AARHUS SCHOOL OF ENGINEERING

E4PRJ4 - iFollow

Arkitektur

Navn:	Studienummer
Jesper Toft Jakobsen	201708777
Hans Krag Halberg	201707343
Mikkel Jensen	201708684
Mathias Juhl Johansen	201708948
Nicklas Grunert	201707773
Lucas Paulsen	201707663

Dato: 29-05-2019

Indholdsfortegnelse

Ordforklaring

Forkortelse	**Forklaring**
iFollow	Det samlede system, prototypen, robotten
WebApp	Webapplikation - Systemets GUI
GUI	Graphical User Interface - Grafisk brugergrænseflade
μ-controller	Micro controller - f.eks. PSoC, RPi
PSoC	Programmable System on Chip (PSoC 5LP)
RPi	Raspberry Pi Zero W
GPS	Global Positioning System - enhed for systemets lokationsdata
SysML	Systems Modeling Language - brugt til HW-arkitektur
UML	Unified Modeling Language - brugt til SW-arkitektur
BDD	Block Definitions Diagram - beskrivelse af system-blokke
IBD	Internal Block Diagram - beskrivelse af forbindelser
HTML	HyperText Markup Language - Sprog til opsætning af WebApp-indhold
CSS	Cascading Style Sheet - Sprog til udseende af WebApp-indhold
JavaScript	Dynamisk programmeringssprog til WebApp'ens funktionalitet
API	Application Programming Interface, grænseflade mellem forskellig software
jQuery	JavaScript-bibliotek til interaktion og animation af elementer
Node.js	Runtime system til udvikling af server-side webapplikationer.
Socket.io	JavaScript-bibliotek til kommunikation mellem server og client
Raspbian	Styresystem, som kører på Raspberry Pi
mySQL	Flertrådet SQL-databaseserver som understøtter flere brugere
MariaDB	Afgrening mySQL database håndteringssystem
MyPHPadmin	Redskab til håndtering af SQL database
C++	Objekt Orienteret Programmeringsprog
GNSS	Global Navigation Satellite System
DatabaseApp	Databaseapplikation
Database	mySQL Database

Indledning

I dette dokument beskrives systemets arkitektur. Arkitekturen tager udgangspunkt i kravspecifikationen. Derudover tages få designmæssige beslutninger i arkitekturen. Disse er blandt andet valg af μ-kontrollere og antal af motorer og sensorer. Det besluttes at benytte en Raspberry Pi Zero W, da systemet skal indeholde en web-applikation og RPi'en er en oplagt enhed at allokere en web-applikation på. Desuden har gruppen erfaring med denne specifikke μ-controller fra tidligere semestre. Dette gælder også PSoC 5LP udviklingsboardet, som skal have ansvaret for motorer, sensorer, regulering, samt læsning af knapper og styring af LED'er. Herudover holdes arkitekturen som udgangspunkt på et abstraktions niveau lige under kravspecifikationen. Til opsætning af hardware arkitekturen anvendes SysML-diagrammerne block definitions diagram (BDD) og internal block diagram (IBD). Til software arkitekturen anvendes UML til opsætning af applikationsmodeller. Her anvendes sekvensdiagrammer til at illustrere delsystemernes opførsel under de forskellige use cases, og klassediagrammer anvendes til at danne overblik over klasser, attributer, funktioner osv.

I dette dokument præsenteres først hardware arkitekturen, hvorefter der dykkes ned i software arkitekturen. Der vil blive præsenteret diagrammer med tilhørende beskrivelser i form af tekst og/eller tabeller.

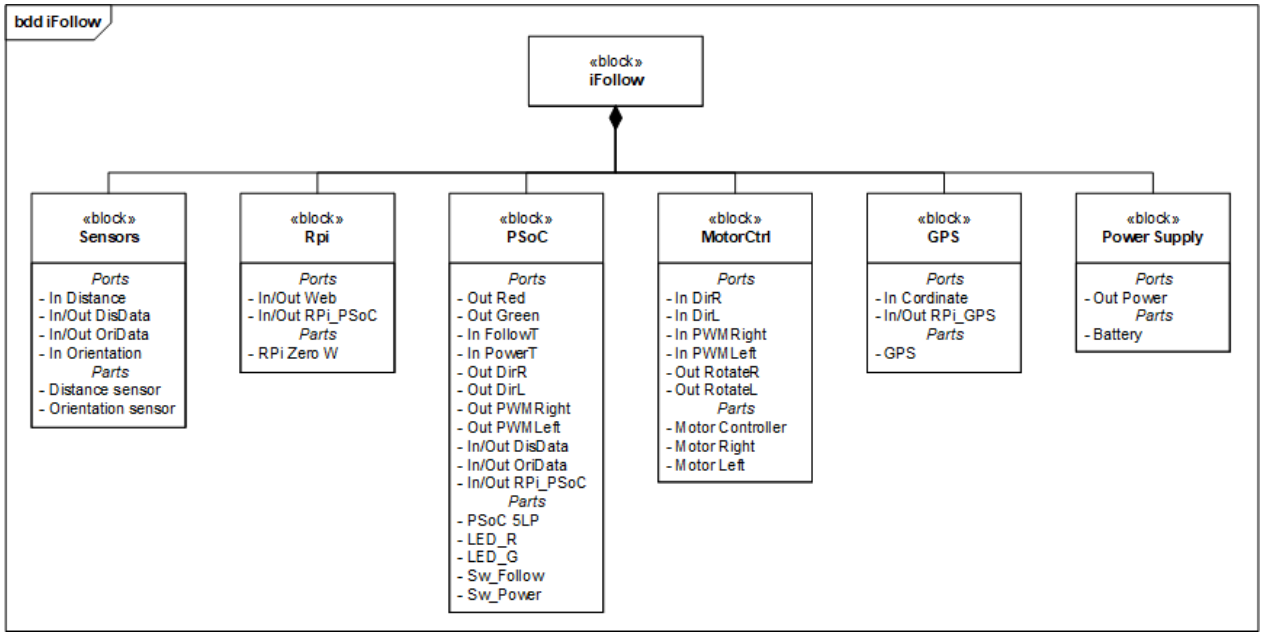
1 Hardware arkitektur

I dette afsnit vil hardware arkitekturen for iFollow blive beskrevet. Her vil de mest hardware-relevante dele af den overordnede systemarkitektur blive uddybet. Til dette opstilles BDD'er og IBD'er for blokkene, som blev identificeret i den overordnede BDD. Blokkene fra BDD'erne og signalerne fra IBD'erne beskrives med henholdsvis en blokbeskrivelse og en signalbeskrivelse.

Blokken Power Supply vil ikke blive beskrevet, da denne bare skal illustrer at der skal være et batteri til at drive systemet. Desuden vil forsyning og stel forbindelser ikke være med på de interne blok diagrammer, da det er gruppens holdning at disse forvirrer mere end de gavner på dette abstraktionsniveau. Alle signaler ligger i spændings intervallet 0-5V.

1.1 Overordnet system

Her vil det overordnet system blive beskrevet. Et BDD over systemet kan ses på Figur 1. Beskrivelse af de forskellige blokke kan ses i Tabel 1. På BDD'et ses de forskellige ports og parts der forventes at være nødvendige i designfasen.



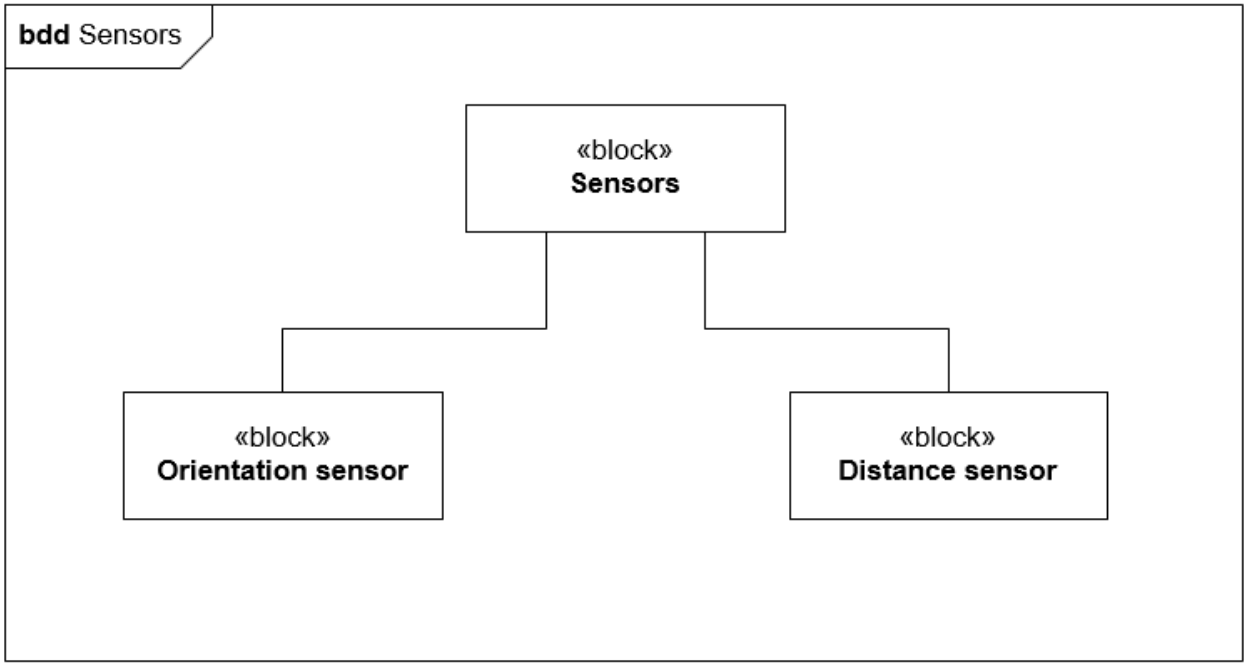
Figur 1: BDD for det overordnet systemet

Navn	Beskrivelse
Sensors	Sensorerne skal give iFollow-robotten information om dens orientering, og hvor langt den er fra en person. Orientations-sensoren skal kunne fortælle robotten, om den er væltet. Når robotten er i følge-tilstand, skal dens afstand til brugeren måles ved brug af en eller flere afstands-sensorer.
RPi	Der anvendes en Raspberry Pi til serveren, som håndterer webapplikationen og håndtering af data fra GPS.
PSoC	PSoC'en hånterer input fra brugeren. Eksempelvis sørger den for at følge-funktionen eksekveres ved læsning fra afstands-sensor og heraf reguleret motorstyring. PSoC'en bruger information fra orientations-sensoren til at beregne, om iFollow-robotten er væltet. Denne information sender den til RPi'en, så en bruger gennem webapplikationen kan se, om robotten er væltet
MotorCtrl	MotorCtrl er den hardware, der sørger for, at PWM- og retnings-signaler fra PSoC'en får motorerne til at rotere på korrekt vis, så iFollow-robotten bevæger sig som ønsket.
GPS	GPS'en er et modul, der med forbindelse til flere satteliter giver information til RPi'en om iFollow-robottens lokation
Power Supply	Som systemets forsyning anvendes et batteri.

Tabel 1: Blok beskrivelse af BDD på Figur 1

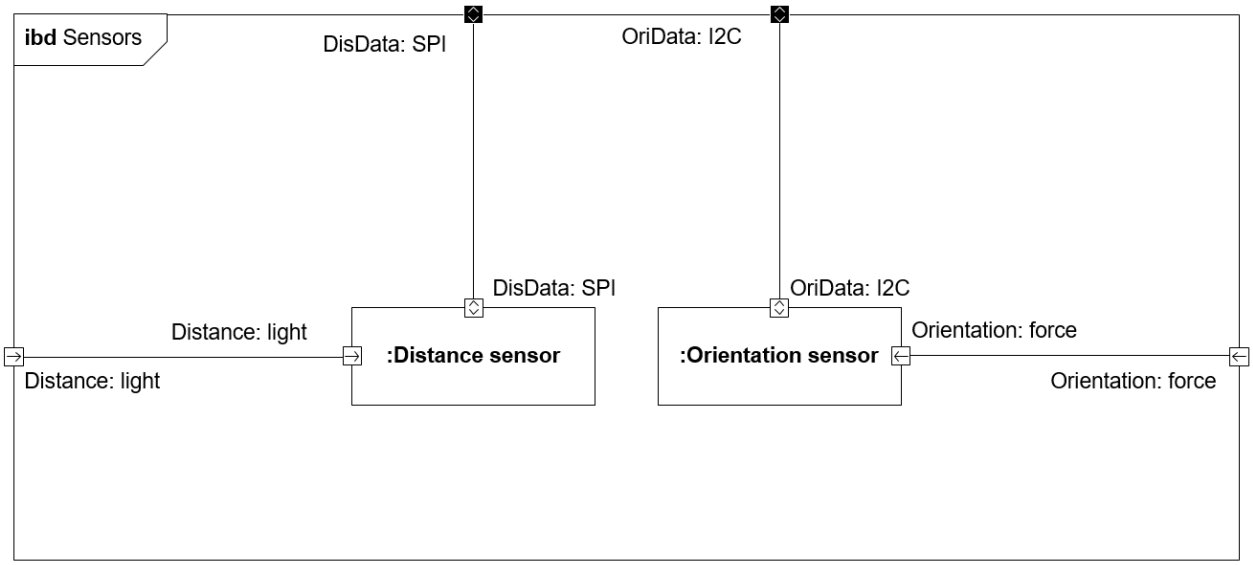
1.2 Sensorer

Sensorer er en betydelig del af systemet. Det skyldes at det er sensorer der skal gøre det muligt at følge efter brugeren, samt afgøre om robotten er væltet. Sensorene er i dette afsnit beskrevet med BDD, IBD, blok beskrivelser og signal beskrivelser. På Figur 2 ses et BDD over sensor blokken fra det overordnet BDD på Figur 1. Det ses, at den består af de to blokke Orientation sensor og Distance sensor. Orientation sensor har til formål at registrere om robotten vælter. Distance sensor skal gøre det muligt for robotten, at følge et objekt.



Figur 2: BDD for systemets sensorer

På Figur 3 ses et IBD over sensor blokken dennes signaler er beskrevet i Tabel 2



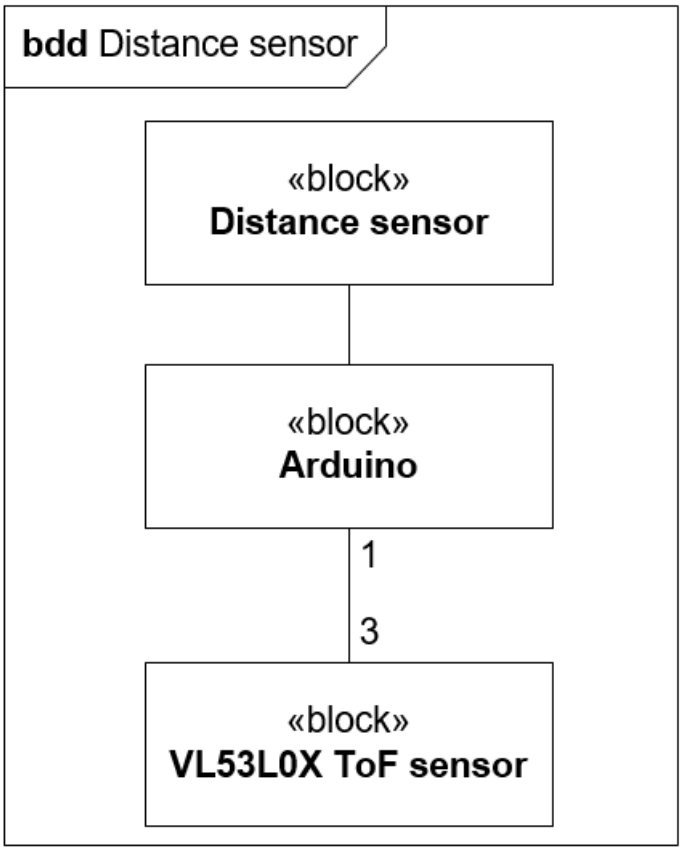
Figur 3: IBD for systemets sensorer

Signal tabel		
Navn	**Beskrivelse**	**Type**
Distance	Fysisk målt afstand af sensor/sensorer.	distance
DisData	Seriel kommunikation mellem afstands sensor og PSoC	SPI
OriData	Seriel kommunikation mellem orienterings sensor og PSoC	I2C
Orientation	Orientations sensorens input fra den fysiske verden	force

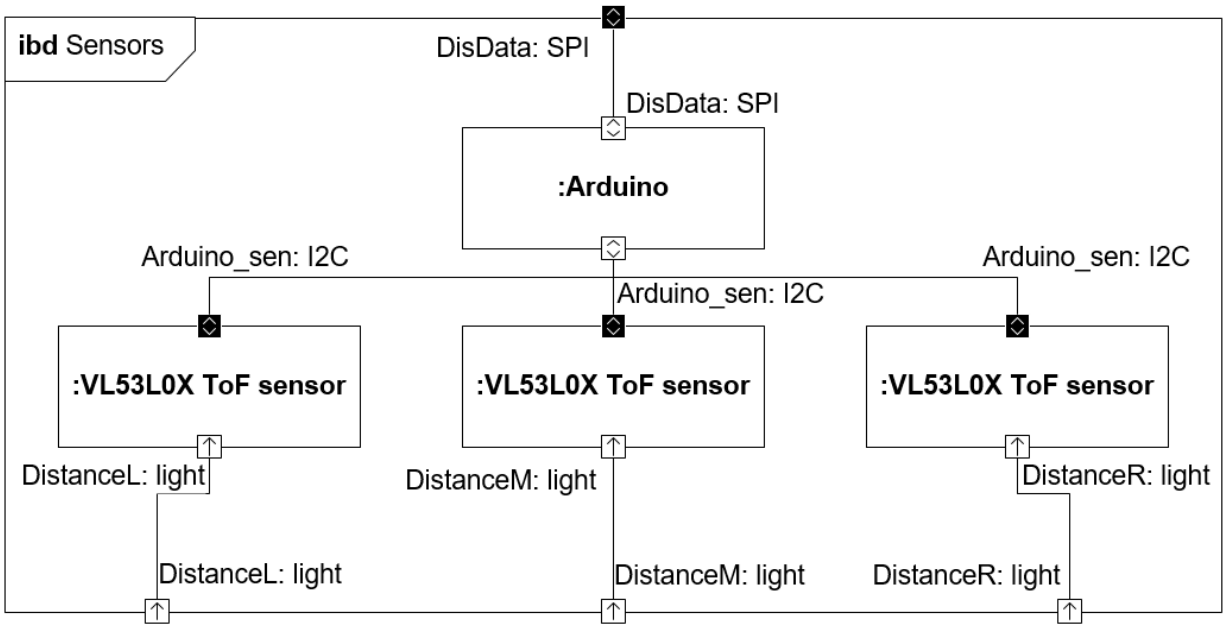
Tabel 2: Signal tabel til IBD på Figur 3

1.2.1 Distance sensor

Da blokken "Distance sensor" på Figur 2 består af flere delkomponenter, dykkes der i dette afsnit ned i både bdd og ibd for blokken. På Figur 4 ses et blokdiagram over distance sensor. Herpå ses en uspecificeret Arduino. Dette skyldes at det i analysen er blevet fastlagt, at der skal bruges en Arduino API. Det er dog ikke fatslagt hvor stor denne Arduino skal være. På Figur 5 ses det tilhørende ibd. Herpå ses to kommunikationsbusser, én I2C bus imellem Arduinoen og de tre sensorer, samt en SPI bus fra arduinoen til PSoC'en.



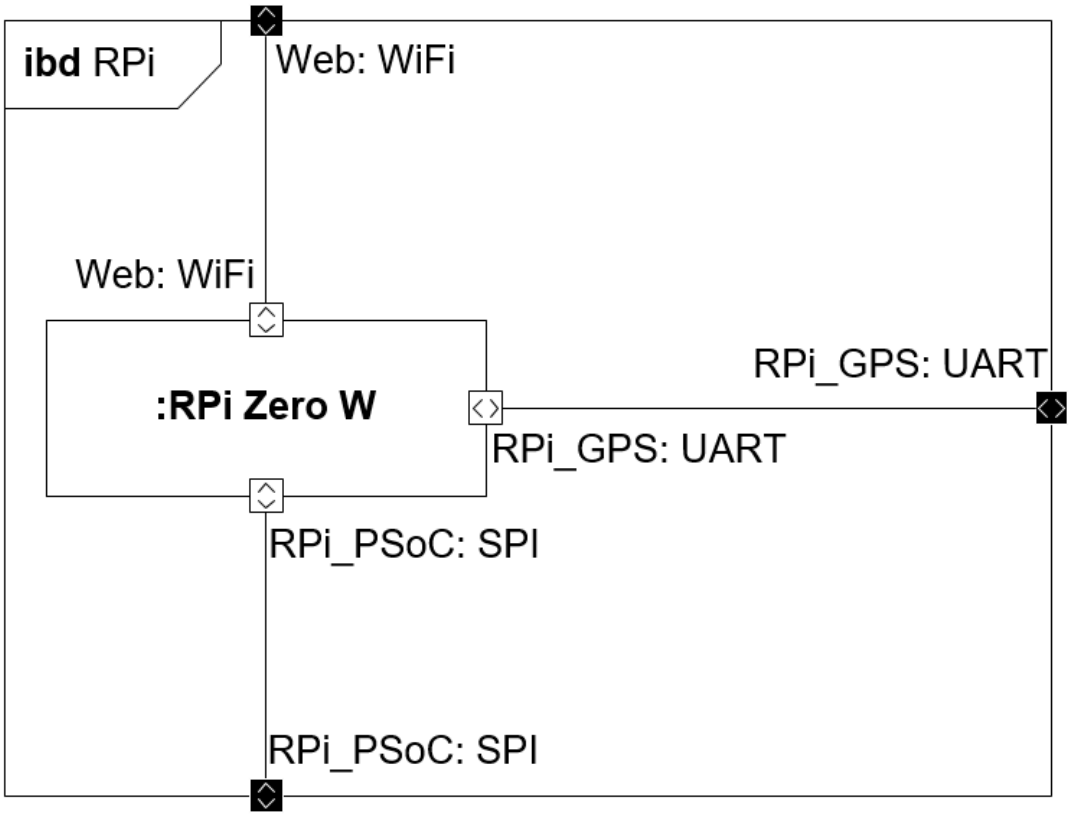
Figur 4: BDD for systemets Distance sensor



Figur 5: IBD for systemets Distance sensor

1.3 RPi

I dette afsnit beskrives forbindelser til og fra RPi'en, dette gøres vha. et ibd og en signaltabel.



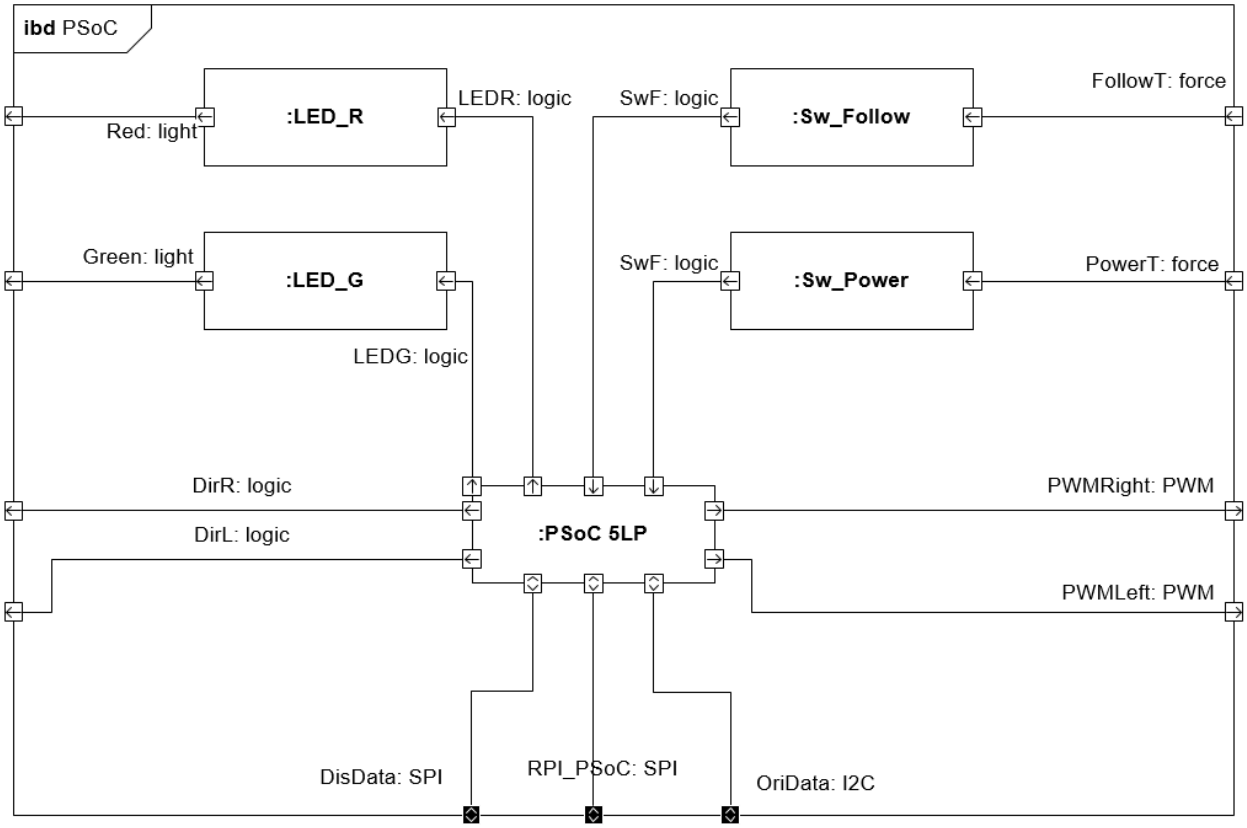
Figur 6: IBD for systemets RPi

Signal tabel		
Navn	**Beskrivelse**	**Type**
Web	Netværkskommunikation på Web-applikationen	WiFi
RPi_GPS	UART kommunikationsbus imellem GPS og RPi	UART
RPi_PSoC	SPI bus imellem PSoC og RPi	SPI

Tabel 3: Signal tabel til IBD på Figur 6

1.4 PSoC

Denne blok er den største, som det også fremgår af det overordnet blok diagram på Figur 1. Denne blok vil blive beskrevet med et ibd og en signal tabel.



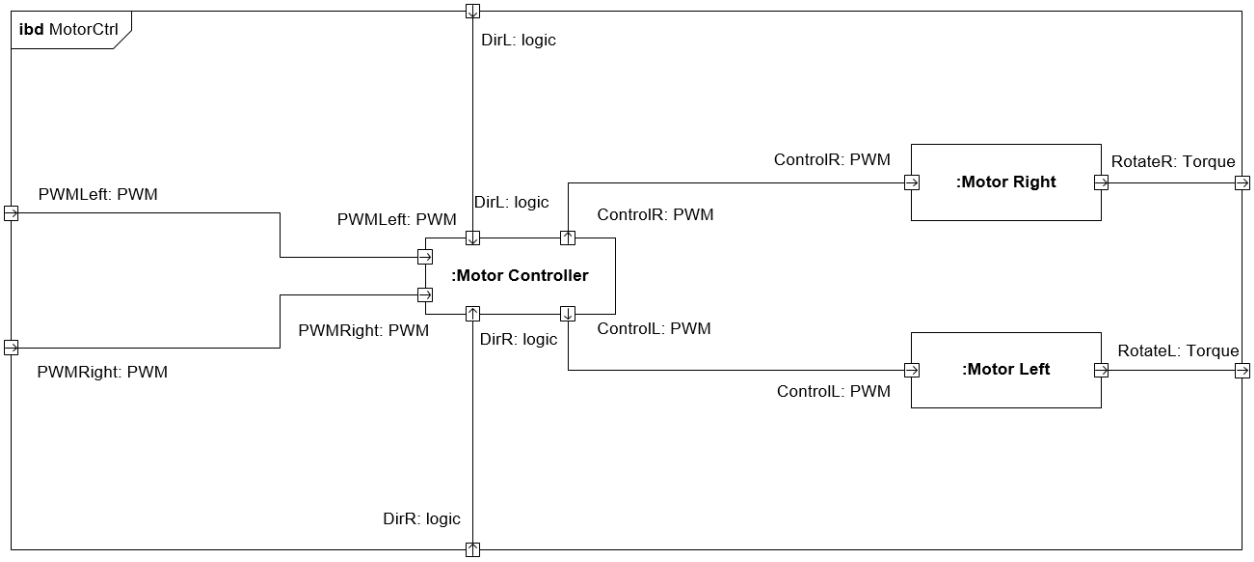
Figur 7: IBD for systemets PSoC blok

Signal tabel		
Navn	**Beskrivelse**	**Type**
Red	Lys fra den røde status LED	light
Green	Lys fra den grønne status LED	light
LEDR	Signal fra PSoC til den røde LED	logic/PWM
LEDG	Signal fra PSoC til den grønne LED	logic/PWM
FollowT	Fysisk tryk på follow toggle switch	force
PowerT	Fysisk tryk på power toggle switch	force
SwF	Signal fra follow switch til PSoC	logic
SwP	Signal fra power switch til PSoC	logic
DirR	Signal til bestemmelse af højre motors retning	logic
DirL	Signal til bestemmelse af venstre motors retning	logic
PWMRight	PWM signal til hastigheds bestemmelse af højre motor	PWM
PWMLeft	PWM signal til hastigheds bestemmelse af venstre motor	PWM
DisData	Kommunikationsbus imellem PSoC og Arduino	SPI
RPi_PSoC	Kommunikationsbus imellem PSoC og RPi	SPI
OriData	Kommunikationsbus imellem PSoC og Orientations sensor	I2C

Tabel 4: Signal tabel til IBD på Figur 7

1.5 MotorCtrl

MotorCtrl blokken indeholder de to motorer samt et motor kontroller print. Denne blok beskrives med et IBD og tilhørende signaltabel.



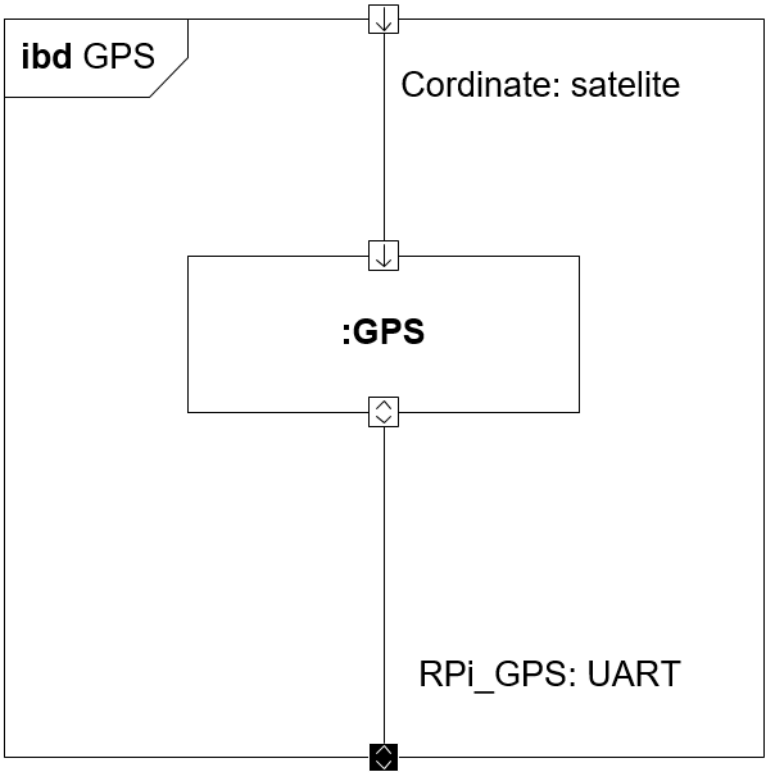
Figur 8: IBD for systemets MotorCtrl blok

Signal tabel		
Navn	**Beskrivelse**	**Type**
RotateR	Højre hjuls rotation	Torque
RotateL	Venstre hjuls rotation	Torque
CotrolR	Styre signal til højre motor	PWM
ControlL	Styre signal til venstre motor	PWM
DirR	Signal til bestemmelse af højre motors retning	logic
DirL	Signal til bestemmelse af venstre motors retning	logic
PWMRight	PWM signal til hastigheds bestemmelse af højre motor	PWM
PWMLeft	PWM signal til hastigheds bestemmelse af venstre motor	PWM

Tabel 5: Signal tabel til IBD på Figur 8

1.6 GPS

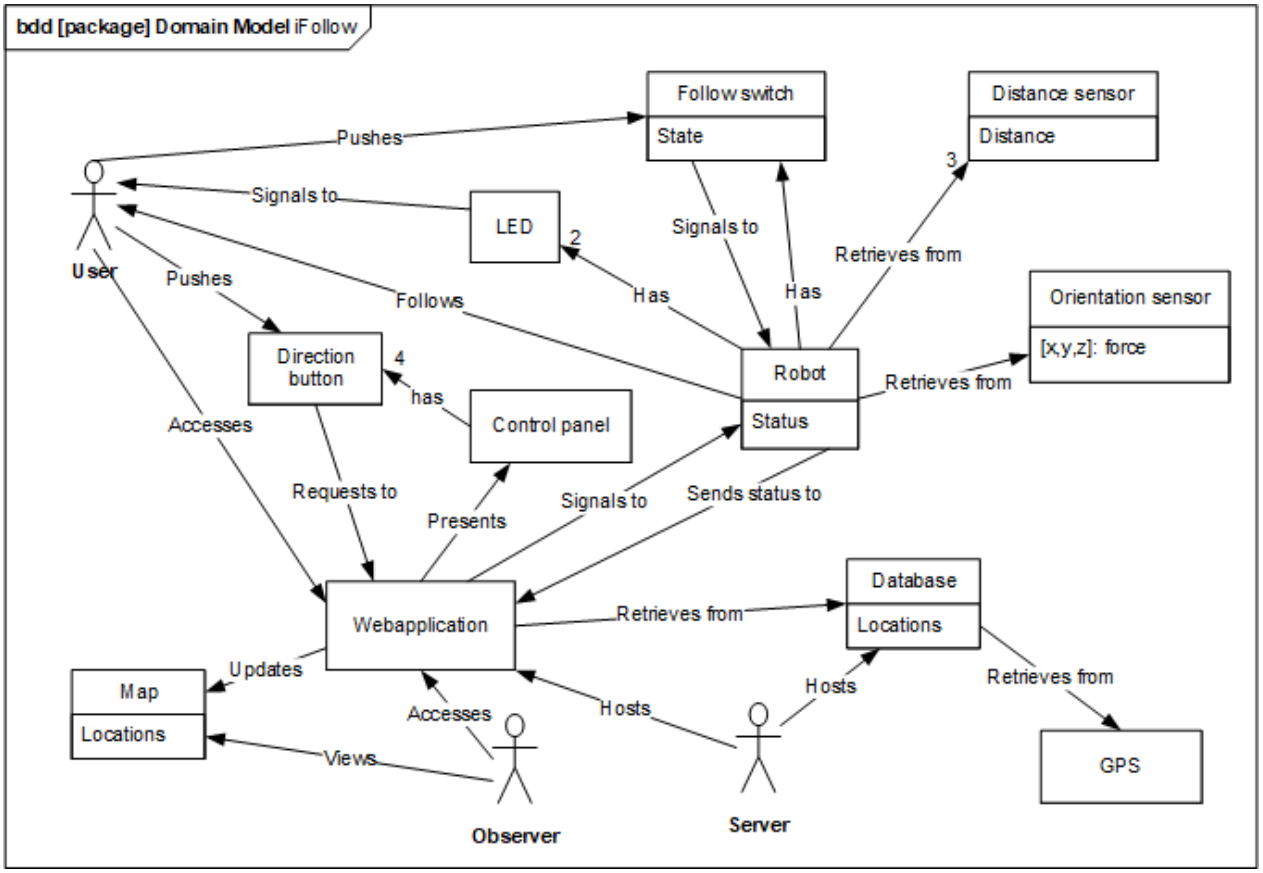
GPS blokken består alene af GPS enheden og denne vil derfor kun blive beskrevet med et IBD. Dette IBD kan ses på Figur 9, her ses det at GPS modulet modtager modtager satelite data og kommunikere med RPi'en vha. UART.



Figur 9: IBD for systemets GPS blok

2 Software arkitektur

I dette afsnit vil arkitekturen for software-delen af iFollow blive beskrevet. Her vil der blive opstillet applikationsmodeller bestående af sekvensdiagrammer, klassediagrammer og andet relevant for de forskellige software-elementer i iFollow. Disse diagrammer vil være udledt af use-cases fra systemets kravspecifikation. (*se Kravspecifikation dokumentet*). For at indlede dette afsnit overordnet, ses en domænemodel for iFollow nedenfor. Denne skal danne et overblik over systemets elementer.



Figur 10: Domænemodel for iFollow systemet

På Figur 10 ovenfor ses en domænemodel for iFollow systemet. Denne er opsat efter en systemdomæneanalyse, hvor centrale dele af systemet identificeres ud fra hovedscenarierne til systemets use cases. Ud fra navneord i hovedscenarierne opsættes og katagoriseres de forskellige domæner i systemet, og relationer mellem disse bestemmes ud fra udsagnsordene. Domænemodellen danner grundlag for den efterfølgende software arkitektur.

2.1 Software allokering

Ud fra domænemodellen i det øvrige afsnit, allokeres tre separate applikationer, én for hver dedikeret proces.

- **WebAPP** - Ansvar for webserver og udlevering af GUI til klient.
- **RobotAPP** - Ansvar for motorstyring med data fra orientation sensor samt afstandssensor.
- **DatabaseAPP** - Ansvar for indsamling af GPS-data samt overføring til database.

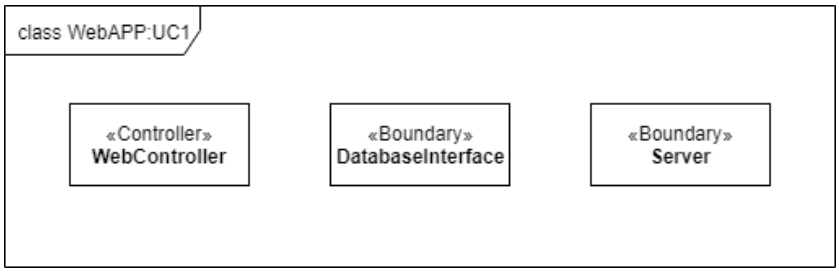
Disse overordnede applikationsklasser beskrives i de næste afsnit, med sekvensdiagrammer, klassediagrammer m.m. Herunder vil applikationernes underklasser, også blive beskrevet.

2.2 WebAPP - Applikationsmodel

I dette afsnit opstilles applikationsmodellen for WebAPP-klassen. Denne klasse sørger for alt kommunikation mellem klient og webserver, herunder præsentation af webapplikation til brugeren. Webserveren skal også sørge for at håndtere brugeraktioner og kalde passende handlers på disse. Klassen indeholder kode skrevet i sprogene: *HTML, CSS og Javascript*. Derudover bruger klassen også pakkerne *jQuery*, *Node.js* samt Node.js-pakken *Socket.io* til håndtering af webserveren. Programmet er skrevet til en Raspberry-pi, som vil køre en lite-version af Raspbian styresystemet. Raspberry-pi'en vil kunne hoste webserveren på dens lokale ip-addresse via. USB eller et hjemmenetværk med WiFi. Nedenfor opstilles klassediagrammer og sekvensdiagrammer opdelt efter relevante use-cases.

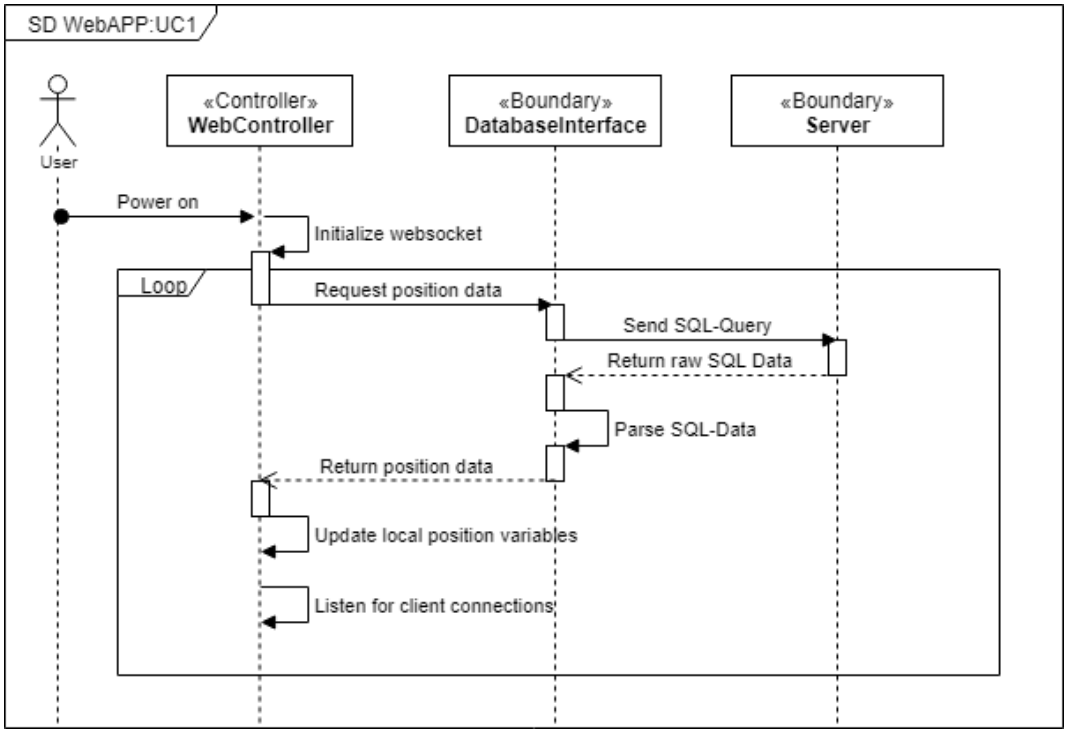
2.2.1 Use-case 1: Start iFollow

Til denne use-case beskrives WebAPP'ens opstartssekvens. Dette kan ikke læses ud fra den fully-dressed use case beskrivelse i *Kravspecifikation*-dokumentet, men er stadigvæk relevant for implementeringen. Nedenfor ses et klassediagram med de relevante klasser til den aktuelle use-case.



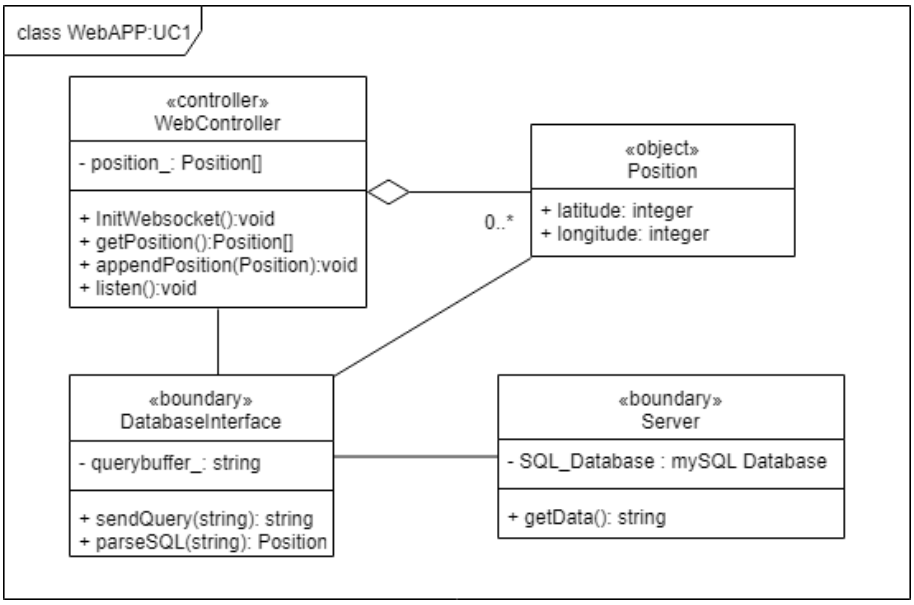
Figur 11: Indledende WebAPP-klassediagram for Use-case 1

For at illustrere hvordan disse klasser kommunikerer med hinanden under systemopstart, er der nedenfor opsat et sekvensdiagram for use-case 1.



Figur 12: Sekvensdiagram for WebAPP for Use-case 1

Ud fra sekvensdiagrammet på Figur 12, opstilles et klassediagram med funktioner og attributter.

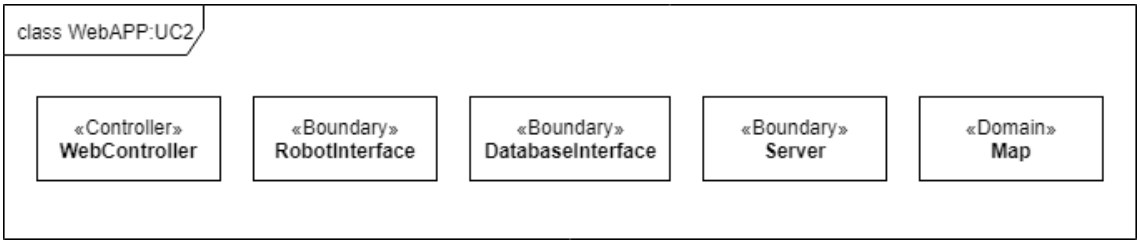


Figur 13: Klassediagram med funktioner og attributter

Ovenfor på Figur 13, ses klassediagrammet for WebAPP med funktionaliteterne fra use-case 1.

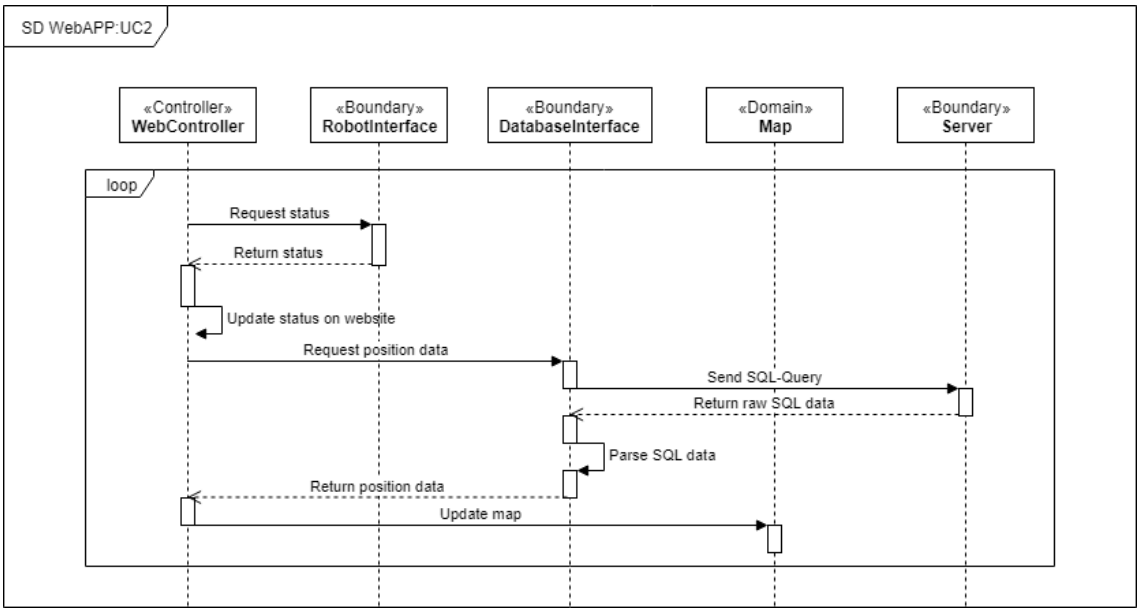
2.2.2 Use-case 2: Follow user

I *Use-case 2: Follow user*, reagerer WebAPP'en ikke direkte på handlinger fra brugeren. Derimod opdateres WebAPP'en uafhængigt af brugerens inputs. Derfor ses der i sekvensdiagrammet for den pågældende use-case et loop, som køres hyppigt for at holde webapplikationen opdateret.



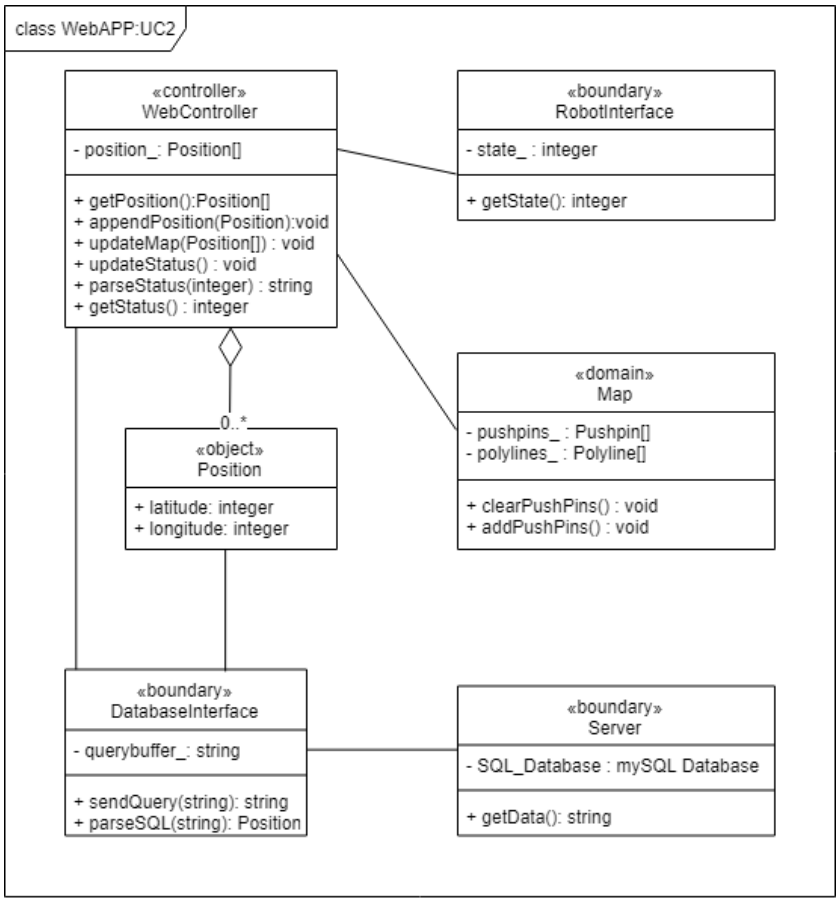
Figur 14: Indledende WebAPP-klassediagram for Use-case 2

Nedenfor ses sekvensdiagrammet for løkken, der sørger for opdatering af hjemmesiden, under følge-funktion.



Figur 15: Sekvensdiagram for WebAPP for Use-case 2

Ud fra sekvensdiagrammet på Figur 15, opstilles et klassediagram med funktioner og attributter.

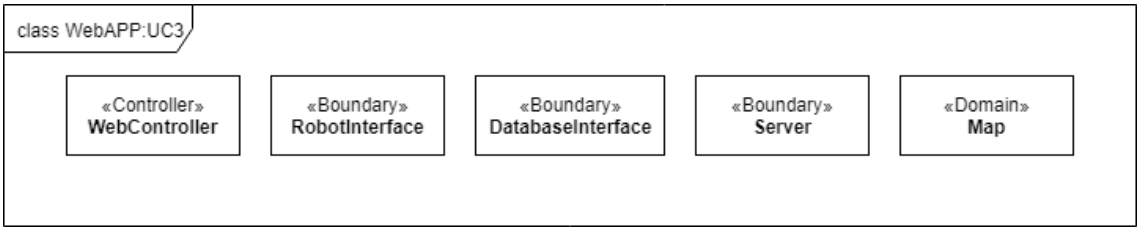


Figur 16: Klassediagram med funktioner og attributter

Ovenfor på Figur 16, ses klassediagrammet for WebAPP med funktionaliteterne fra use-case 2.

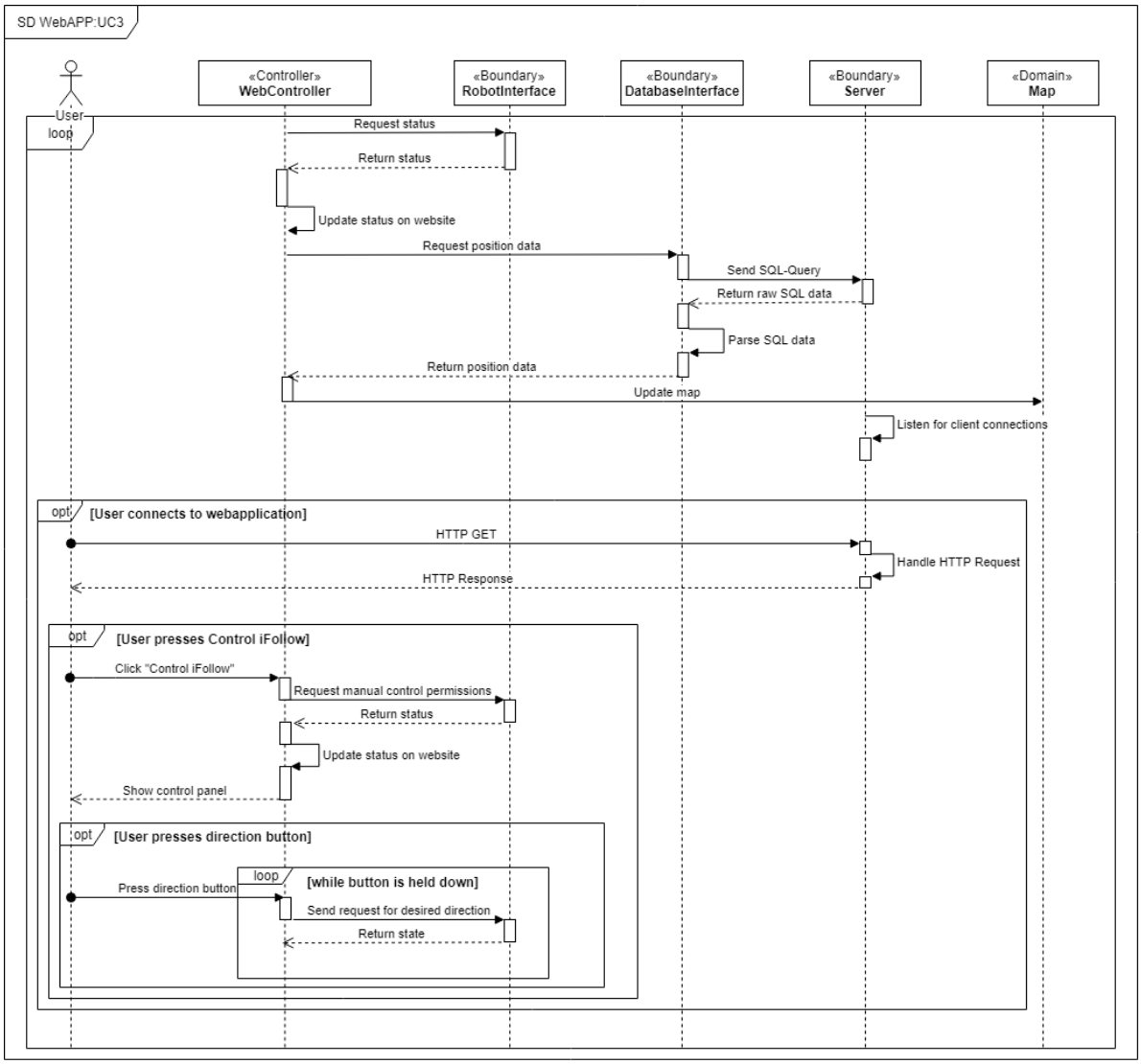
2.2.3 Use-case 3: Control iFollow

I *Use-case 3: Control iFollow*, sættes robotten til at kunne styres manuelt over webapplikationen. Fra webapplikationen skal det være muligt at styre robottens retning og fart. Nedenfor ses et indledende klassediagram, med de tænkte anvendte klasser.



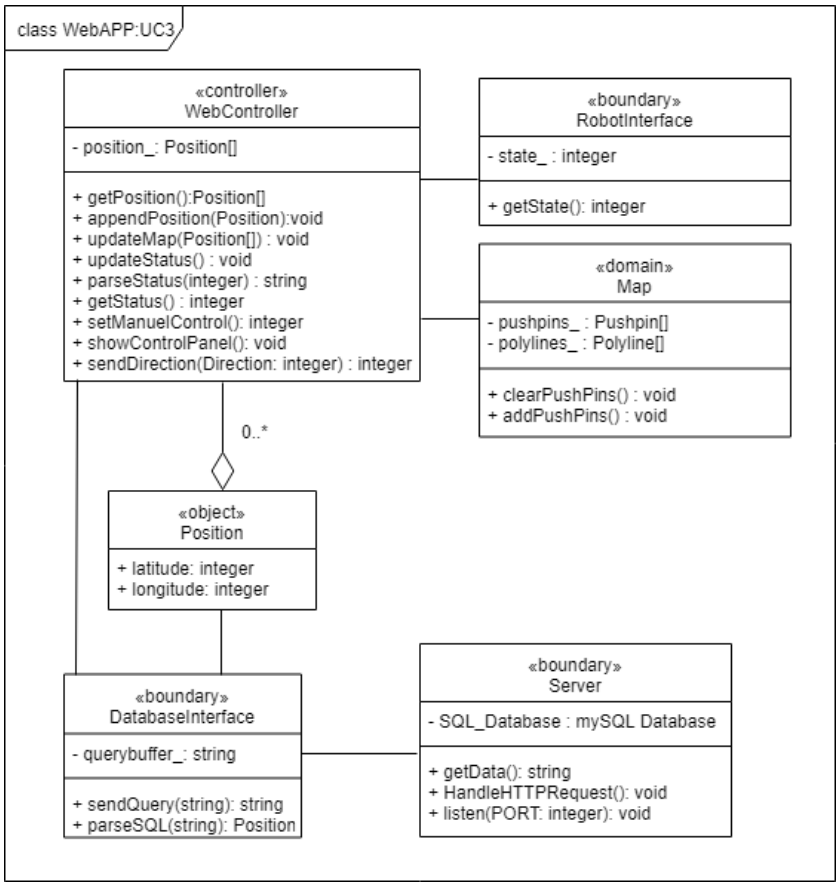
Figur 17: Indledende WebAPP-klassediagram for Use-case 3

Ud fra det ovenstående klassediagram og Use-case beskrivelsen for *Use-case 3: Control iFollow*, opstilles et sekvensdiagram med tænkte handlinger.



Figur 18: Sekvensdiagram for WebAPP for Use-case 3

På den ovenstående Figur 18, kan det ses at loopet fra de forrige sekvensdiagrammer fortsat er med da dette kode stadig køres hyppigt. Derudover er der nu sat to optionals på diagrammet, for at vise eventhåndteringen ved tryk på en retning. Farten bestemmes ved varigheden af et tryk. Det skal nævnes at det jo ikke er brugeren der direkte sender et HTTP-request, men derimod brugerens webbrowser så snart brugeren tilgår webapplikationen. Nedenfor ses et klassediagram med de relevante funktioner fra den pågældende use-case.

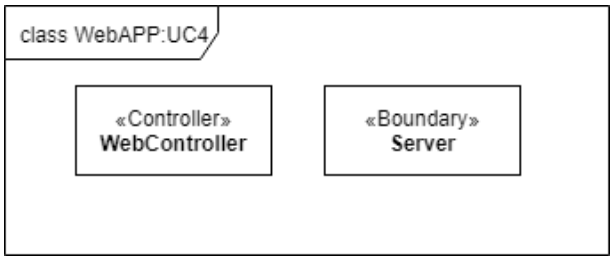


Figur 19: Klassediagram med funktioner og attributter

Ovenfor på Figur 19, ses klassediagrammet for WebAPP med funktionaliteterne fra use-case 3.

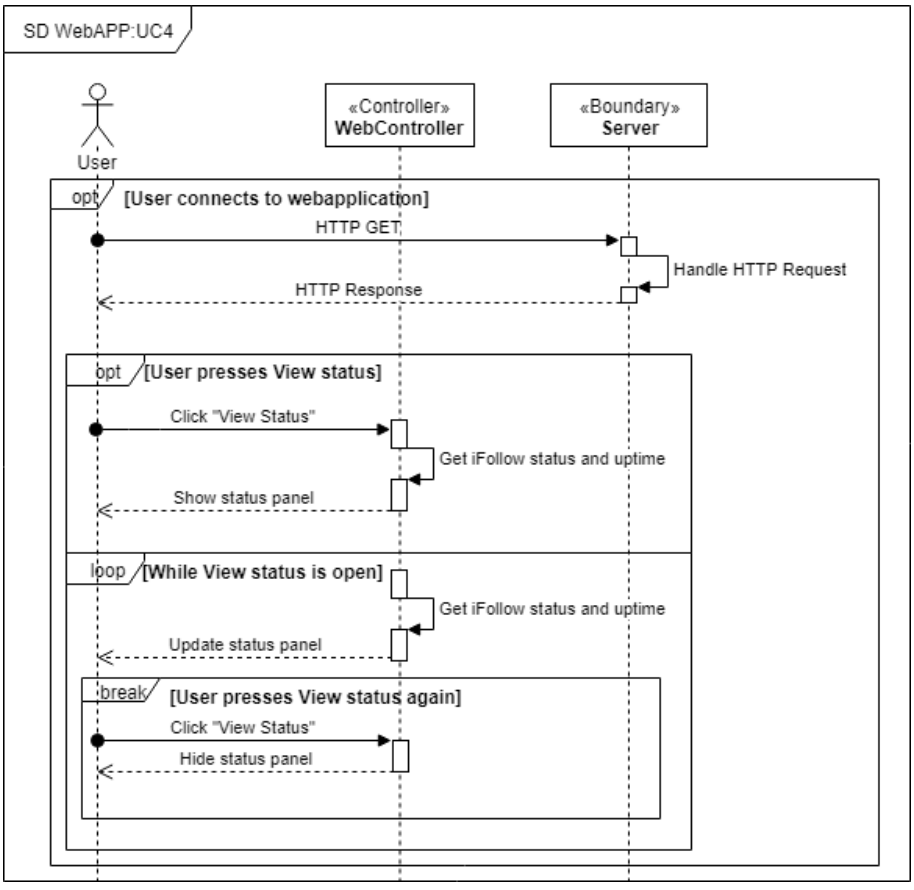
2.2.4 Use-case 4: View status

I *Use-case 4: View status*, vises status omkring iFollows tilstand og driftstid. Til denne applikationsmodel medtages opdateringsløkken ikke. En lokal status-variable bliver derfor opdateret hyppigt uafhængigt. Derfor vil det indledende klassediagram kun indeholde to klasser, som kan ses nedenfor.



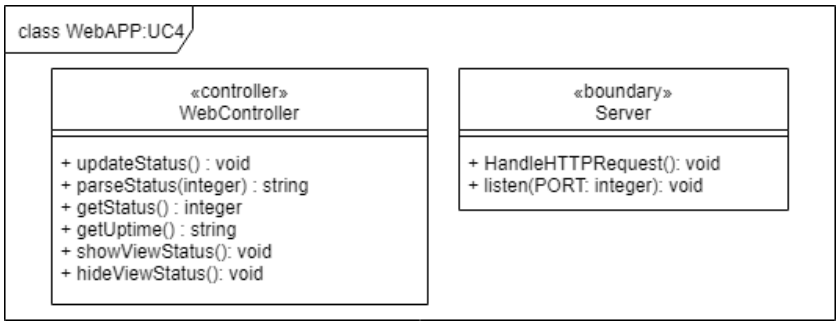
Figur 20: Indledende WebAPP-klassediagram for Use-case 4

Ud fra det ovenstående klassediagram og Use-case beskrivelsen for den pågældende use-case, opstilles et sekvensdiagram. På sekvensdiagrammet er opdateringsløkken ikke medtaget, og man skal derfor forestille sig, at en lokal status-variable opdateres hyppigt, i løkken som kan ses på Figur 12. Nedenfor ses sekvensdiagrammet for Use-case 4.



Figur 21: Sekvensdiagram for WebAPP for Use-case 4

Ud fra det ovenstående sekvensdiagram på Figur 21, opsættes et klassediagram med relevante funktioner og attributter.

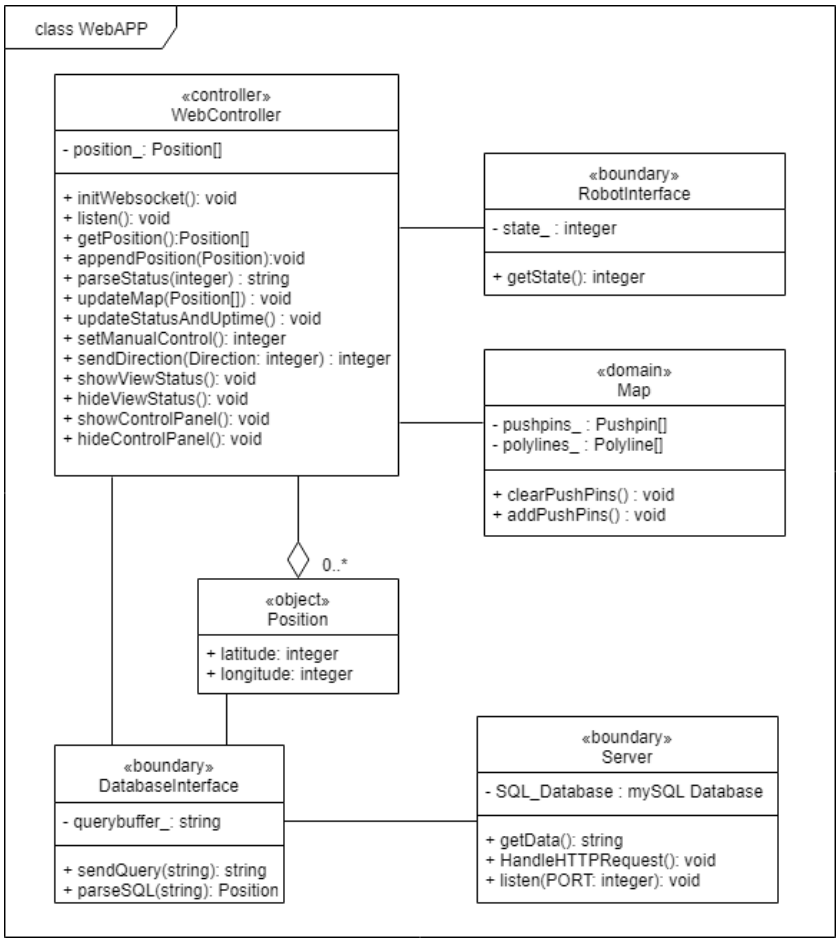


Figur 22: Klassediagram med funktioner og attributter

Ovenfor på Figur 22, ses klassediagrammet for WebAPP med funktionaliteterne fra use-case 4.

2.2.5 Samlet klassediagram

Ud fra klassediagrammerne for de fire use-cases, opstilles et samlet klassediagram for WebAPP. Heri ses de tænkte funktioner for at opfylde de fire use-cases.



Figur 23: Samlet klassediagram med funktioner og attributter

På den ovenstående Figur 23, ses et UML-klassediagram over de tænkte klasser for WebAPP, sammen med deres funktioner og attributter.

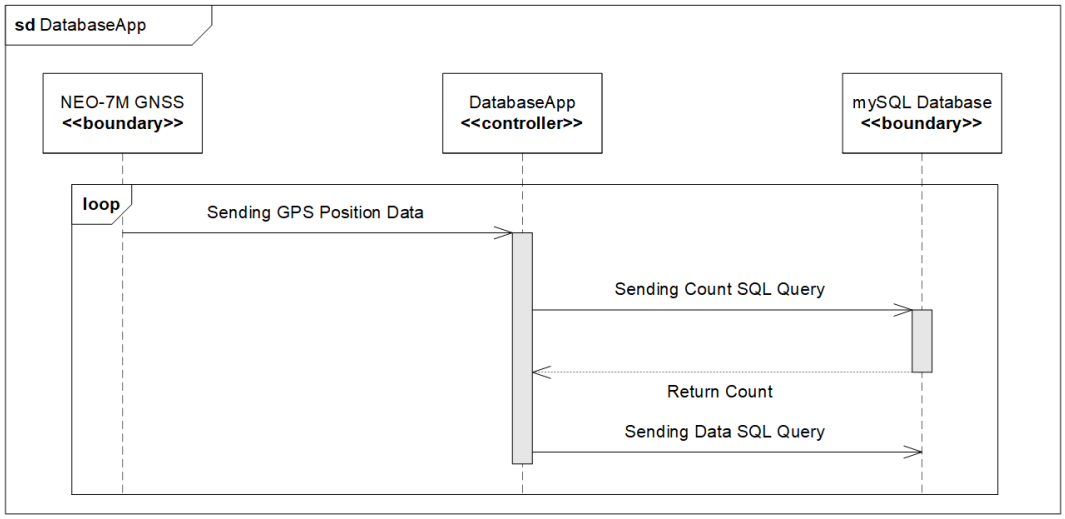
2.3 DatabaseAPP - Applikationsmodel

Dette afsnit vil opstille applikationsmodellen for DatabaseAPP-klassen. Denne klasse vil sørge for at indsamle GPS data fra GNSS modulet, Opdele strengen i de forskellige datatyper, tjekke om hvorvidt at placeringen har ændret sig i forhold til tidligere sendte data til mySQL databasen og sidst men ikke mindst sende data til databasen. Databasen er en mySQL database som benytter MariaDB[1] plugin og phpmyadmin[3] til at tilgå databasen fra en webbrowser. Koden til indsamling af data, opdele dette og sende til databasen er skrevet i sproget C++, med brug at mySQL C++ Connector[2] pakken til at sende querys til databasen. Hele systemet vil køre på den seperat Raspberry Pi kørende Raspbian styresystemet. Denne vil hose databasen på dens lokale IP-adresse over

WiFi. Nedenunder vil der blive opstilt relevante klassediagrammer og sekvensdiagrammer opdelt efter relevante use cases.

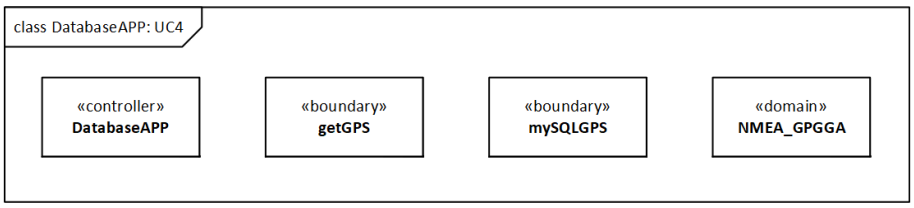
2.3.1 Use case 4: View Status

På Figur 24 kan det overordnede sekvensdiagrammet for DatabaseApp ses. Der kan ses hvordan GPS Modulet, Programmet og Databasen kommunikere med hinanden.



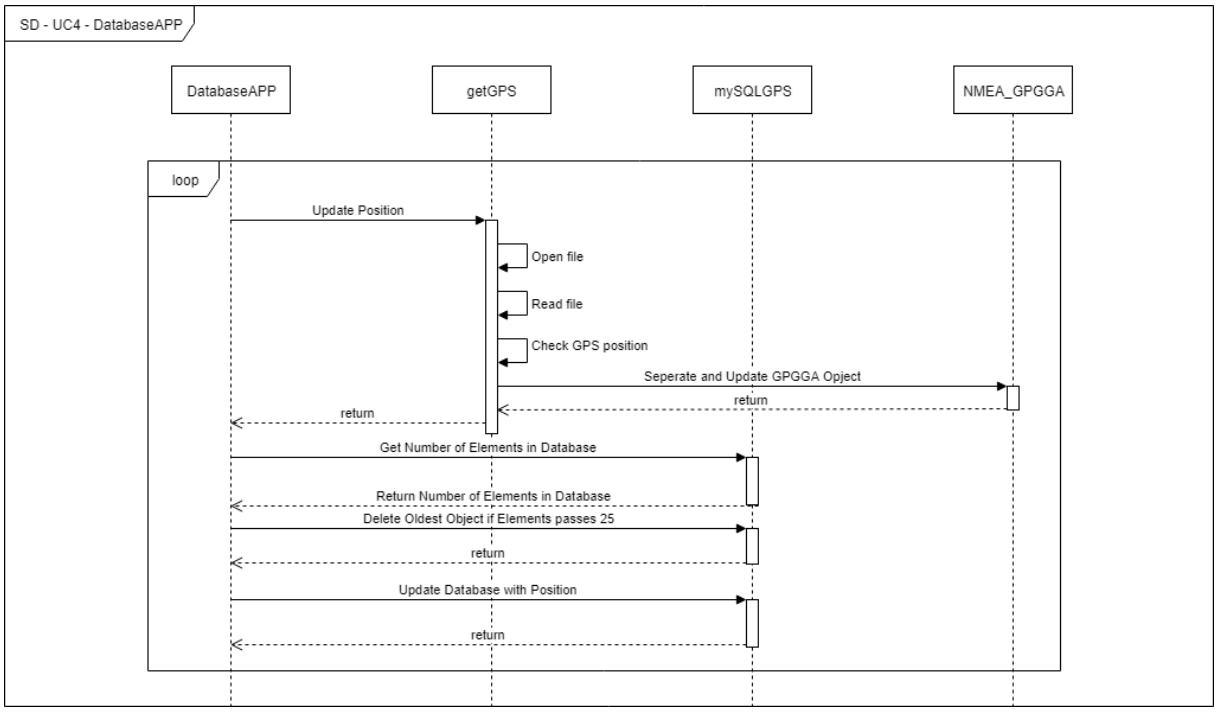
Figur 24: Overordnet Sekvensdiagram for Database Applikation

I Use-case 4, trykker brugeren på knappen View Status på Web aplikationen. Nedenfor vil der blive vist hvordan Database applikationen vil blive implementeret og interagere med mySQL databasen med tiltænkte klasse og sekvensdiagrammer.



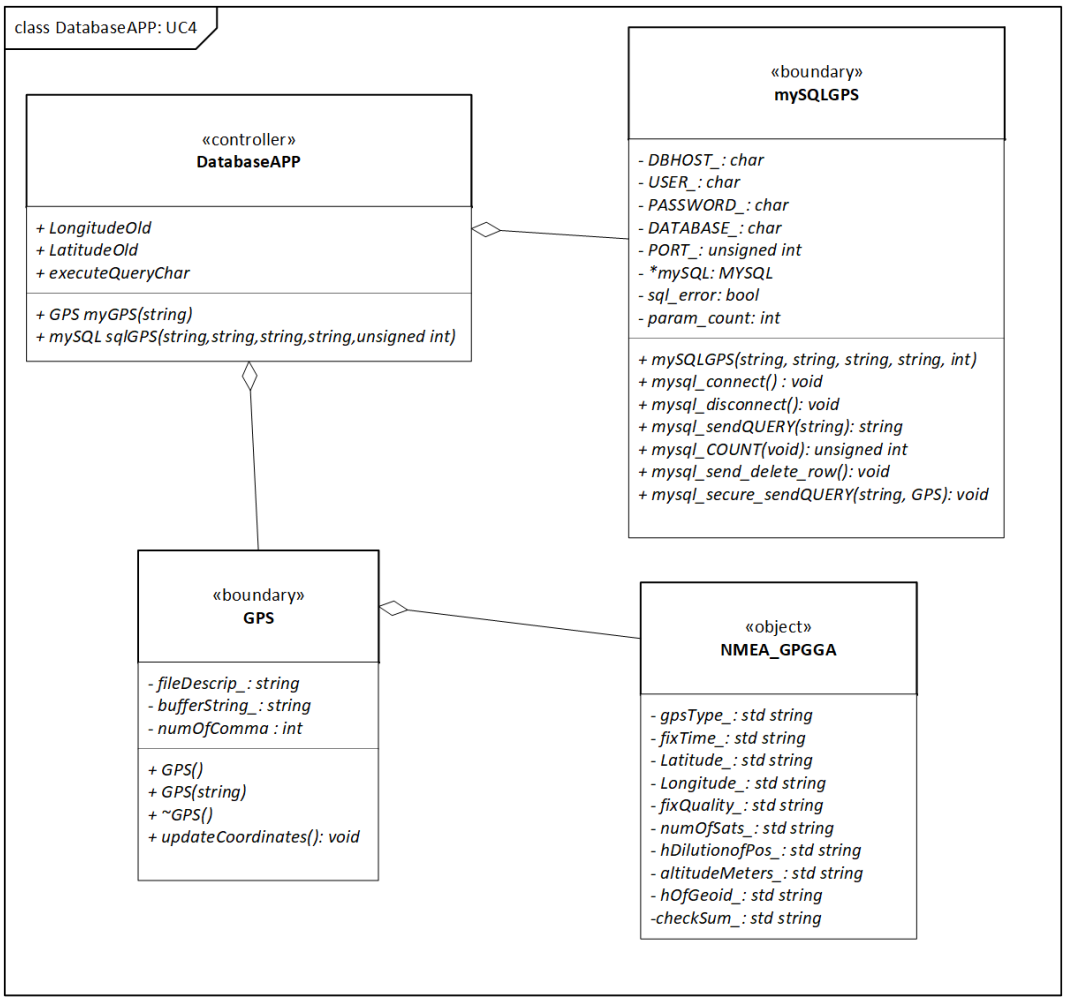
Figur 25: Klassediagram for Database Applikationen

Ud fra ovenstående klassediagram og Use case 4 vil beskrivelsen for Use case 4: View Status, opstilles et sekvensdiagram med tiltænkte handlinger for Database applikationen



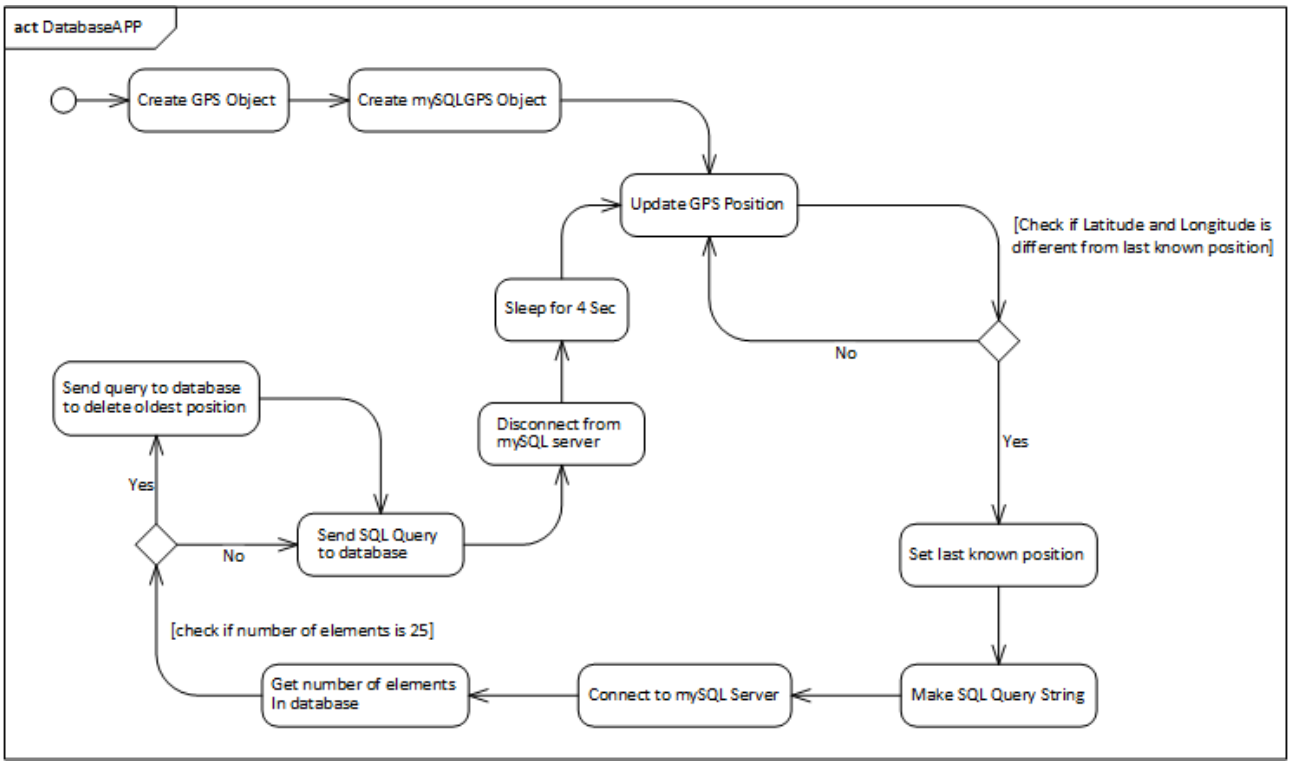
Figur 26: Sekvensdiagram for Database Applikationen

På diagrammet ses at hele systemet for Database Applikationen kører i et loop. Dette er fordi programmet køres separat fra alle de andre applikationer. Dette gør at der ikke er nogle problemer med hvornår Web applikationen henter data og når en position opdateres. Nedenunder her kan der med ses et klassediagram med funktioner og attributer lavet ud fra forgående klassediagram ses på figur 25 og sekvensdiagram ses på figur 26. [**?**]



Figur 27: Udvidet Klassediagram for Database Applikationen

Til videre forståelse om hvordan DatabaseApplikationen fungerer er der blevet udarbejdet et aktivitetsdiagram som skal hjælpe med til at forklare de tjek som der bliver lavet i programmet. Dette diagram kan ses nedenunder på figur 28.



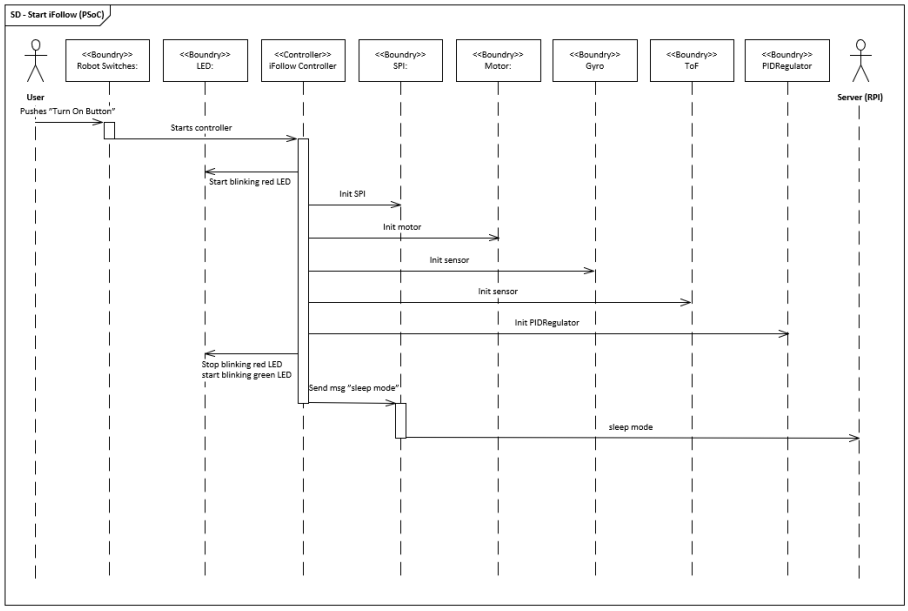
Figur 28: Aktivitetsdiagram for DatabaseApp

2.4 RobotAPP - Applikationsmodel

I dette afsnit opstilles applikationsmodellen for RobotAPP-klassen. Denne læasse sørger for at kontrollere iFollow og sende informationer omkring status videre til WebAPP for senere at kunne udskrive dette til brugeren. Der er på PSoC'en blevet valgt at skrive i sproget C++. Dette er valgt ud fra at det er et stærkere sprog, da det er muligt at lave klasser, som ikke er muligt i ren C.

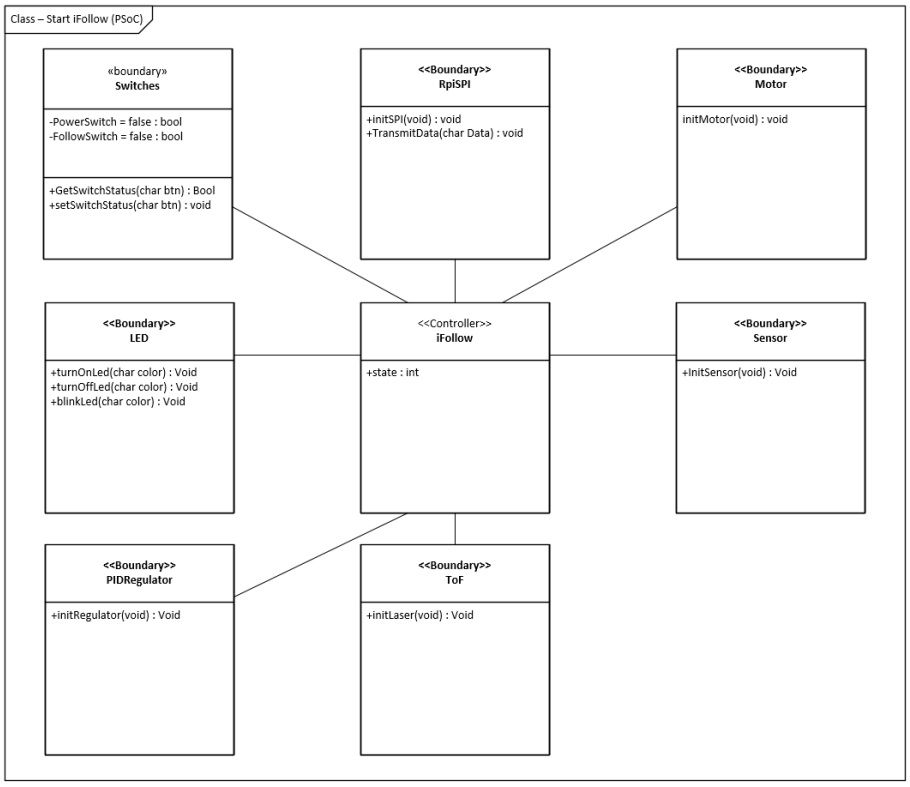
2.4.1 Use-case 1: Start iFollow

Denne usecase omhandler at tænde op for iFollow. Dette kan læses ud fra de fullydessed usecases, som er blevet omskrevet i kravspecificationen. På figur 29 herunder kan sekvensdiagrammet for start iFollow på PSoC ses.



Figur 29: Klassediagram med funktioner og attributter

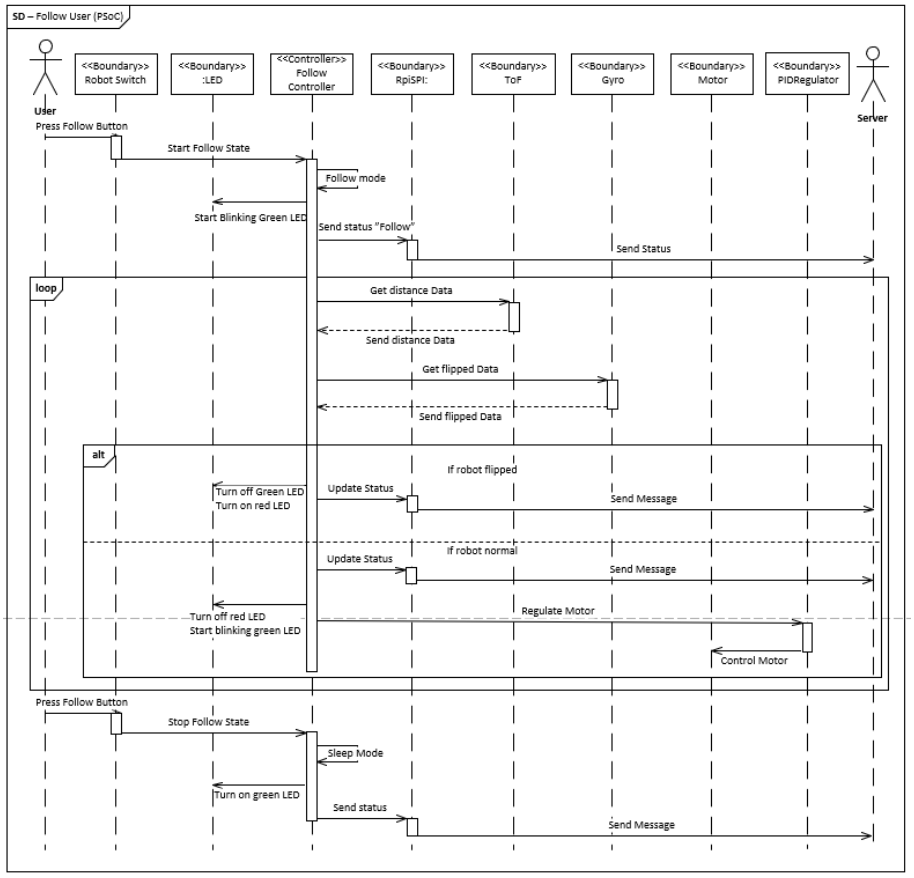
Efter at have lavet sekvensdiagrammet blev der lavet et klassediagram, som indeholder de funktioner som er fundet ud fra sekvensdiagrammet. Der kan dog være flere funktionskald, som bliver fundet i designsfasen.
Klassediagrammet kan ses på figur 30 herunder.



Figur 30: Klassediagram med funktioner og attributter

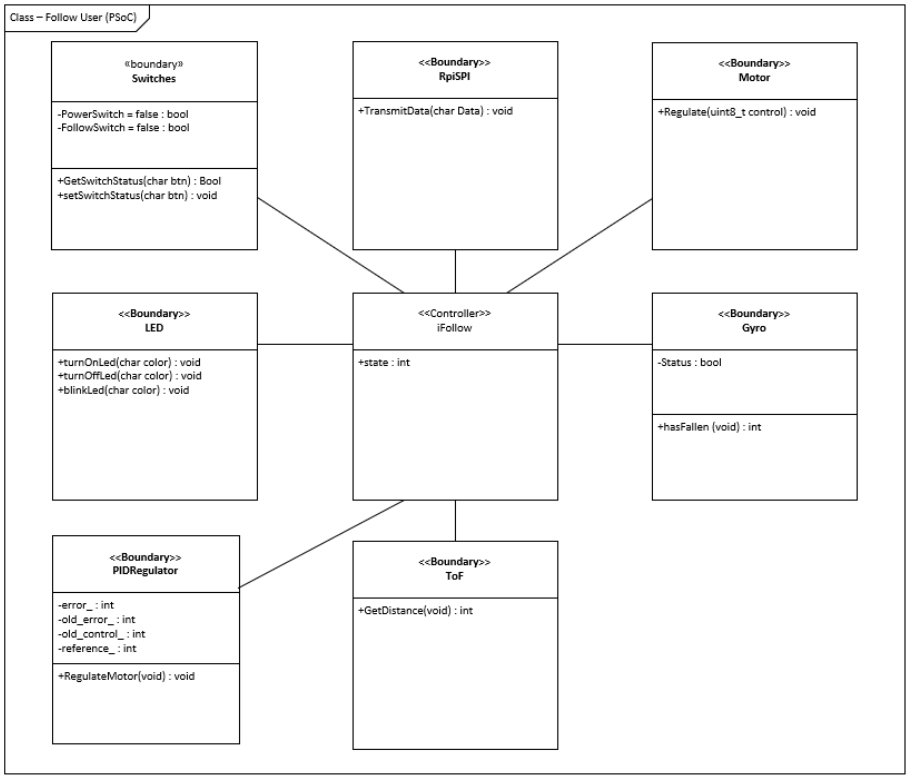
2.4.2 Use-case 2: Follow user

I Use-case 2: Follow user, reagerer PSoC'en på en handlig fra brugen i form af et tryk på en knap. Denne knap sørger for at få iFollow til at starte usecasen op og begynde at regulere på sensor inputs. Sekvensdiagrammet kan ses på figur 31 herunder.



Figur 31: Klassediagram med funktioner og attributter

Ud fra sekvensdiagrammet blev der så lavet et klassediagram med de funktioner som der skulle bruges til at implementere dette system. Klassediagrammet kan ses på figur 32 herunder.

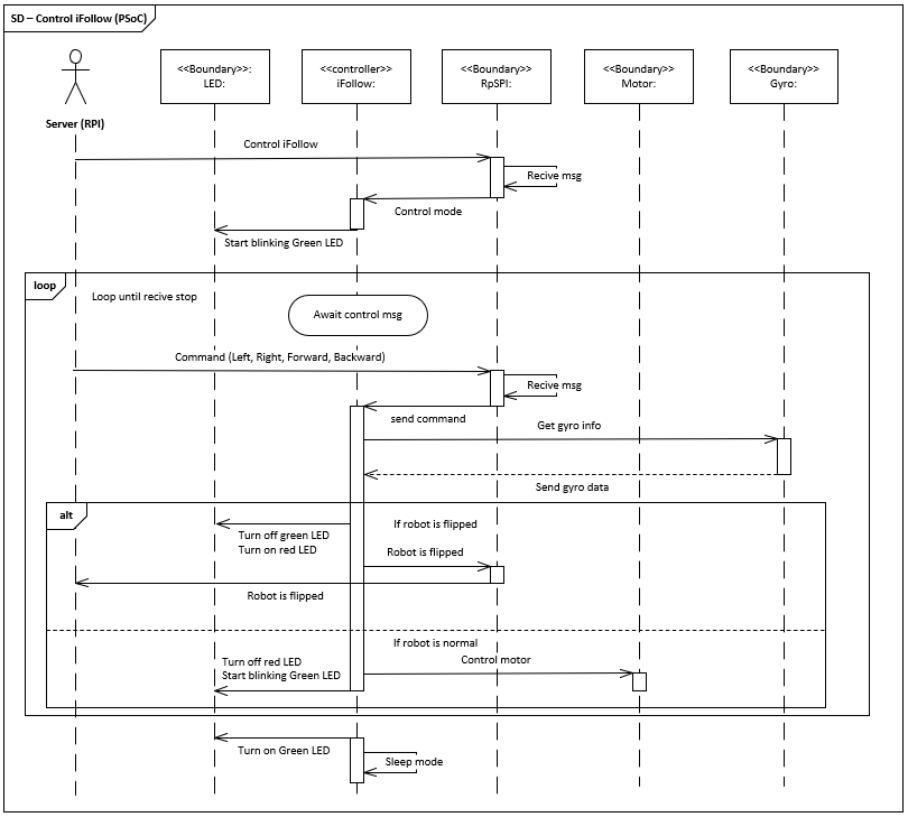


Figur 32: Klassediagram med funktioner og attributter

Som man kan se er der ikke så mange funktioner men dette kan dog ændre sig i løbet af designet da der med god chance skal bruges nogle flere funktioner til f.eks. at hente data ind eller styre motorerne.

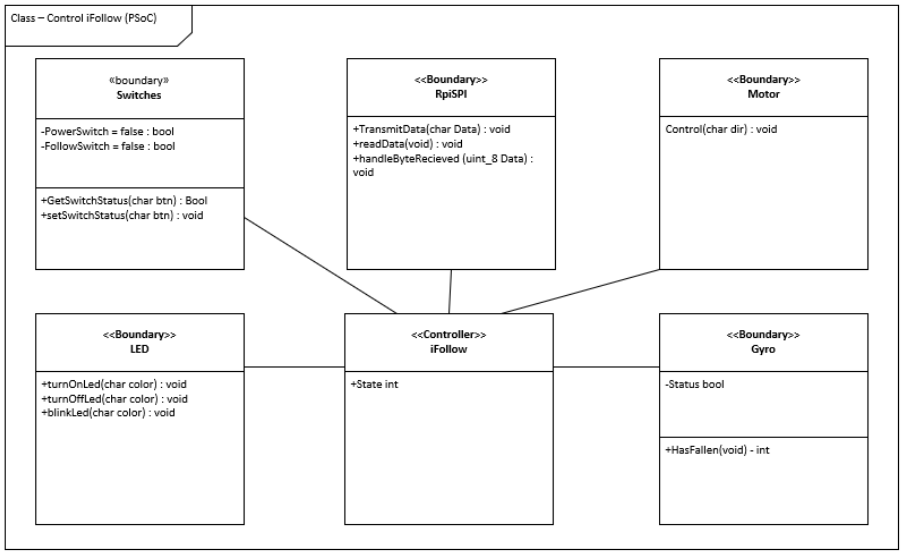
2.4.3 Use-case 3: Control iFollow

Denne usecase omhandler fjernstyret kontrol af iFollow så brugeren har mulighed for at kører robotten tilbage hvis den er kørt væk. Denne usecase afhænger af kommandoer fra en RPI, som kommunikere ved hjælp af en SPI bus til PSoC'en. Sekvensdiagrammet for denne UC kan ses på figur 33 herunder.



Figur 33: Klassediagram med funktioner og attributter

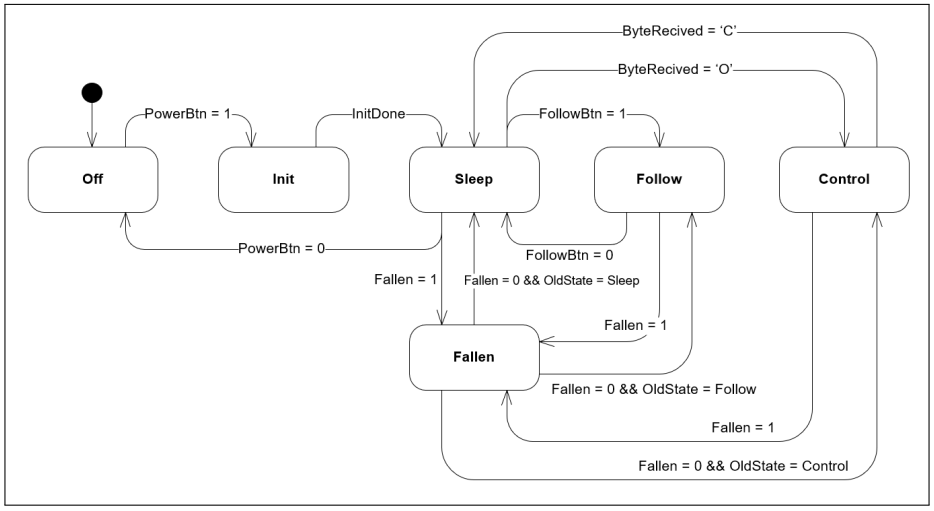
Ud fra klassediagrammet blev der opstillet et klassediagram med de funktioner, som blev fundet ud fra sekvensdiagrammet. Klassediagrammet kan ses på figur 35 herunder.



Figur 34: Klassediagram med funktioner og attributter

2.4.4 State machine for PSoC

Da PSoC'en har forskellige stadier den kan være i er der blevet lavet en state machine, som viser, hvilket statie robotten er i, hvis der kommer et specefikt input. State machinen kan ses på figur **??** herunder.



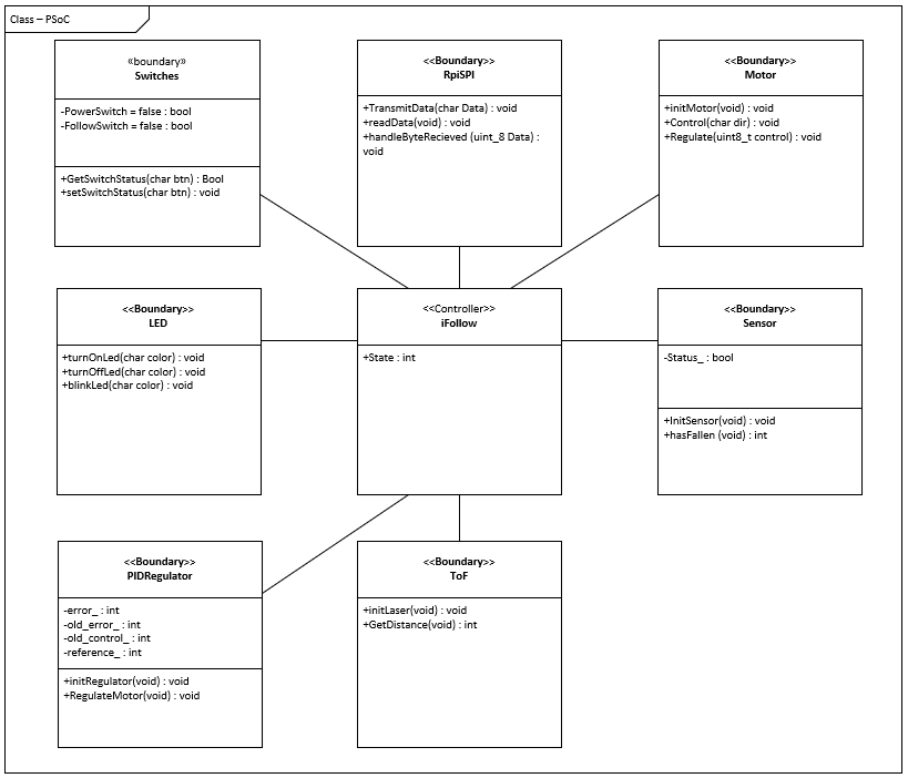
Figur 35: State machine for PSoC

Som man kan se på state machinen, kan robotten robotten, så snart den er startet hoppe imellem sleep, control og follow. Hvis der så skulle ske det at robotten vælter kan den gå i fallen stadiet.

Hvis den så bliver rejst op igen vil den fortsætte, hvor den kom fra.

2.4.5 Samlet klassediagram for PSoC

Efter at have lavet alle applikationsmodellerne for PSoC, kunne der opskrives et samlet klassediagram. Dette klasse diagram indholder alle funktioner fra usecase klassediagrammerne og vil derfor indeholde de nødvendige funktioner for at blokkene kan snakke sammen. Klassediagrammet kan ses på figur 36 herunder.



Figur 36: Klassediagram med funktioner og attributter

3 Anvendte protokoller

Dette afsnit vil beskrive de anvendte protokoller mellem software-applikationerne. Disse beskrivelser vil både dække de generelle kommunikationsprotokoller som: *SPI*, *I2C* og *UART*, men også specifikt definerede protokoller anvendt mellem applikationerne.

3.1 SPI - RPI til PSoC

Imellem PSoC og RPI er der blevet brugt en SPI kommunikation. Denne kommunikationslinje er valgt ud fra den meget hurtige dataoverførsel og rimelig simple protokol.
Måden PSoC og RPI kommunikere med hinanden på er ved at PSoC'en altid sender sin status til RPI'en. Ligemeget, hvilket kommando, som bliver sendt fra RPI'en.
For at RPI'en kan få lov til at kontrollere robotten er det nødvendigt at gå i kontrol mode på PSoC'en. Dette bliver gjort ved at sende et 'o' til PSoC'en. Herefter vil PSoC'en være i kontrol mode og kan tage imod kommandoer fra RPI'en. Disse kommandoer kan være 'f' for forward, 'b' for backward, 'l' for left og 'r' for right. Jo flere gange disse kommandoer bliver sendt efter hinanden vil gøre at robotten kører hurtigere og hurtigere indtil den rammer sin tophastighed.
For at stoppe kontrolmode bliver der sendt et 'c' til PSoC'en, som herefter vil gå ud af kontrolmode.

I tabellen herunder kan de forskellige kommandoer ses.

Sender enhed	Sendt	Betydning
PSoC	0	PSoC er i OFF mode
PSoC	1	PSoC er i Init mode
PSoC	2	PSoC er i Sleep mode
PSoC	3	PSoC er i Control mode
PSoC	4	PSoC er i Follow mode
PSoC	5	PSoC er i Fallen mode
RPI	'o'	Start kontrol mode
RPI	'c'	Stop kontrol mode
RPI	'f'	Kør robot fremad
RPI	'b'	Kør robot tilbage
RPI	'l'	Kør robot til venstre
RPI	'r'	Kør robot til højre

4 Referenceliste

[1] Mariadb plugin for mysql database. URL `https://mariadb.org/`. Sidst besøgt 23 Maj 2019.

[2] Library for mysql in c++. URL `https://packages.debian.org/stretch/default-libmysqlclient-dev`. Sidst besøgt 23 Maj 2019.

[3] phpmyadmin. URL `https://www.phpmyadmin.net/`. Sidst besøgt 23 Maj 2019.