af klassediagrammer. State machines bruges til at beskrive flow mellem forskellige states.

8.2.1 Hardware

I dette afsnit beskrives hvordan systemets hardwarearkitektur er udformet vha. SysML diagrammer. Indledningsvis bruges block definition diagrammer til at identificere og beskrive systemets blokke. Senere åbnes udvalgte blokke og de interne og eksterne forbindelser vises med internal block diagrammer.

Block definition diagram

Det overordnede bdd på figur 8.5 viser hvilke hardware blokke systemet består af, samt hvilke parts blokkene indeholder.



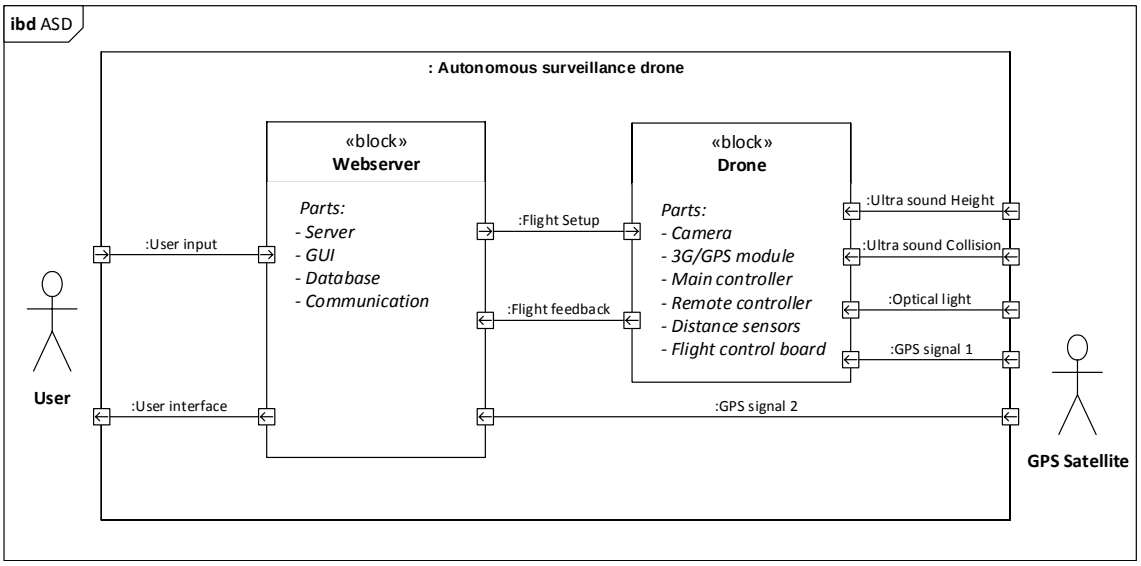
Figur 8.5: Overordnet bdd for systemet

Drone
Drone blokken indeholder alle de hardwareenheder der giver funktionalitet til drone. Beskrivelse af de eksterne og interne forbindelser mellem de forskellige parts er beskrevet nærmere med ibd'er.

Webserver
Webserver blokken indeholder server, database, communication og webapplikationen.

Internal block diagram

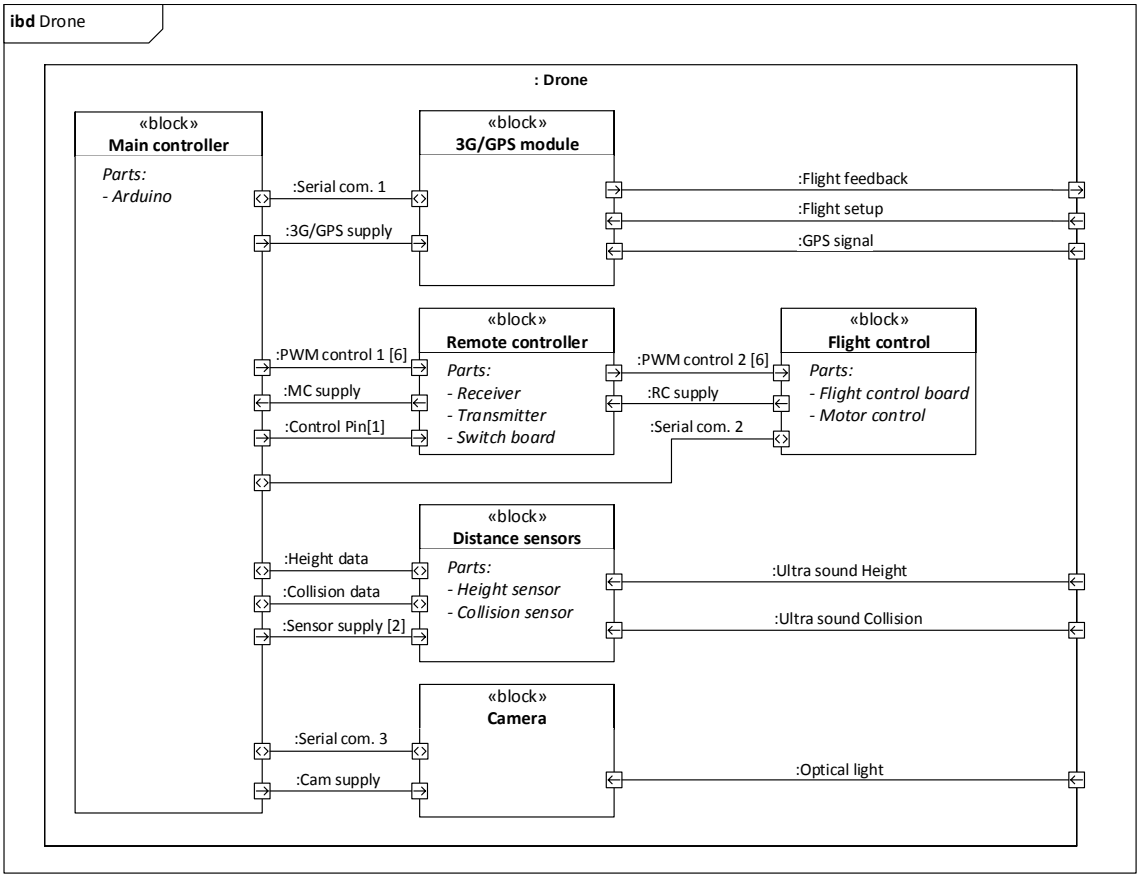
På figur 8.6 vises et overordnet ibd for systemet. Det overornede ibd viser hvordan systemets største blokke kommunikerer med hinanden og hvilken type signaler der anvendes mellem blokkene.



Figur 8.6: Overordnet ibd for systemet

Da drone blokken er stor og indeholder mange parts kræves yderlige beskrivelse. På den følgende side åbnes drone blokken og de interne forbindelser i blokken vises.

På figur 8.7 vises et ibd, der går mere i dybden med drone blokken. ibd'et åbner drone blokken og viser hvordan parts i drone blokken kommunikerer med hinanden.



Figur 8.7: ibd drone

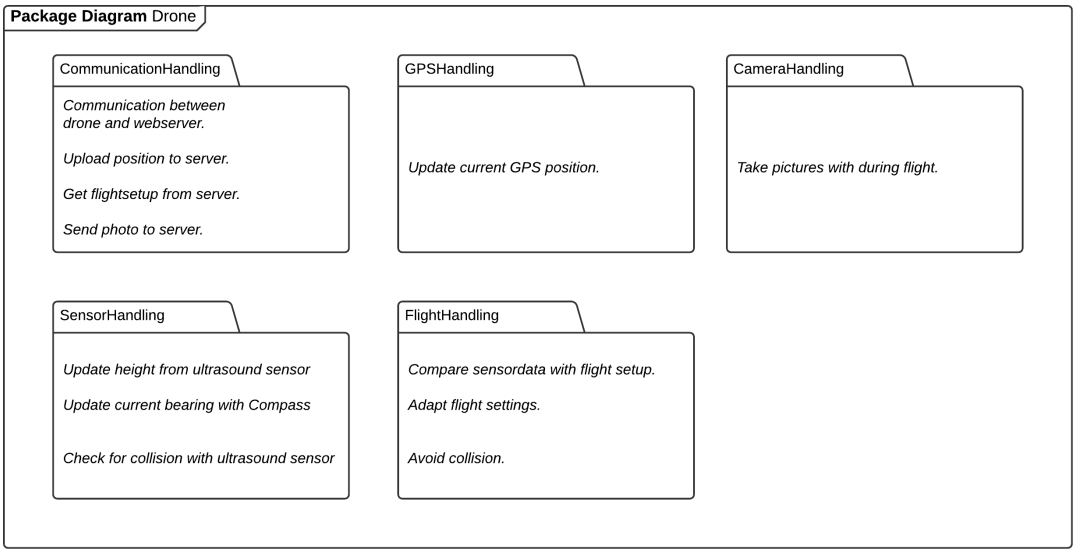
For yderlige beskrivelser af parts fra drone blokken henvises til ibd'er i hardwarearkitektur [3]. Der vises ikke udvidede ibd'er for webserver, da webserver ikke indeholder hardware og derfor beskrives nærmere ved brug af UML diagrammer i software afsnittet.

8.2.2 Software

I dette afsnit beskrives hvordan softwarearkitekturen ved brug af N+1 og applikations modellen tager udgangspunkt i use cases og ender ud i detaljerede klassedigrammer.

Pakkediagrammer

Indledningsvis identificeres pakkediagrammer ud fra use cases og domain modellen. På figur 8.8 vises et pakkediagram tilhørende dronens software. Pakkerne i pakkediagrammet bruges til at definere de forskellige ansvarsområder i systemts software. For nærmere beskrivelser af pakkediagrammer over systemets software henvises til logical view [4].

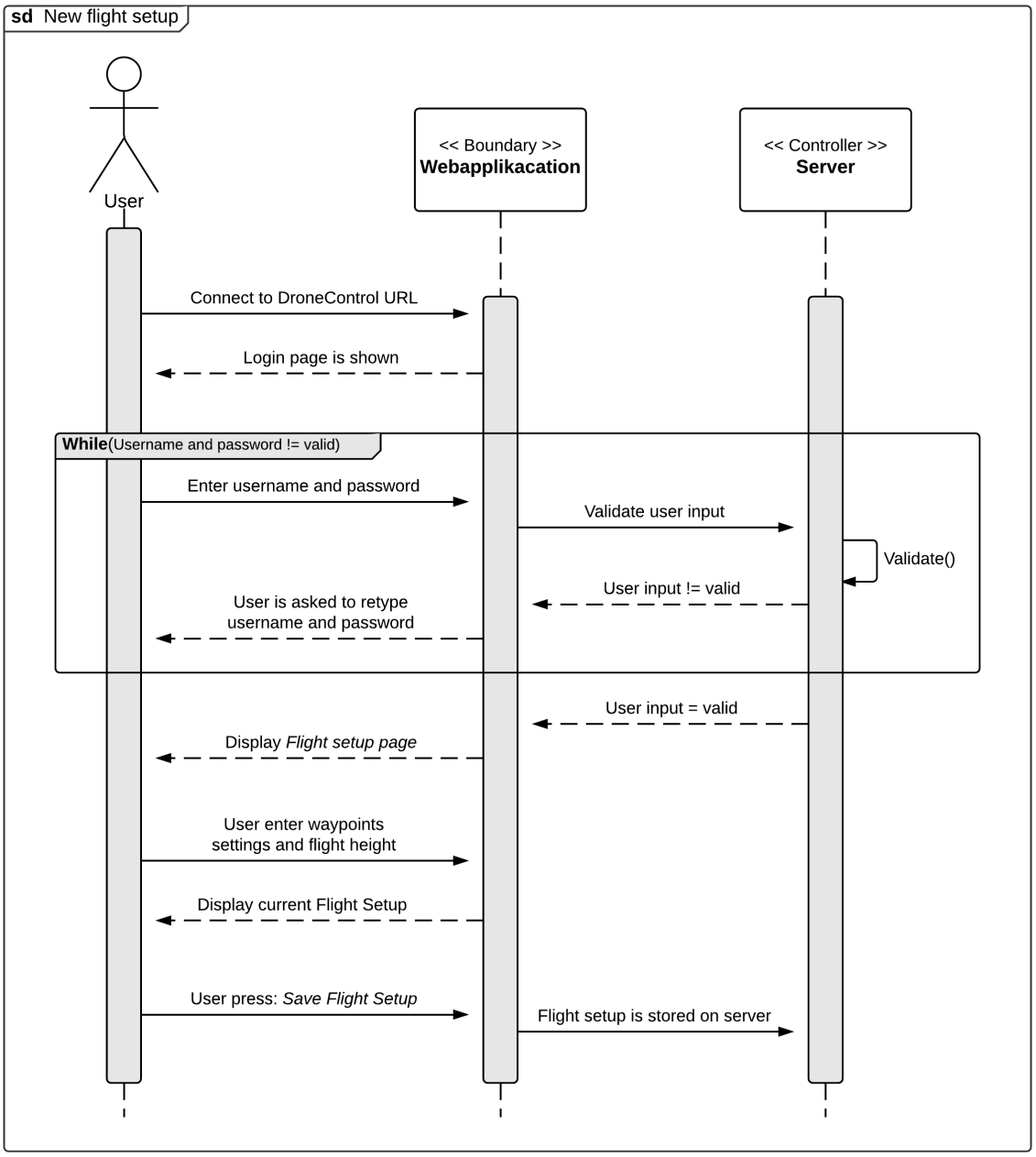


Figur 8.8: Overordnet pakkediagram for drone

Sekvensdiagrammer

Efter at have udformet pakkediagrammer påbegyndes udformning af sekvensdiagrammer. Disse diagrammer bruges til at identificere softwareklasser, klassernes metoder, klassernes indbyrdes forhold, samt timing i systemet. Enheder der indgår i sekvensdiagrammerne tages ud fra domain modellen.

Sekvensdiagrammet på figur 8.9 viser et eksempel på en use case omsat til et sekvensdiagram. Sekvensdiagrammet bruges til overordnet at vise hvordan bruger interagerer med webapplikation, når der laves ny flyveopsætning. For yderlige beskrivelser af sekvensdiagrammer tilhørende systemet henvises til logical view [4].

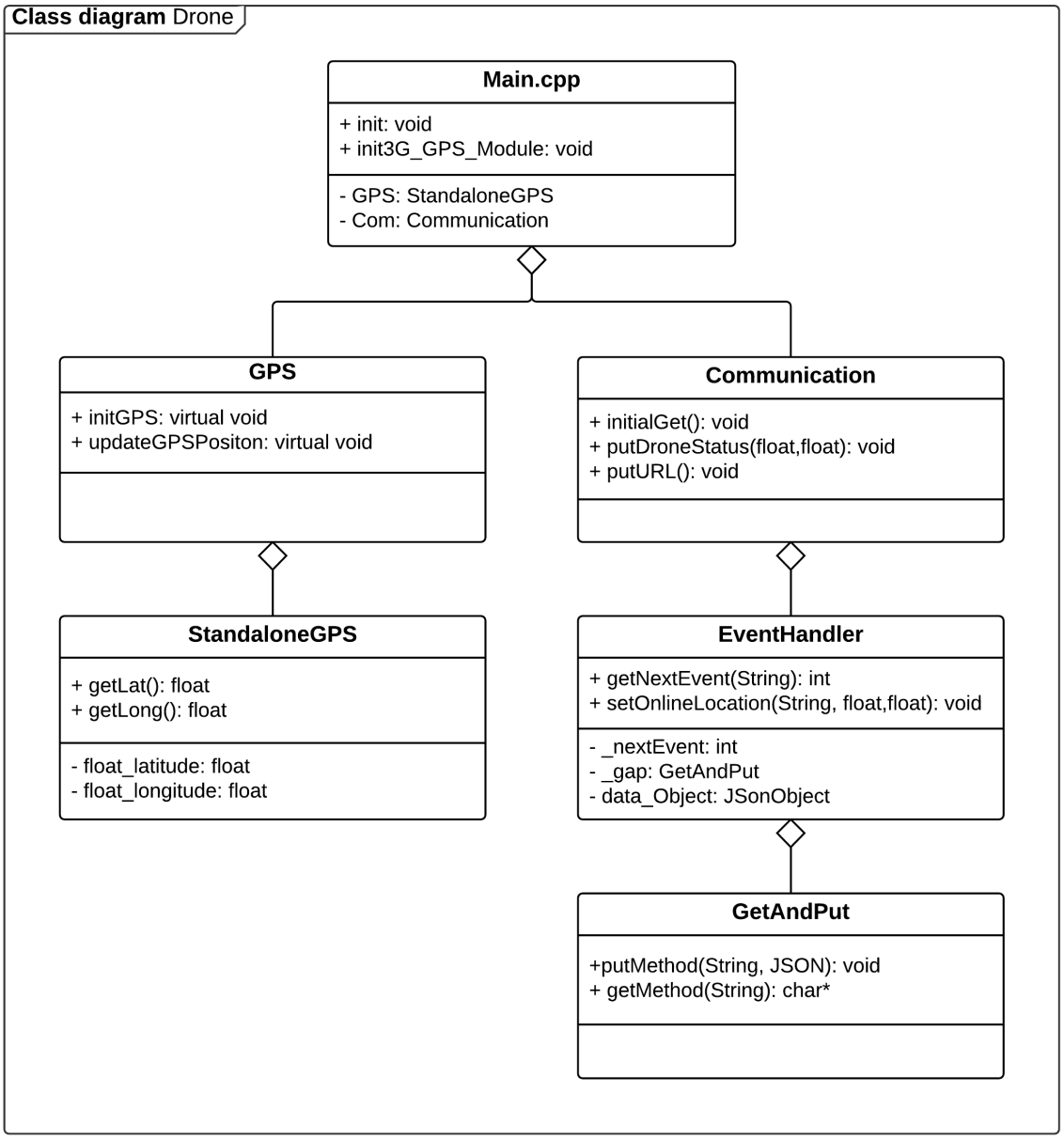


Figur 8.9: Sekvensdiagram - ny flyveopsætning

Klassediagrammer

Efter at have udformet sekvensdiagrammer er der opbygget et vist kendskab til de forskellige klasser, deres metoder og deres indbyrdes forhold. For på overskuelig vis at illustrere og beskrive systemets softwareklasser laves klassediagrammer.

På figur 8.11 vises et klassediagram tilhørende dronens software. Bemærk at klassediagrammet er fra første iteration, hvilket betyder, det er relativt lille og ikke indeholder så mange klasser og metoder. For mere information om de enkelte klasser og deres ansvarsområder henvises til logical view [4].



Figur 8.10: Klassediagram drone

8.3 Design

I dette afsnit beskrives systemets design. Systemet indeholder overordnet set de tre enheder drone, server og webapplikation samt en kommunikations protokol derimellem. Drone bruges til at overvåge og tage billeder, server står for håndtering af data, og webapplikation er systemets grænseflade til bruger. Mellem drone/server og webapplikation/server gøres brug af HTTP protokollen til kommunikation og udveksling af data.

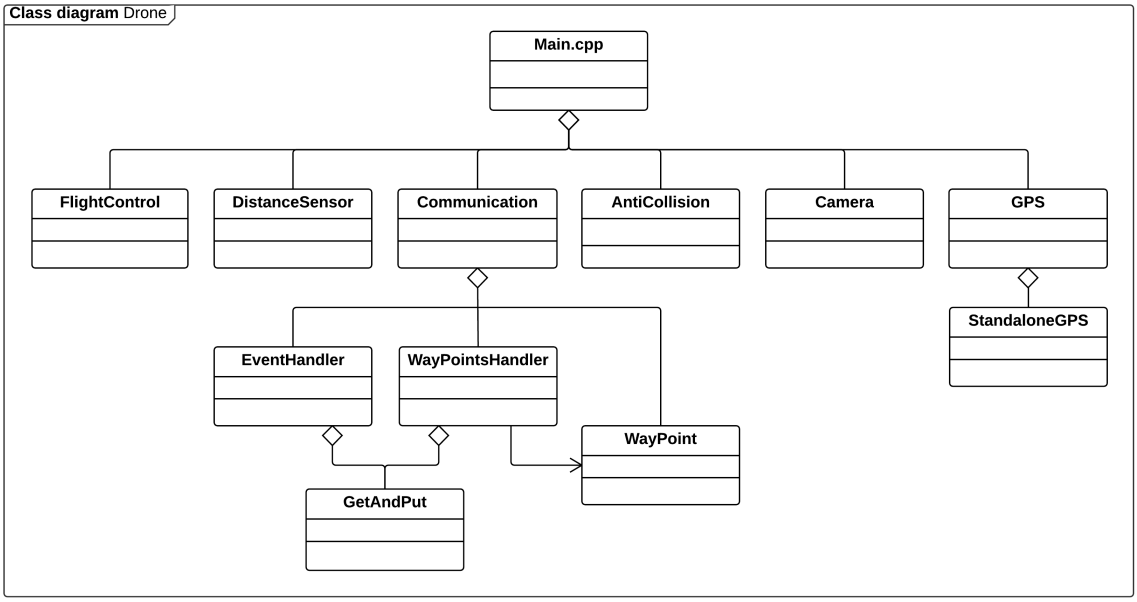
8.3.1 Drone

Systemets bruger er ansvarlig for at lave flyveopsætninger, som indeholder informationer om det område drone skal overvåge.

Når drone tændes skal den løbende og uden menneskelig indblanding kontrollere om der er en ny flyveopsætning tilgængelig på server. Hvis der er en ny flyveopsætning tilgængelig hentes den, og en ny flyvning påbegyndes.

Under flyvning flyver drone på egen hånd til de definerede GPS positioner. Afhængig af om bruger har valgt, der skal tages billeder eller ej, tager dronen billeder som via mobilt netværk sendes til server. Under flyvning kontrollerer drone løbende egen GPS position, flyvehøjde og flyveretning. Dette gør dronen for løbende at kunne tilpasse flyvehøjde og orientering og for at kunne sende information om nuværende GPS position til server.

På figur 8.11 vises et simplificeret klassediagram for drone. I det viste klassediagram vises softwareklasser og deres indbyrdes forhold. For yderlige information om de enkelte klasser, deres metoder og deres ansvarsområder henvises til logical view [4]



Figur 8.11: Klassediagram drone

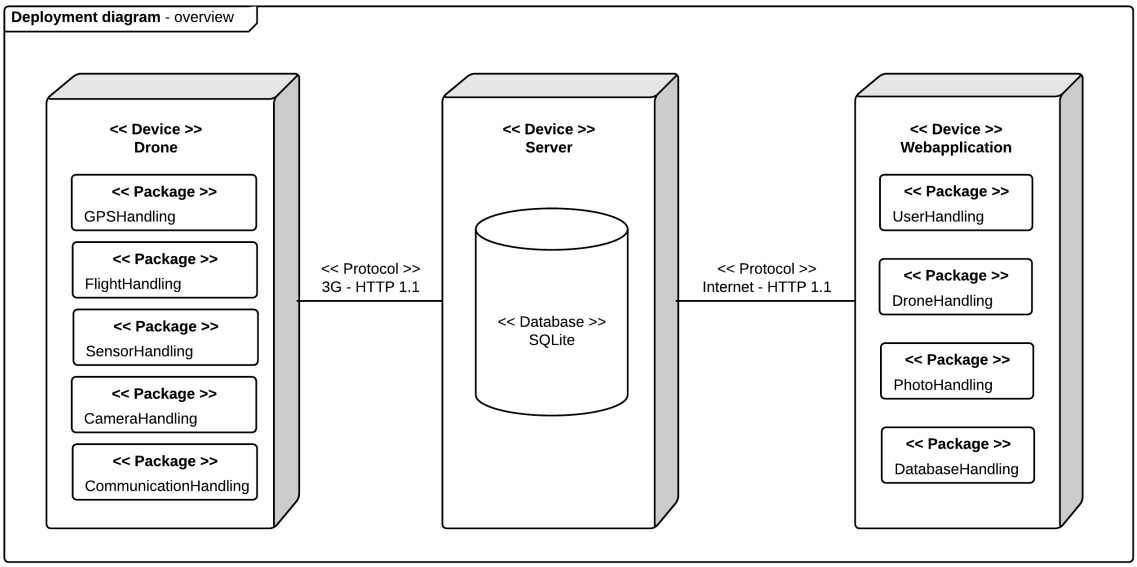
8.3.2 Server

Server består af SQLite database som tilgås via et REST API. I SQLite databasen gemmes og hentes løbende information om systemets brugere, samt information om flyveopsætninger, flyveruter og billeder.

Server er en passiv enhed, som er ansvarlig for håndtering af systemets data. Server tager aldrig initiativ til udveksling af data, den står i stedet og venter på at blive igangsat af enten drone eller webapplikation.

For systemets bruger kan det se ud som om udveksling af data og information går direkte fra webapplikation til drone eller omvendt. Men reelt set er webapplikation og drone aldrig i direkte kontakt med hinanden. I stedet går al kommunikation til og fra server via HTTP protokollen.

På figur 8.12 ses et deployment diagram, der viser hvordan drone og webapplikation kommunikerer med server for at hente og gemme information. For yderlige information om server og tilhørende SQLite database henvises til data view [5].



Figur 8.12: Deployment diagram

8.3.3 Webapplikation

Webapplikationen fungerer som grænseflade til systemets bruger. Da det ikke ønskes at alle og enhver kan tilgå webapplikationen, skal bruger logge ind før webapplikations funktionalitet kan benyttes.

Via webapplikationen kan bruger lave nye flyveopsætninger samt monitorere data og information fra tidligere flyvninger. Da det ikke er muligt at rette eller stoppe en aktuel flyvning, skal bruger være opmærksom på, hvordan en ny flyveopsætning indstilles.

Mellem server og webapplikation laves en socket connection. Dette betyder at indholdet af webapplikationen opdateres hver gang der tilføjes eller ændres i eksisterende data på serverens SQLite database. Yderlige information om webapplikations diagrammer findes i logical view [6] og implementation view [7].

8.3.4 HTTP protokol

Det ønskes at webapplikation og drone skal benytte samme kommunikationsprotokol, da der kun skal laves et interface til server, hvis drone og webapplikation benytter samme protokol.

Webapplikationen kan benytte stort set alle kommunikations protokoller, mens 3G/GPS modulet kun kan benytte et begrænset antal.

Det vælges at benytte HTTP protokollen, da 3G/GPS modulet understøtter denne protokol og fordi protokollen er blandt de mest anvendte protokoller til kommunikation mellem webapplikationer og servere.

Ønskes yderlige viden om HTTP protokollen henvises til W3 [11], mens der henvis til deployment view [8], hvis der ønskes mere viden om brug af protokollen.

8.4 Implementering

Implementering af systemets funktionalitet tager udgangspunkt i de udarbejdede hardware- og softwarediagrammer. Hardwarediagrammer bruges til at definere hvordan systemets hardware kobles, mens softwarediagrammerne bruges til at definere og implementere software.

Implementeringsfasen udarbejdes iterativt, og de højst prioriterede use cases implementeres først. Dette afsnit beskriver implementering af systemet.

8.4.1 Drone

Drone indeholder alt systemets hardware. Software til drone er udviklet i programmerings sproget C++, og er opdelt i forskellige ansvarsområder. Nogle softwareklasser bruges til at aflæse data fra højdesensor, 3G/GPS modul og kompas, mens andre klasser er ansvarlige for kommunikation. Information fra de forskellige softwareklasser samles og behandles på dronens main controller. Ud fra den indsamlede data tilpasses dronens flyveindstilling via dertil indrettede klasser.

Håndtering af GPS sker via den abstrakte *GPS* klasse og *StandAloneGps* klassen der nedarver fra den abstrakte klasse. Når nuværende og ønsket GPS position kendes, kan korrekt flyveretning og afstand til ønsket GPS position beregnes. Til at beregne korrekt flyveretning og afstand til ønsket GPS position bruges de to metoder *calBearingToTarget* og *calDistToTarget* fra *FlightControl* klassen.

Dronen kommunikerer med server via HTTP protokollen, til dette anvendes flere klasser. GetAndPut klassen som er den mest hardware nære klasse, håndterer *GET* og *PUT* requests. Metoder i *GetAndPut* klassen anvendes af de to klasser, *EventHandler* og *WayPointsHandler*. Disse klasser sorterer informationen fra *GetAndPut* og returnerer den nødvendige information. *Communication* klassen bruger Handler klasserne, hvilket giver systemet lav kobling. Ved at have denne opdeling, er klasserne uafhængige af hinanden og tests kan udføres enkelt.

Under flyvning aflæser main controller løbende flyvehøjde fra ultralydssensor via *Distance-Sensor* klassen og nuværende orientering fra flight control boardet via *FlightControl* klassen. *FlightControl* klassen benyttes også når der på baggrund af indsamlet data er brug for ændring af flyveindstilling, eksempelvis korrektion af *throttle*. I *FlightControl* klassen er der dels implementeret metoder, der initierer timere som kontrollerer PWM styrings signaler. Desuden er der implementeret set-metoder der anvendes til korrektion af *throttle*, *yaw*, *picth* og *roll*.

Af sikkerhedshensyn er det implementeret, at bruger med en fjernbetjening kan skifte fra autonom til manuel styring. Under flyvning kalder main controller løbende metoden *checkIfControllerIsOn*, for at kontrollere hvorvidt dronen skal styres autonomt eller manuelt.

8.4.2 Server

Indledningsvis i implementeringen blev der lagt meget fokus på server, da den spiller en afgørende rolle i kommunikationen mellen drone og webapplikation. Server er udviklet i programmerings sproget Python og med webframeworket Django[12]. Tilsammen udgør Python og Django en SQLite database med et RESTful API. Der benyttes en række API-endpoints for at tilgå eller ændre data på server.

Databasen er designet modulært så systemet har stor mulighed for udvidelser, fx. mulighed for håndtering af flere brugere og flere droner. Omdrejningspunktet i databasen er et event, der indeholder information omkring hvilke bruger som har oprettet event og til hvilke droner.

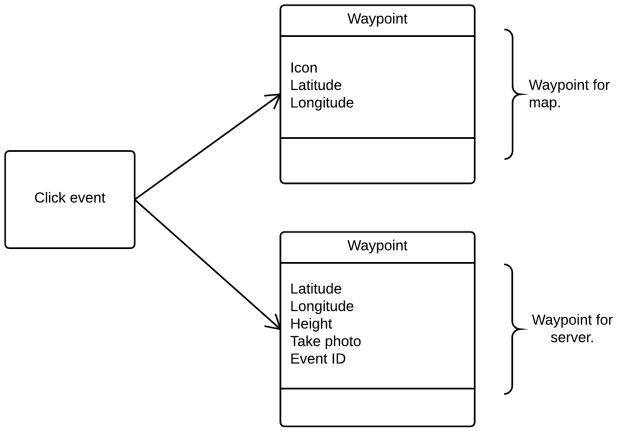
Til udviklingen af serveren er der også benyttet Django REST frameworket[13], som er en udvidelse til Django. Ved brug af dette framework kan tags som "@api_view(['GET', 'POST'])"benyttes til at fortælle server hvilke HTTP requests der er tilladt til et givet endpoint. Ved brug af REST frameworket kan der gøres brug af serializers, som bruges til at informere serveren om hvordan data skal formateres på. Med et browsable API simplificeres udviklingen med frameworket.

8.4.3 Webapplikation

Webapplikationen er udviklet i programmeringssprogene HTML, CSS og JavaScript. Webapplikationens frontend er udviklet med frameworket AngularJS[14], som er framework udviklet af Google. Derudover benyttes et google maps API til webapplikationen. Brug af google maps API'et muliggør brug af kort på webapplikationen. Da AngularJS er benyttet til udvikling, er frontend opdelt efter MVC-modellen[15].

Google maps kortet er implementeret ved brug af et google-maps directive[16], som pakker google maps API'et ind og gør det mere effektivt i forbindelse med Angular udvikling. Dette medfører at dele af den funktionalitet, som google maps oprindelig tilbyder ikke kan bruges.

Da det var et krav at waypoints skulle oprettes ved klik på kortet, blev der udviklet et klik event. Når der klikkes på kortet, bliver der automatisk tegnet et waypoint på kortet. I den bagvedliggende kode bliver der oprettet to waypoint objekter. Det ene waypoint objekt er til kortet, og indeholder icon, latitude og longitude til den placering waypointet skal tegnes. Det andet waypoint sendes til serveren når bruger ønsker at publicere den oprettede flyveopsætning. Dette er nødsaget, da google-maps directivet ikke kan tegne waypoints hvis de indeholder data som directivet ikke kender. Figur 8.13 illustrere hvordan klik eventet opretter to waypoint objekter.



Figur 8.13: Click event eksempel

Under udviklingen af webapplikationen har test vægtet en stor del, derfor er systemet udviklet så det er testbart. Til udviklingen blev AngularJS's dependency injection teknikker brugt. Eksempelvis er kommunikationen med backend implementeret i en service, som alle andre services og controller benytter sig af via dependency injection. Lignende teknikker er benyttet for alle services, som indeholder størstedelen af systemets logik.

Alle requests til server sker asynkront. Når webapplikation laver et request til server ventes der ikke på svar. Webapplikation arbejder videre, og når data fra server er klar får webapplikation et interrupt og præsenterer derefter data.

8.5 Test

I projektforløbet er der løbende blevet udført test. Indledningsvis udføres enhedstest i takt med nye moduler/enheder udvikles, senere laves integrationstest der tester kommunikation og samarbejde mellem flere enheder og til sidst udføres accepttest.

Nedenfor ses en kort beskrivelse af de udførte tests:

Enhedstest
Enhedstest udføres løbende i takt med nye enheder udvikles. Testene udarbejdes for at sikre kvalitet, funktionalitet og grænseflader i de nyudviklede enheder.

Hovedformålet med enhedstest er at teste tidligt i projektforløbet for at opdage eventuelle fejl og mangler, hvilket i sidste ende kan spare gruppwn doe meget tid og besvær, da fejl og mangler ofte er svære og mere tidskrævende at rette sent i et projektforløb.

Integrationstest
I integrationstest testes koblingen mellem to eller flere enheder. Her testes om grænseflader mellem enhederne fungerer og om enhederne kan kommunikere og arbejde sammen.

Accepttest
Accepttesten er en todelt test der tester systemet som helhed. Først udføres accepttest af funktionelle krav og dernæst testes ikke-funktionelle krav. Accepttest af funktionelle krav foregår via en gennemgang af use cases, som bruges til at kontrollere systemets funktionalitet. Ikke-funktionelle krav bruges til at teste systemspecifikationer.

For yderligere og mere konkret information om de forskellige tests og tilhørende resultater henvises til testdokumentet [9].

8.6 Resultater

I dette afsnit beskriver resultatet af accepttesten som blev foretaget i projektets afsluttende fase. Accepttesten er bygget op om seks test cases, som bruges til at teste det samlede systems funktionalitet. For mere information om de forskellige test cases henvises til test dokumentet. [9]

Test case 1 er succesfuldt gennemført. Når dronen tændes, initieres main controller samt 3G/GPS modul og dronen opdaterer sin nuværende GPS position. Efter at have opdateret egen GPS position opretter dronen forbindelse til mobilt 3G-netværk og sender information om sin nuværende GPS position til server.

Test case 2 er delvist gennemført. Bruger har mulighed for at tilgå webapplikation, logge ind på applikationen og oprette nye flyveopsætninger. Webapplikationen kan både hente og vise information fra server. Det er dog ikke muligt at sende information fra webapplikation til server, da kommunikation mellem webapplikation og server ikke er fuldt ud implementeret. Dette betyder at bruger ikke kan uploade nyoprettede flyveopsætninger til server.

Test case 3 er ligesom test case 2 delvist gennemført. Dronen kan sende sin nuværende GPS position til server og hente flyveopsætninger fra server. Det bemærkes dog, at flyveopsætninger, der hentes fra server, ikke er oprettet via webapplikationen men derimod tilføjet server via backend. Tilpasning af flyveindstillinger er implementeret i tre trin. Først tilpasses flyvehøjde, dernæst tilpasses flyveretning og når både flyvehøjde og flyveretning er tilpasset flyves fremad. Tilpasning af flyveindstillinger er ikke blevet optimeret, hvilket får dronen til at afvige fra det ønskede resultat og gør dele af test case 3 fejler.

Test case 4, 5 og 6 er ikke godkendt, da funktionalitet kun er designet og ikke implementeret.

8.7 Diskussion af resultater

I dette afsnit beskrives og diskuteres de resultater gruppen opnåede og der laves en beskrivelse af de punkter der kunne være lavet på bedre, smartere eller anderledes vis. Diskussionen tager udgangspunkt i de opnåede resultater beskrevet i resultatafsnittet.

Accepttesten forløb stort set som planlagt indtil test case 4-6, som omhandlede opsamling af billeder, visning af billeder og tilføjelse af antikollision. Test case 4-6 blev ikke godkendt, da de kun blev designet og ikke implementeret. Da arbejdsprocessen var planlagt i iterationer, var de fejlede test cases ikke kritiske for det resterende system.

Drone
Drone kan under flyvning finde nuværende flyvehøjde, flyveretning og GPS position ved aflæsning af data fra ultralydssensor, kompas og GPS. Ud fra det aflæste data tilpasses flyveindstillinger. Via det mobile 3G-netværk sender drone information om nuværende position til server og henter flyveopsætninger fra server.

Tilpasning af flyveindstillinger er implementeret i tre trin. Først tilpasses flyvehøjde, dernæst tilpasses flyveretning og når både flyvehøjde og flyveretning er tilpasset flyves der fremad. Hver del er implementeret og fungerer hver for sig selv, men samlet set fungerer det ikke optimalt pga. manglende optimering. Til justering af flyvehøjde og flyveorientering kunne der med fordel være gjort brug af PID regulering. I implementerings fasen opstod problemer med timing, specielt når 3G/GPS modulet blev brugt. Ved at indføre brug af flertrådet main controller kunne disse problemer undgås.

Webapplikation
Før systemets bruger kan få lov at tilgå webapplikationens funktionalitet er login påkrævet. Efter succesfuld login hentes og fremvises data fra server. Når bruger er logget ind kan der laves nye flyveopsætninger, men pga. manglede implementering kan webapplikationen ikke sende de nyoprettede flyveopsætninger til server.

Overordnet designvalg til webapplikation har fungeret tilfredsstillende. Dette har betydet at der undervejs i projektet ikke er lavet væsentlige ændringer i design og brug af udviklingsværktøjer. Webapplikationen er udviklet ved brug AngularJS, CSS3 og HTML5. Server er udviklet med en kombination af Django og Django-REST-framework, på samme vis kunne webapplikationen være blevet udviklet. Men server og webapplikation er bevidst udviklet adskilt og med forskellige værktøjer. Dette er gjort for at bevise at der er lav kobling mellem server og webapplikation, og for at undgå problemer med syntaks.

Server
Det er lykkedes at implementere en fuld funktionel server, der fungerer som bindeled mellem drone og webapplikation. Serveren er implementeret som en passiv enhed, der er ansvarlig for håndtering af systemets data og tilgås via HTTP protokollen.

Billeder i serverens database er koblet op til events, de kunne med fordel i stedet være koblet til waypoints. Dette havde givet en bedre kobling med GPS koordinater, og havde overordnet set forbedret database designet. Udover dette har de overordnet server design valg fungeret tilfredsstillende. Dette har betydet at der er blevet holdt fast i de værktøjer og beslutninger, der blev truffet i starten af projektet. Eksempelvis indeholder server en SQLite database, som til projektet har fungeret fint. Der kunne dog i stedet være gjort brug af en MongoDB database, der er en databasetype der håndterer data som objekter.

8.7.1 Færdigimplementering af manglende funktionalitet

Som det fremgik af resultatafsnittet, er det ikke lykkedes at implementere systemets fulde funktionalitet. Derfor bør der inversteres tid i færdigudvikling af systemet før eventuel videreudvikling foretages. I dette afsnit beskrives hvordan det restende systemfunktionalitet kan implementeres.

Test case 4
Før test case 4 kan gennemføres succesfuldt, skal håndtering af kamera implementeres. Sekvens- og klassediagrammer er designet, så det eneste der mangler er oprettelse af en *camera* klasse. Klassen oprettes og tilføjes de metoder der er beskrevet i klassediagrammet. Desuden skal dronen være i stand til at sende billeder til server og tage imod accept af billeder fra systemets bruger. Dette gøres ved udvidelse af *communication* klassen.

Test case 5
For at test case 5 kan gennemføres skal logikken til archive view'et implementeres. Helt overordnet er målet, at visning af billeder og flyveruter skal implementeres, så billeder og flyveruter fra forskellige flyvninger præsenteres på korrekt vis. Dette gøres bla. ved brug af kalender funktionen i AngularJS.

Test case 6
Test case 6 indeholder antikollision. Antikollision skal sikre at dronen ikke flyver ind i eventuelle forhindringer uden flyvning. For at implementere antikollisions funktionaliteten, kan der gøres brug af *DistanceSensor* klassen. Det er muligt at lave objekter af *DistanceSensor* klassen som kan bruges til altikollision.

8.7.2 Videreudvikling

I dette afsnit beskrives nogle af de muligheder der er for videreudvikling af systemet.

Optimering af højdemåling
Den ultralydssensor der bruges til aflæsning af flyvehøjde kan maksimalt måle afstande på 4-5 meter og er knap så præcis, når dronen er i bevægelse. Til at optimere måling af flyvehøjden kan systemet udvides med endnu en teknologi der måler flyvehøjde. En løsning kan være at tilføje andre sensorer, der i fællesskab med ultralydssensoren bruges til måling af flyvehøjde. På flight control board'et befinder der sig et barometer, dette kan eksempelvis benyttes til at supplere ultralydssensoren. Ved kombination af to teknologier, vil ultralydssensoren kunne anvendes når dronen skal lette eller lande og barometeret kan anvendes under flyvning. Måling af flyvehøjde med både ultralydssensor og barometer vil være en stabil løsning, da begge teknologier har stærke og svage sider, og de svage fjernes ved kombination af teknologierne.

PID regulering
Flyvehøjde og orientering er to parametre der konstant skal reguleres for at sikre dronen flyver på bedst mulig vis. Til tilpasning af flyvehøjde og orientering anvender dronen simpel regulering, der er bygget op omkring det data der aflæses fra højdemåler og kompas. I den implementerede regulering tages der ikke forbehold for tidligere fejl, hvilket betyder reguleringen får flyvehøjde og orientering til at svinge meget. Et alternativ til den nuværende regulering kunne være implementering af PID regulering. Med PID regulering måles og sammenlignes nuværende fejl med den tidligere fejl og reguleringen indstilles på mere dynamisk og flydende vis.

Flertrådet main controller
Den nuværende main controller kan ikke håndtere flere tråde samtidig, hvilket betyder main controlleren kun kan afvikle en metode af gangen. Dette giver i nogle tilfælde problemer og betyder at kritiske metoder komme til at vente på mindre kritiske metoder færdigafvikles. Aflæsning af GPS position fra 3G/GPS modulet er et eksempel på en tidskrævende men ikke særlig kritisk metode. Men i det nuværende system har aflæsning af GPS position samme prioritet som tilpasning af flyvehøjde og orientering, hvilket kan skabe kritiske situationer. Ved at anvendelse af et flertrådet system, kan der indføres prioriteringsliste og afviklingen af forskellige metoder kan optimeres. Desuden vil implementering af en flertrådet main controller tillade, at aflæsning af GPS position vil kunne afvikles samtidig som tilpasning af flyvehøjde og orientering.

Sikkerhed af webapplikation og server
På nuværende tidspunkt er der ikke ingen sikkerhed i forbindelse med webapplikation og server. Derfor bør øget sikkerhed være et fokuspunkt for fremtidig videreudvikling af systemet. Eksempelvis kunne der benyttes tokens i kommunikationen mellem server og webapplikation og kryptering af brugernavne/passwords i databasen.

8.8 Konklusion

Dette bachelorprojekt er udarbejdet for Ingeniørhøjskolen Aarhus Universitet. Målet for bachelorprojektet var at udvikle en overvågningsdrone, der ud fra brugers anvisninger autonomt kunne overvåge et defineret område.

Projektets udviklingsforløb har været iterativ og agilt, og til projektstyring er der blevet gjort brug af RUP og ASE-modellen. RUP er brugt til at danne de overordnede rammer for projektet og til at sikre en klar sammenhæng mellem projektets faser og forskellige arbejdsopgaver. ASE modellen er brugt til at danne grundlag for løbende udarbejdelse af dokumentation. Til store dele af projektets systemarkitektur er N + 1 modellen anvendt.

I projektforløbet er det lykkedes at implementere en server, hovedparten af en webapplikation til opsætning af drone og en række enheder der tilsammen skulle udgøre dronen. Det er ikke lykkedes at færdigudvikle webapplikation og dronen. Webapplikationen kan kun hente data fra server og de enheder der tilsammen skulle udgøre drone virker hver for sig, men ikke som samlet enhed.

Det er lykkedes at implementere kommunikationslink mellem drone og server, hvilket betyder dronen kan sende information om sin nuværende GPS position til server og hente flyveopsætninger fra server. Derudover kan dronen finde sin nuværende flyvehøjde, flyveretning og GPS position ved aflæsning af ultralydssensor, kompas og GPS. Ud fra det aflæste data tilpasser drone automatisk flyveindstillinger, men grundet dårlige vejrhold i implementeringsperioden og få testflyvninger kom automatisk ændring af flyveindstillinger ikke til at fungere optimalt.

Webapplikationen er udviklet så bruger ikke kan få lov at tilgå webapplikationens funktionalitet uden at være logget ind. Efter succesfuld login hentes og fremvises data fra server. Ved oprettelse af en ny flyveopsætning, bruges et kort til at markere waypoints. Hver gang der trykkes på kortet oprettes et nyt waypoint, og bruger bliver bedt om at indtaste hvorvidt der skal tages et billede på den valgte position og i hvilken højde billedet skal tages. Når flyveopsætning er færdigudarbejdet trykker bruger *upload*, men pga. manglede implementering kan webapplikationen ikke sende den nyoprettede flyveopsætning til server.

Da al kommunikation i systemet går til og fra server via et REST API, blev der i projektets udviklingsfase lagt stor fokus på at udvikle en velfungerende server. Det er lykkedes at implementere en fuld funktionel server, der fungerer som bindeled mellem drone og webapplikation. Serveren er implementeret som en passiv enhed, der er ansvarlig for håndtering af systemets data og tilgås via HTTP protokollen. Den udviklede server indeholder af en SQLite database. I SQLite databasen gemmes og hentes løbende information om systemets brugere, samt information om flyveopsætninger, flyveruter og billeder.

På baggrund de opnåede resultater kan det konkluderes, at de grundlæggende principper bag systemet er eftervist og testet. Det er lykkedes at få mange enheder og delsystemer til at fungere efter hensigten, men det overordnede mål for projektet er ikke opfyldt på tilfredsstillende vis.

Referencelisten 9

Referencer til dokumentation

[1] Information om krav:
Kravspecifikation

[2] Information om udviklingsværktøjer:
Foranalyse

[3] Information om bdd'er og ibd'er:
Hardware i Systemarkitektur og Design

[4] Information om pakke-, sekvens-, klasse-, og state machine diagrammer:
Logical view i Systemarkitektur og Design

[5] Information om server og tilhørende SQLite database:
Data view i Systemarkitektur og Design

[6] Information om udforming af webapplikation:
Logical view i Systemarkitektur og Design

[7] Information om opsætning af webapplikation:
Implementation view i Systemarkitektur og Design

[8] Information om systemets hardware elementer og deres sammenspil:
Deployment view i Systemarkitektur og Design

[9] Information om test og resultater:
Test

Weblinks

[10] Beskrivelse af N+1 model:
http://en.wikipedia.org/wiki/4%2B1_architectural_view_model

[11] Beskrivelse af HTTP protokol:
http://www.w3.org/Protocols/rfc2616/rfc2616.html

[12] Beskrivelse af Django:
https://www.djangoproject.com/

[13] Beskrivelse af Django REST framework:
http://www.django-rest-framework.org/

[14] Beskrivelse af AngularJS:
https://angularjs.org/

[15] Beskrivelse af MVC model:
http://msdn.microsoft.com/en-us/library/ff649643.aspx

[16] Beskrivelse af Google Maps Directive:
https://docs.angularjs.org/guide/directive