

Test

Autonom overvågningsdrone



Rasmus Lydixen
11647

Anders Opstrup
11726

Kevin Grooters
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. Lydixsen
11762 – Anders H. Opstrup
11655 – Kevin Grooters

Vejleder:

Torben Gregersen

Samlet sidetal: 27

Projekt afsluttet: 17-12-2014

Indholdsfortegnelse

Kapitel 1	Intro	1
1.1	Revisionshistorik	1
1.2	Indledning	2
1.2.1	Enhedstest	2
1.2.2	Integrationstest	3
1.2.3	Accepttest	4
Kapitel 2	Enhedstest	5
2.1	3G modul	5
2.2	Afstandssensor	7
2.3	GPS	8
2.4	PWM opsætning	9
2.5	Webapplikation	11
Kapitel 3	Integrationstest	12
3.1	Kommunikation mellem drone og server	12
3.2	Tilpas flyvehøjde	14
3.3	Tilpas orientering	15
3.4	Webapplikation	16
Kapitel 4	Accepttest	
	Funktionelle krav	17
4.1	Test case 1: Start drone	18
4.2	Test case 2: Ny flyveopsætning	19
4.3	Test case 3: Flyv til position	20
4.4	Test case 4: Billede af position	21
4.5	Test case 5: Vis tidligere flyvning	22
4.6	Test case 6: Anti kollision	22
Kapitel 5	Accepttest	
	Ikke funktionelle krav	23
Kapitel 6	Fejl og mangler	26
6.1	Funktionelle krav	26
6.2	Ikke funktionelle krav	27

1.1 Revisionshistorik

Rev. Nr	Dato	Initialer	Ændring
1.0	04-09-14	KG, RL, AO	Test dokument oprettet.
1.1	11-09-14	KG, RL, AO	Tilføjet accepttest af funktionelle og ikke funktionelle krav.
1.2	08-12-14	KG, RL, AO	Tilføjet enhedstest og integrationstest.
1.3	10-12-14	KG, RL, AO	Udført accepttest.

Tabel 1.1: Revisionshistorik

1.2 Indledning

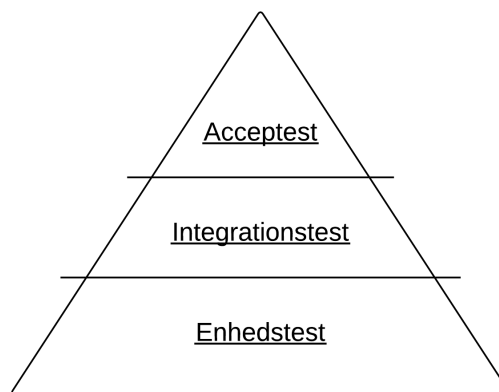
I projektforsløbet er der løbende blevet udført test. Indledningsvis udføres enhedstest i takt med nye enheder udvikles, senere laves integrationstest der tester kommunikation og samarbejde mellem flere enheder og til sidst udføres accepttest, der bruges til at teste det samlede system. Nedenfor beskrives de forskellige tests:

1.2.1 Enhedstest

Enhedstest er en testmetode der benyttes til at isolere og teste systemets enkelte enheder. Enhedstest udføres løbende i takt med nye enheder udvikles. Testene udarbejdes for at sikre kvalitet, funktionalitet og grænseflader af de nyudviklede enheder.

Hovedformålet med enhedstest er at teste tidligt i projektforsløbet for at opdage eventuelle fejl og mangler. Hvilket i sidste ende sparer gruppen for meget tid og besvær, da fejl og mangler er svære og mere tidskrævende at rette sent i et projektforsløb.

Figur 1.1 illustrerer hvordan mange fejl kan findes ved brug af enhedstest. Fejl der ellers "først" var blevet fundet når systemets enheder blev koblet sammen.



Figur 1.1: Mængde af fejl fundet ved test

Enhedstest bygges som regel op efter AAA-modellen¹.

Arrange: Opsætning af input til test og håndtering af afhængigheder.

Act: Hardware/software enhed under test stimuleres.

Assert: Der kigges på output og det afgøres om testen er succesfuld eller ej.

På figur 1.2 ses et eksempel på software test efter AAA modellen. I testen bliver der testet om den korrekte metode bliver kaldt når bruger prøver at logge in på webapplikationen.

```
33 it('authenticationUser method should be called on login', function () {  
34   Arrange spyOn(authServices, 'authenticationUser');  
35  
36   Act scope.onLogIn();  
37  
38   Assert expect(authServices.authenticationUser).toHaveBeenCalled();  
39 });  
40
```

Figur 1.2: AAA eksempel

¹<http://c2.com/cgi/wiki?ArrangeActAssert>

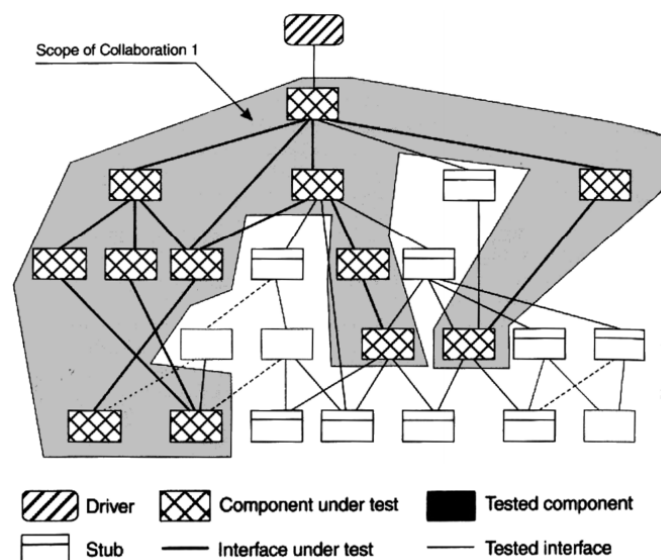
1.2.2 Integrationstest

Integrationstest bruges til at teste kobling mellem to eller flere enheder. Integrationstest bruges til at kontrollere om enhederne under test kan kommunikere og arbejde sammen.

Integrationstest af hardware og tilhørende software klasser er udført efter Collaboration modellen, mens software til server og webapplikation er integrationstestet efter metoderne buttom-up, top-down og collaboration.

Collaboration

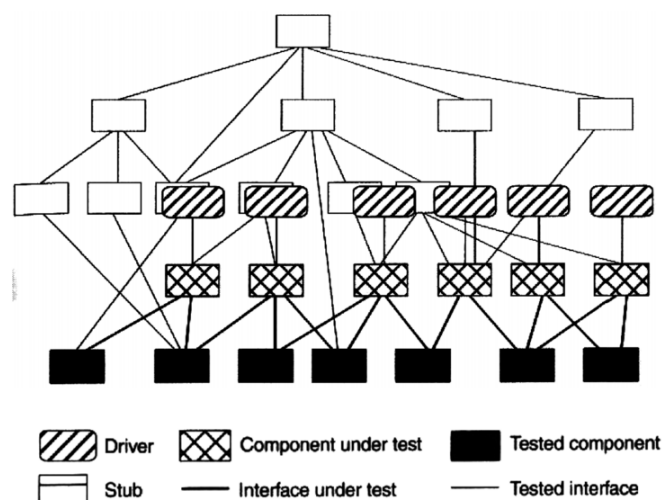
Når der testes efter collaboration samles en række enheder. Inden sammenkobling udgør hver enhed en lille klump funktionalitet, men tilsammen udgør enhederne en væsentlig del af systemts samlede funktionalitet. På figur 1.3 illustreres et eksempel på collaboration.



Figur 1.3: Collaboration

Button-Up

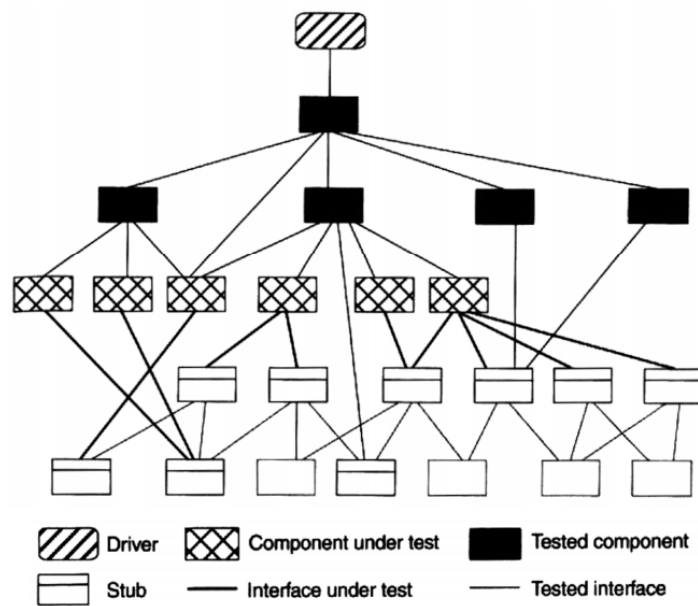
På figur 1.4 ses et eksempel på buttom up testing. Figuren viser hvordan systemets mest grundlæggende klasser er først testet og derefter bliver der testet op igennem systemet.



Figur 1.4: Buttom up

Top-Down

På figur 1.5 ses et eksempel på top down testing. Figuren viser hvordan systemets top klasser er testet først og derefter testet ned igennem klasserne.



Figur 1.5: Top down

1.2.3 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.

Enhedstest 2

I dette afsnit beskrives hvordan enhedstest af systemets enheder er udført. Enhedstesten bruges til at verificerer, at hver enkelt del virker uafhængigt af andre enheder. Der testes primært basal funktionalitet af enhederne, og i nogle tilfælde undersøges hvorvidt krav fra kravspecifikationen opfyldes.

2.1 3G modul

Enhedstest af 3G modulet foregår ved test af PUT og GET request, som er de metoder der er implementeret på dronen.

For udelukkende at have fokus på 3G modulet under enhedstesten, testes der ikke op imod den server som ellers anvendes til projektet. I stedet benyttes hjemmesiden requestb¹, en hjemmeside der tillader test af HTTP kommunikation. Ved at tilgå Requestb og starte en session tildeles en URL der kan bruges til test. Requestb indsamler information om alle HTTP request der er sendt til den tildelte URL.

På figur 2.1 vises et GET request der sendes fra 3G modulet til en requestb.in server. Den første linje efter *data:* fortæller om requestet er gennemført succesfuldt eller ej. Svaret *200 ok* betyder at GET requestet er gået igennem og det ønskede data på korrekt vis er hentet.

```
AT+CHTTPACT="requestb.in",80
GET /15zd1kn1 HTTP/1.1
Host: requestb.in
Content-Length: 0

+
Data received: 220
data:
http/1.1 200 ok
connection: keep-alive
server: gunicorn/18.0
date: mon, 08 dec 2014 08:53:01 gmt
content-type: text/html; charset=utf-8
content-length: 2
sponsored-by: https://www.runscope.com
via: 1.1 vegur
?
```

Figur 2.1: GET Request

¹<http://www.requestb.in>

PUT request bruges til at sende data til eller opdatere information på server. På figur 2.2 vises et PUT der sendes fra 3G modulet til `www.requestb.in`. De første seks linjer på figur 2.2 ses requestet header med linje syv er requestet body.

```
AT+CHTTPACT="requestb.in",80
PUT /1d8w2501 HTTP/1.1
Host: requestb.in
Cache-Control: no-cache
Content-Type: application/json
Content-Length: 12

Test message

+
sent
```

Figur 2.2: PUT metode

På figur 2.3 vises det information der tilgængelig på server efter PUT requestet er fuldført. *RAW BODY* viser det data der er modtaget og *HEADERS* indeholder parametrene for HTTP protokollen.

FORM/POST PARAMETERS	HEADERS
None	Host: requestb.in
	Content-Type: application/json
	Total-Route-Time: 0
	Connection: close
	Connect-Time: 0
	Cache-Control: no-cache
	X-Request-Id: e86f6f92-411b-4e35-93ab-1bc893ed41b6
	Via: 1.1 vegur
	Content-Length: 12
RAW BODY	
Test message	

Figur 2.3: Information på server

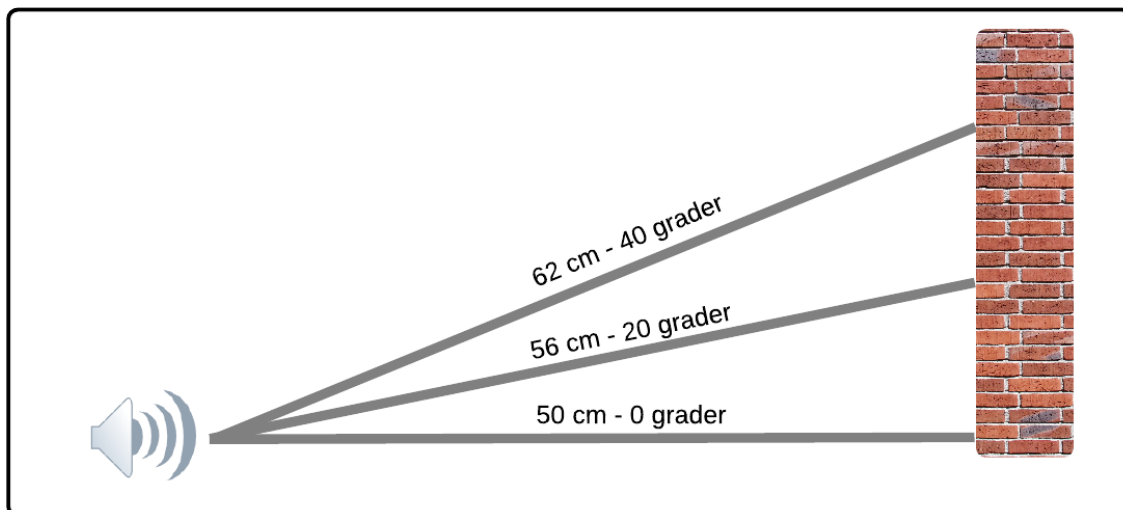
2.2 Afstandssensor

I denne sektion beskrives hvordan ultralydssensoren HC-SR04 er testet for at fastslå hvor pålidelig den er. Testen er udført ved at lave gentagne målinger af afstand mellem ultralyds sensor og en væg. For hver deltest er der foretaget 100 målinger og maksimal værdi, minimal værdi og gennemsnit fundet.

Ultralydssensoren bruges både til højdemåling og antikollision. Testen bruges til at kontrollere hvorvidt sensoren kan opfylde krav om nøjagtighed til højdemåling².

Undervejs i testen ændres vinklen mellem ultralydssensor og væg, hvilket indirekte betyder afstande også ændres til væggen. I udgangspunkt skyder sensoren lyd ud i vandret retning og efterhånden ændres vinklen. Det formodes at sensoren fungerer bedst når der er en 90 graders vinkel mellem sensor og væg.

På figur 2.4 vises en skitse af den anvendte testopstilling og tabellen nederst på siden viser resultater af den udførte test.



Figur 2.4: Skitse af testopstilling

Vinkel	Afstand	Maks måling	Min måling	Middel værdi
0	50 cm	49.0 cm	48.0 cm	48.02 cm
20	56 cm	49.0 cm	49.0 cm	49.0 cm
40	62 cm	50.0 cm	51.0 cm	50.0 cm

Tabel 2.1: Test ultralydssensor

²Se ikke funktionelle krav i *Kravspecifikation*

2.3 GPS

Der er to mål med enhedstest af GPS. Først og fremmest testes hvorvidt GPS'en virker, desuden testes hvor præcis GPS'en er. I kravspecifikationen er det defineret hvor stor en afvigelse GPS modulet må have, i testen undersøges det om afvigelse overholdes.

Hvis der hentes GPS koordinater lige efter at GPS'en tændes gøres der brug af *cold start*. Ved brug af cold start fås data fra GPS'en med det samme GPS'en har forbindelse til GPS satellitter, men dataen er upålidelig og upræcis. Derfor gøres der altid brug af *hot start*, både under test og når GPS'en bruges på dronen. Når der bruges hot start er GPS'en i udgangspunkt i forbindelse med GPS-satellitter og er derfor mere præcis.

Under test tændes modulet og får 30 sekunder til opstart. Efter de 30 sekunder er gået hentes data ud fra GPS modulet. Raw data fra GPS kommer i formatet Degrees/Minutes, og ved omformning fås decimal degrees.

For at beregne GPS'en præcision beregnes forskel på nuværende position og den position der angives af GPS'en. Resultat af enhedstesten er vist i tabellen nedenfor.

GPS moduls koordinater	Aktuelle position	Afvigelse
Latitude:10.191518	Latitude: 10.191560	4 meter
Longitude: 56.171863	Longitude: 56.171896	

Tabel 2.2: GPS modul

2.4 PWM opsætning

Testen bruges til at fastslå hvorvidt det PWM signal, der genereres af main controller, udsendes med korrekt frekvens og pulsbredde. Testen foregår ved at ændre værdien af compare-registeret³. Ændringer i compare registerets værdi bør medføre ændringer af pulsbredde i det PWM signal der udsendes fra main controller.

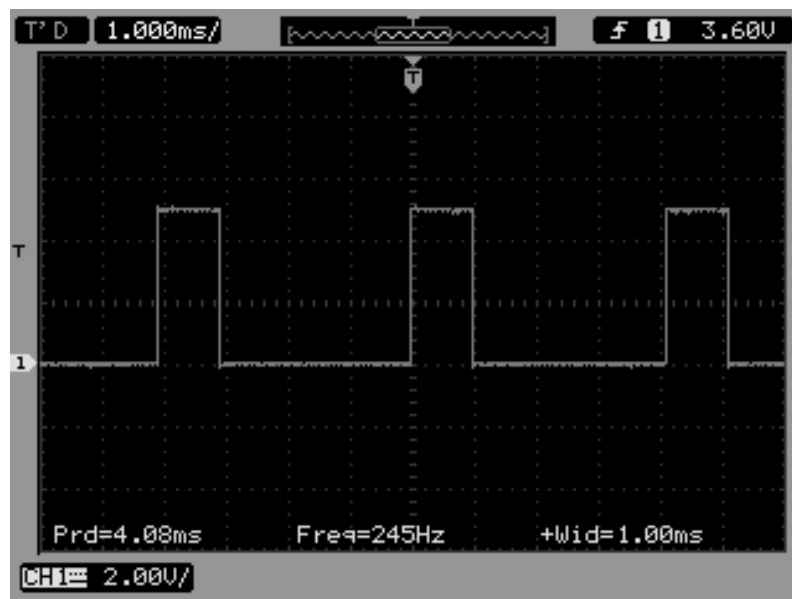
Indledningsvis indstilles compare-registeret med værdi på 16000, derefter ændres værdien til 24000 og til sidst sættes compare-registerets værdi til 32000. Ved brug af oscilloskop kontrolleres om PWM signal har den korrekte pulsbredde.

Nedenfor ses en tabel der viser hvordan testen forløb. Efter tabellen vises oscilloskop billeder taget under testen.

Værdi compare reg.	Frekvens	Forventet pulsbredde	Faktisk pulsbredde
16000	245 Hz	1.00 ms	1.00 ms
24000	245 Hz	1.50 ms	1.50 ms
32000	245 Hz	2.00 ms	2.00 ms

Tabel 2.3: Test resultat

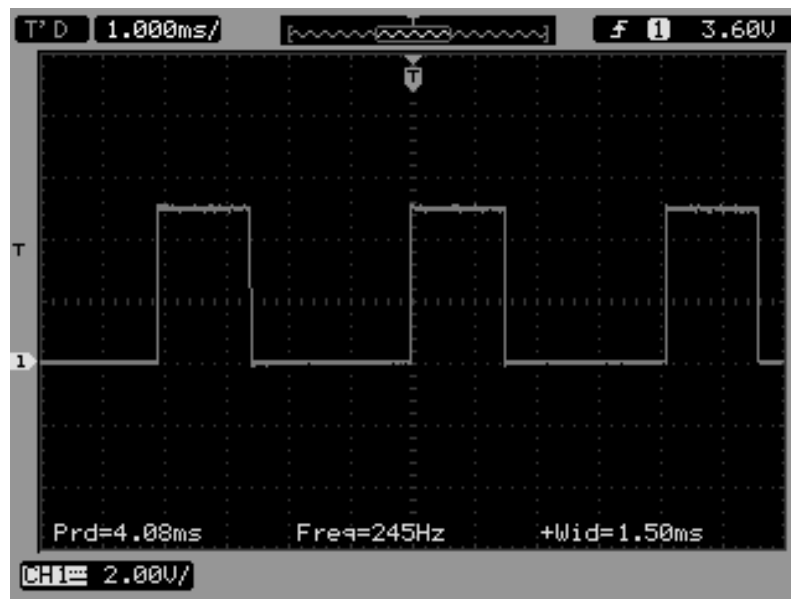
Figur 2.5 vises PWM signal hvor compare register har værdi på 16000.



Figur 2.5: PWM signal - Compare register 16000

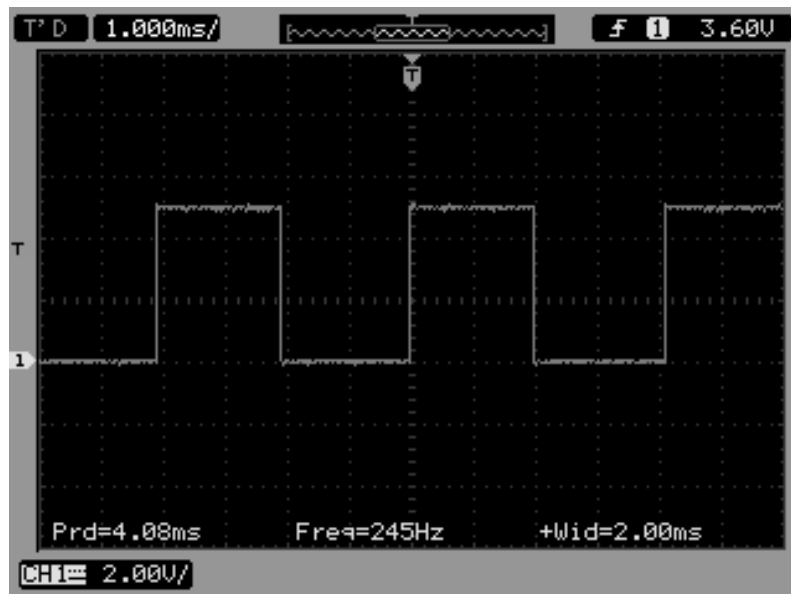
³Se implementation view i *Systemarkitektur og Design*

Figur 2.6 viser PWM signal hvor compare register har værdi på 24000.



Figur 2.6: PWM signal - Compare register 24000

Figur 2.7 viser PWM signal hvor compare register har værdi på 32000.



Figur 2.7: PWM signal - Compare register 32000

2.5 Webapplikation

Applikationen er enheds testet med test frameworket Jasmine⁴, som er et JavaScript framework. Frameworket benytter ikke DOM'en til test af applikationer. Jasmine skal bruge to filer for at kunne udføre tests.

karma.conf.js

Alt setup til Jasmine gøres i karma.conf.js filen som ligger i roden af projektet. Her bestemmes en række af opsætnings muligheder til frameworket. De mest væsentlige er følgende, hvilke filer Jasmine skal teste, hvilket output der ønskes til test, om testene skal køres en gang og derefter lukkes test programmet, eller om testene skal afvikles ved hvert gem af filer.

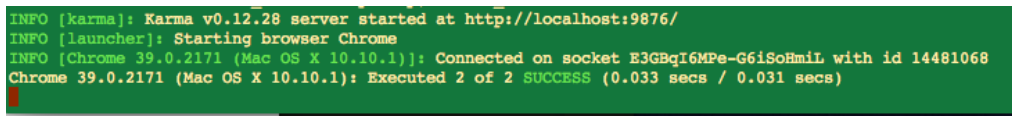
test fil

Jasmine skal også bruge test filer for at kunne afvikle test. Her i skrives test logikken for testene. Disse tests er opdel efter klasser.

Alle test bliver afviklet via terminalen ved at køre kommandoen "start" på filen karma.

```
1 $ ./node_modules/karma/bin/karma start
```

Efter start kommandoen er kørt starter frameworket en test server op som så afvikler koden og printer test resultater ud i terminalen. Som kan ses på figur 2.8. Efter serveren er startet køre den hele tiden i baggrunden og hver gang en fil i projektet bliver gemt køre den alle tests igennem igen, hvilket giver en god synergi mellem produktions kode og test kode.



```
INFO [karma]: Karma v0.12.28 server started at http://localhost:9876/
INFO [launcher]: Starting browser Chrome
INFO [Chrome 39.0.2171 (Mac OS X 10.10.1)]: Connected on socket E3GBqI6MPe-G6iSoHmIL with id 14481068
Chrome 39.0.2171 (Mac OS X 10.10.1): Executed 2 of 2 SUCCESS (0.033 secs / 0.031 secs)
```

Figur 2.8: Test eksempel

Yderligere information omkring test, se test filerne i mappen test i roden af projektet.

⁴<http://jasmine.github.io/2.0/introduction.html>

Integrationstest 3

3.1 Kommunikation mellem drone og server

Dette afsnit beskriver testen af kommunikationen mellem drone og server.

Kommunikationen er testet ved at bruge 3G modulet på drone som benytter HTTP protokollens GET og PUT metoder. Idet serveren er en passiv enhed, er det 3G modulet der skal tage initiativet til at hente eller sende data.

På figur 3.1 vises hvordan informationen på serveren ser ud. Ved brug af et GET request ønskes disse information hentet ned:

```
GET /api/drones/1/

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: PUT, OPTIONS, GET

{
  "id": 1,
  "is_online": true,
  "model": "aeroquad",
  "next_event": 1,
  "latitude": "56.172171",
  "longitude": "10.191837999999962"
}
```

Figur 3.1: Webserver UI

Ved brug af GET metoden på serverens URL (iha-11726.iha.dk/api/drones/1/), får 3G modulet informationen på figur 3.2 returneret. Dette information deles op i en header og body, hvor headeren kasseres og body sorteres ved hjælp af et JSon bibliotek.

```
AT+CHTTPACT="iha-11726.iha.dk",80
GET /api/drones/1/ HTTP/1.1
Host: iha-11726.iha.dk
Content-Length: 0

+
Data received: 377
data:
http/1.1 200 ok
server: nginx/1.4.6 (ubuntu)
date: mon, 08 dec 2014 12:39:35 gmt
content-type: application/json
transfer-encoding: chunked
connection: keep-alive
vary: accept, cookie
x-frame-options: sameorigin
allow: put, options, get

77
{"id": 1, "is_online": true, "model":
"aeroquad", "next_event": 1, "latitude":
"56.1721", "longitude": "10.1918379999"}
0
```

Figur 3.2: GET data fra server

Ved brug af PUT metoden, skal der sendes et request til serveren. Dette request skal indeholde serverens URL og de informationer der ønskes sendt. Et request der er brugt af 3G modulet ser ud som på figur 3.3.

```
AT+CHTTPACT="iha-11726.iha.dk",80
PUT /api/drones/1/ HTTP/1.1
Host: iha-11726.iha.dk
Cache-Control: no-cache
Content-Type: application/json
Content-Length: 90

{"id":1,"is_online":"true","model":"aeroquad",
"next_event":1,"latitude":50,"longitude":10}

+
sent
```

Figur 3.3: PUT request til server

Ved at sende den viste string, opdateres informationerne på serveren. På figur 3.4 vises det opdaterede information på serveren.

```
{
  "id": 1,
  "is_online": true,
  "model": "aeroquad",
  "next_event": 1,
  "latitude": "50",
  "longitude": "10"
}
```

Figur 3.4: PUT fra server

3.2 Tilpas flyvehøjde

Dette afsnit beskriver test af tilpasning af flyvehøjde.

Når flyvehøjden og de ønskede GPS positioner er kendt, begynder dronen at lette. Main controlleren anvender højdesensorer til at finde højden med. Så længe at flyvehøjden ikke er indenfor det ønskede interval, skal den hæve eller sænke sin højde, det gør den ved enten at forøge eller formindske throttle.

Først måles den aktuelle flyvehøjde, denne sammenlignes med minimums- og maksimalhøjden. Hvis flyvehøjden er udenfor intervallet reguleres der på throttle.

Figur 3.5 og 3.6 viser hvad systemet gør, hvis flyvehøjden ikke passer i intervallet. Change value værdien indikerer om throttle øges eller sænkes. Hvis flyvehøjden ikke er indenfor intervallet, vil change value værdien ændres. Når change value ændres, vil det påvirke duty cyclen af PWM signalet og derved ændre throttle værdien.

```
Current FlyHeight in cm: 6
FlyHeight < minFlyHeight
Change value: 2
Mode: Throttlecontrol
New value : 22240
```

Figur 3.5: Tilpasning af flyvehøjde

```
Current FlyHeight in cm: 76
FlyHeight > maxFlyHeight
Change value: -2
Mode: Throttlecontrol
New value : 22160
```

Figur 3.6: Tilpasning af flyvehøjde

Selvom flyvehøjden er indenfor intervallet, vil der stadig reguleres. Dette gøres for at holde dronen stabil og indenfor intervallet. Selve reguleringen foregår i mindre steps end hvis flyvehøjden var udenfor intervallet. Figur 3.7 viser hvad den aktuelle flyvehøjde er og hvor meget den er skiftet siden sidste måling.

```
Current FlyHeight in cm: 14
Forskel paa currFlyHeight og newFlyHeight = -2
```

Figur 3.7: Interval højde skift

3.3 Tilpas orientering

Dette afsnit beskriver test af dronens orientering. Dronen skal have den rette orientere, før den kan flyve fremad. For at finde orienteringen, bruges 3G/GPS modulet, kompas og main controlleren.

Til testen af orientering er nuværende og ønskede GPS positioner hardcoded. Den nuværende og ønskede GPS positions sammenlignes og orienteringen findes. Fra kompasset aflæser main controlleren dronens orientering og sammenligner med den ønskede orientering. Hvis orienteringerne ikke ligger indenfor det ønskede interval, skal dronen ændre retning. Dronens flyveretning ændres ved kortvarigt at forøge Yaw parameteren.

På figur 3.8 vises hvad main controlleren gør. Main controlleren sammenligner nuværende orientering med ønskede orientering, hvis orienteringen ikke er indenfor det ønskede interval, ændres denne ved ændre på indstillingen af Yaw.

```
Target Bearing 46.35
Current Bearing 104.57

Difference in bearing is more than 10 degrees.
Adjusting bearing

Change value: 12
Mode: Yawcontrol
New value : 24480

Change value: -12
Mode: Yawcontrol
New value : 24000
```

Figur 3.8: Ændring af orientering

Hvis orienteringen er indenfor intervallet, vil outputtet være som på figur 3.9.

```
Target Bearing 46.35
Current Bearing 52.57
Current bearing is in accepted range
```

Figur 3.9: Orientering er i intervallet

3.4 Webapplikation

Applikationen er integrationstestet med test frameworket Jasmine og Protractor¹. Protractor er et node.js test framework som google har udviklet specielt til test af AngularJS applikationer. Testene bliver afviklet på en test server som går ind på webapplikationen og simulere brugen af applikationen. Protractor skal bruge to filer for at kunne udføre tests på applikationen.

protractor.conf.js

Config filen skal bruges til at køre testene. Her bestemmes hvilke test server der skal benyttes til at udføre testene på, hvor test filerne ligger, browser type osv. Som standart bruger Protractor chrome som test browser. I config filen er rootElement også sat, hvilket bruges til at fortælle Protractor at testene ikke skal begynde for projektets rootElement er loaded.

test fil

Det sidste protractor skal bruge for at kunne køre her en test fil eller flere test filer som skal afvikles.

Test afvikling

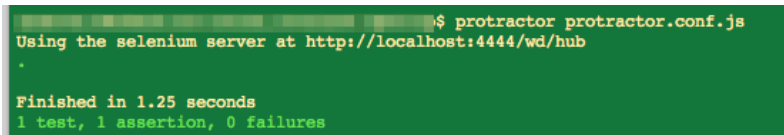
For at testene kan afvikles skal testserveren startes. Dette gøres via commandoen:

```
1 $ webdriver-manager start
```

Efter testserveren er oppe, kan testene afvikles ved kommandoen:

```
1 $ protractor protractor.conf.js
```

På figur 3.10 ses et udsnit af integrations test outputtet.



```
$ protractor protractor.conf.js
Using the selenium server at http://localhost:4444/wd/hub
.
Finished in 1.25 seconds
1 test, 1 assertion, 0 failures
```

Figur 3.10: Test eksempel

Yderligere information omkring test kan findes i test filerne i mappen test under integrations test som ligger i roden af projektet.

¹<http://angular.github.io/protractor/>

Accepttest

Funktionelle krav 4

Formål

Dette afsnit specificerer accepttest af systemets funktionelle krav.

Hvis der under accepttesten opstår fejl, der umuliggør fortsat test af efterfølgende testcases afbrydes denne test. Hvis der opstår fejl i enkelte testcases, men fortsat accepttest er mulig, underkendes den enkelte test og accepttesten fortsættes.

Såfremt en test afbrydes, underkendes eller ikke kan gennemføres, angives den eller de step der ikke kan gennemføres i kapitlet **Fejl og mangler**.

Testspecifikation

Software der skal testes:

Software	Version	Release dato	Bemærkning
Server	Rev. 226b85b	23/11-2014	
Webapplikation	Rev. 7969aac	04/12-2014	

Tabel 4.1: Software til test

Hardware der skal testes:

Hardware	Version	Release dato	Bemærkning
Kompas	Rev. 9d3ec7b	17/11-2014	
Højdemåler	Rev. 9d3ec7b	17/11/2014	
3G/GPS modul	Rev. 51ec5ac	7/11-2014	
Switch board	Rev. 9d3ec7b	17/11/2014	

Tabel 4.2: Hardware til test

Testprocedure

De individuelle use cases og scenarier i kravspecifikationen testes i enkelte test cases.

- Hvis et teststep gennemføres fejlfrit markeres dette med "Godkendt" i feltet "resultat" for det pågældende test step.
- Hvis et teststep gennemføres med ubetydelige fejl, markeres dette med "(OK)" i feltet "resultat" for det pågældende test step.
- Hvis et teststep ikke kan gennemføres eller gennemføres med betydelige fejl, markeres dette med "Ikke godkendt".

4.1 Test case 1: Start drone

Use case under test: UC 1: Start drone.

Forudsætninger: Ingen.

Hovedscenarie

Step	Handling	Forventet resultat	Resultat
1	Bruger tilslutter batteri og dronen tændes.	Systemet tilføres forsyning og ESC'er signalerer de er forbundet til forsyning.	Godkendt.
2	Main controller initialiseres.	Main controller startes.	Godkendt.
3	GPS initialiseres og dronens nuværende GPS position opdateres.	Nuværende GPS position opdateres og gemmes lokalt på main controller.	Godkendt.
4	3G initialiseres og dronen opretter forbindelse til 3G-netværket.	Dronen tilsluttes det mobile 3G-netværk.	Godkendt.
5	Dronens nuværende GPS position sendes til server.	På webapplikation vises drones nuværende position og at dronen er online.	Godkendt.

Extension 1: Forbindelse kan ikke oprettes.

Step	Handling	Forventet resultat	Resultat
a	Systemet indikerer fejl og bruger genstarter systemet.	Systemet genstartes og forbindelse oprettes.	Ikke Godkendt.

4.2 Test case 2: Ny flyveopsætning

Use case under test: UC 2: Ny flyveopsætning.

Forudsætninger: Bruger er oprettet i systemet og UC#1 er succesfuld gennemført.

Hovedscenarie

Step	Handling	Forventet resultat	Resultat
1	Bruger logger på webapplikation.	Login lykkes, og webapplikationens forside vises.	Godkendt.
2	Fra forsiden navigerer bruger til flyveopsætning.	Flyveopsætnings siden vises.	Godkendt.
3	Bruger laver en ny flyveopsætning og vælger: <ul style="list-style-type: none"> - GPS lokationer der skal flyves til. - Om der skal tages billeder ved de valgte lokationer. - Højde billeder skal tages fra. - Generel flyvehøjde. 	Flyveopsætning klargjort	(OK).
4	Bruger gemmer flyveopsætning og gør den tilgængelig på server.	Flyveopsætning gøres tilgængelig på server	Ikke godkendt.

Extension 1: Fejl i login.

Step	Handling	Forventet resultat	Resultat
a	Bruger føres tilbage til login.	Login kan påny forsøges.	Godkendt.

Extension 2: Der laves ikke ny flyveopsætning.

Step	Handling	Forventet resultat	Resultat
a	Gemt flyveopsætning bruges.	Flyveopsætning klargjort.	Ikke godkendt.

4.3 Test case 3: Flyv til position

Use case under test: UC 3: Flyv til position.

Forudsætninger: UC#1 og UC#2 er succesfuld gennemført.

Hovedscenarie

Step	Handling	Forventet resultat	Resultat
1	Drone henter flyveopsætning fra server.	Der oprettes forbindelse mellem server og drone og flyveopsætning sendes.	Godkendt.
2	Nuværende position opdateres.	Ingen handling	Godkendt.
3	Flyvehøjde tilpasses.	Flyvehøjde justeres	(OK).
4	Flyveorientering tilpasses.	Orientering justeres	Ikke godkendt.
5	Drone flyver mod ønsket position.	Drone nærmer sig ønsket position	Ikke godkendt.
6	Ønsket position er nået.	Ønsket position er nået.	Ikke godkendt.

Extension 1: Dronen henter ikke flyveopsætning.

Step	Handling	Forventet resultat	Resultat
a	Flyvning med en anden flyveopsætning er aktiv.	UC #3 fortsættes med den aktive flyveopsætning.	Godkendt.

Extension 2: Ugyldig GPS koordinat.

Step	Handling	Forventet resultat	Resultat
a	Drone går i fejlmode #1.	Forsøger at finde gyldig GPS koordinat, mislykkes dette lander dronen.	Ikke godkendt.

Extension 3: Ugyldig flyvehøjde.

Step	Handling	Forventet resultat	Resultat
a	Drone går i fejlmode #2.	Forsøger at finde gyldig flyvehøjde, mislykkes dette lander dronen.	Ikke godkendt.

4.4 Test case 4: Billede af position

Use case under test: UC 4: Billede af position.

Forudsætninger: UC#3 er succesfuld gennemført.

Hovedscenarie

Step	Handling	Forventet resultat	Resultat
1	Drone tager et billede af nuværende position.	Kamera aktiveres og der tages et billede.	Ikke godkendt.
2	Billedet sendes til server.	Billedet sendes til server gøres tilgængelig på webapplikationen	Ikke godkendt.
3	Bruger giver accept af billedet via webapplikation.	Billedet gemmes i database	Ikke godkendt.

Extension 1: Bruger beder om nyt billede.

Step	Handling	Forventet resultat	Resultat
a	Drone instrueres til at ændre flyvehøjde, orientering eller position.	Flyvehøjde, orientering eller position ændres.	Ikke godkendt.

Extension 2: Bruger svarer ikke inden for tidsgrænsen.

Step	Handling	Forventet resultat	Resultat
a	Drone får automatisk tildelt accept.	Drone påbegynder flyvning mod næste GPS lokation.	Ikke godkendt.

4.5 Test case 5: Vis tidligere flyvning

Use case under test: UC 5: Vis tidligere flyvning.

Forudsætninger: Bruger er oprettet i systemet.

Hovedscenarie

Step	Handling	Forventet resultat	Resultat
1	Bruger logger på webapplikation.	Login lykkes og webapplikations forside vises.	Godkendt.
2	Fra forsiden navigerer bruger til historik.	En oversigt over tidligere flyvninger vises.	Ikke godkendt.
3	Bruger vælger en specifik tidligere flyvning.	Billeder, video og flyverute tilhørende valgte flyvning vises.	Ikke godkendt.

Extension 1: Fejl i login.

Step	Handling	Forventet resultat	Resultat
a	Bruger føres tilbage til login.	Login kan påny forsøges.	Godkendt.

4.6 Test case 6: Anti kollision

Use case under test: UC 6: Anti kollision.

Forudsætninger: UC#3 er igangværende.

Hovedscenarie

Step	Handling	Forventet resultat	Resultat
1	Anti kollision system detekterer en forhindring.	Bremser dronens fremdrift.	Ikke godkendt.
2	Undvigelsesmanøvre udføres.	Drone passere forhindring og flyver videre.	Ikke godkendt.

Extension 1: Forhindringen kan ikke undviges.

Step	Handling	Forventet resultat	Resultat
a	Drone går i fejlmode #3.	Forsøger at finde måde at passere forhindring. Mislykkes det lander dronen.	Ikke godkendt.

Accepttest

Ikke funktionelle krav

5

Formål

Dette afsnit specificerer accepttests af systemets ikke funktionelle krav. Indledningsvis gennemgås krav til systemet som helhed og senere testes krav der har henblik på specifikke enheder i systemet.

Såfremt krav ikke kan opfyldes angives de i kapitlet **Fejl og mangler**.

Testprocedure

De individuelle use cases og scenarier i kravspecifikationen testes i enkelte test cases.

- Hvis et teststep gennemføres fejlfrit markeres dette med "Godkendt" i feltet "resultat" for det pågældende test step.
- Hvis et teststep gennemføres med ubetydelige fejl, markeres dette med "(OK)" i feltet "resultat" for det pågældende test step.
- Hvis et teststep ikke kan gennemføres eller gennemføres med betydelige fejl, markeres dette med "Ikke godkendt".

Generelle krav				
Krav nr.	Krav	Test	Forventet resultat	Resultat
1.1	Kommunikation mellem drone og webapplikation skal foregå trådløst.	Opsætning sendes til drone via webapplikation.	Den sendte opsætning modtages.	Godkendt.
1.2	Trådløs kommunikation benytter 3G protokol eller ældre.	Der undersøges hvilken slags mobilnet 3G-modul er forbundet til.	Det verificeres at protokollen passer.	Godkendt.
1.3	Højdemåler skal måle højde ± 10 cm.	Højdemåler placeres i en kendt afstand til en væg.	Afstanden måles med korrekt længde.	Godkendt.

Krav til server				
Krav nr.	Krav	Test	Forventet resultat	Resultat
2.1	Indholder database med billeder og flyveruter.	Databasen tilgås, billeder og flyveruter vises.	Databasen indeholder billeder og flyveruter.	Godkendt.
2.2	Indholder database med brugere.	Databasen tilgås og en liste med alle brugere vises.	De oprettede brugere vises.	Godkendt.
2.3	Indholder database med flyveopsætninger.	Databasen tilgås og en liste med flyveopsætninger vises.	De oprettede flyveopsætninger vises.	Godkendt.

Krav til webapplikation				
Krav nr.	Krav	Test	Forventet resultat	Resultat
3.1	Webapplikation skal kunne tilgås via både computere og telefoner.	Webapplikationen tilgås på både computer og telefon.	Applikationen tilpasser sig til den forbundne enhed.	Godkendt.
3.2	Load time skal være under 0.5 sekunder.	Webapplikationens URL tilgås.	Siden er loadet inden 0.5 sekunder er gået.	Godkendt.
3.3	Bruger laver flyveopsætning ved hjælp af kort.	Der startes en ny flyveopsætning og GPS koordinater vælges ud fra et kort.	Ved hjælp af kortet findes GPS koordinater.	Godkendt.

Krav til drone				
Krav nr.	Krav	Test	Forventet resultat	Resultat
4.1	Skal forsynes fra batteri.	Batteri tilsluttes drone.	Lyde fra ESC'er indikerer at drone er klar til flyvning.	Godkendt.
4.2	Batterilevetiden skal minimum være 15 minutter.	Der laves teoretiske beregninger.	Beregnet leveringstid er over 15 minutter.	Godkendt.
4.3	Flyvehastigheden skal minimum være $2\frac{m}{s}$.	Det noteres hvor lang tid dronen bruger på at flyve 10 meter.	Flyvehastigheden er minimum $2\frac{m}{s}$.	Ikke godkendt.
4.4	Flyvehøjde kan reguleres i følgende 3 intervaller: 1-1.5m, 1.5-2.0m og 2-2.5m.	I flyveopsætning sættes flyvehøjde mellem 1-1.5 meter.	Drone flyver i den ønskede højde.	Ikke godkendt.
4.5	Højde der tages billeder fra, kan reguleres mellem 1 og 2,5 meter.	Den ønskede højde sendes til dronen.	Drone tager billeder i den ønskede højde.	Ikke godkendt.

Krav til opsamling af data				
Krav nr.	Krav	Test	Forventet resultat	Resultat
5.1	Tiden mellem et billede tages og til det er tilgængeligt på webapplikation skal maksimalt være 5 sekunder.	Tiden det tager at sende et billede måles.	Billedet tager maksimalt 5 sekunder om at nå til server.	Ikke godkendt.
5.2	Gyldig højdemåling ligger i intervallet 0,5 til 4,5 meter.	Drone flyttes højde sættes først til mindre end 0,5 meter. Derefter placeres den i en højden 0.5-4.5 meter. Til sidst sættes højde over intervallet.	Drone går i fejlmode #2 udenfor interval på 0,5-4,5 meter.	Godkendt.
5.3	GPS skal angive koordinat indenfor $\pm 2,5$ meter.	Drone flyttes rundt og GPS lokationen findes og verificeres.	GPS koordinater er angivet indenfor intervallet.	Ikke godkendt.

Fejl og mangler 6

I dette afsnit vises en liste med fejl og mangler tilhørende accepttesten. Bemærk, afsnittet ikke bruges til at forklare hvordan eller hvorfor step eller krav fejlede. Ønskes kommentarer til og diskussion af opnåede resultater henvises til afsnittet *Diskussion af resultater* i projektetrapporten.

6.1 Funktionelle krav

Test case 2:

1. Step 4
2. Extension 2

Test case 3:

1. Step 4
2. Step 5
3. Step 6
4. Extension 2
5. Extension 3

Test case 4:

1. Step 1
2. Step 2
3. Step 3
4. Extension 1
5. Extension 2

Test case 5:

1. Step 1
2. Step 2
3. Step 3
4. Extension 1

Test case 6:

1. Step 1
2. Step 2
3. Extension 1

6.2 Ikke funktionelle krav

Krav til drone:

1. Krav 4.3
2. Krav 4.4
3. Krav 4.5

Krav til opsamling af data:

1. Krav 5.1
2. Krav 5.3