Bilag L - Arkitektur

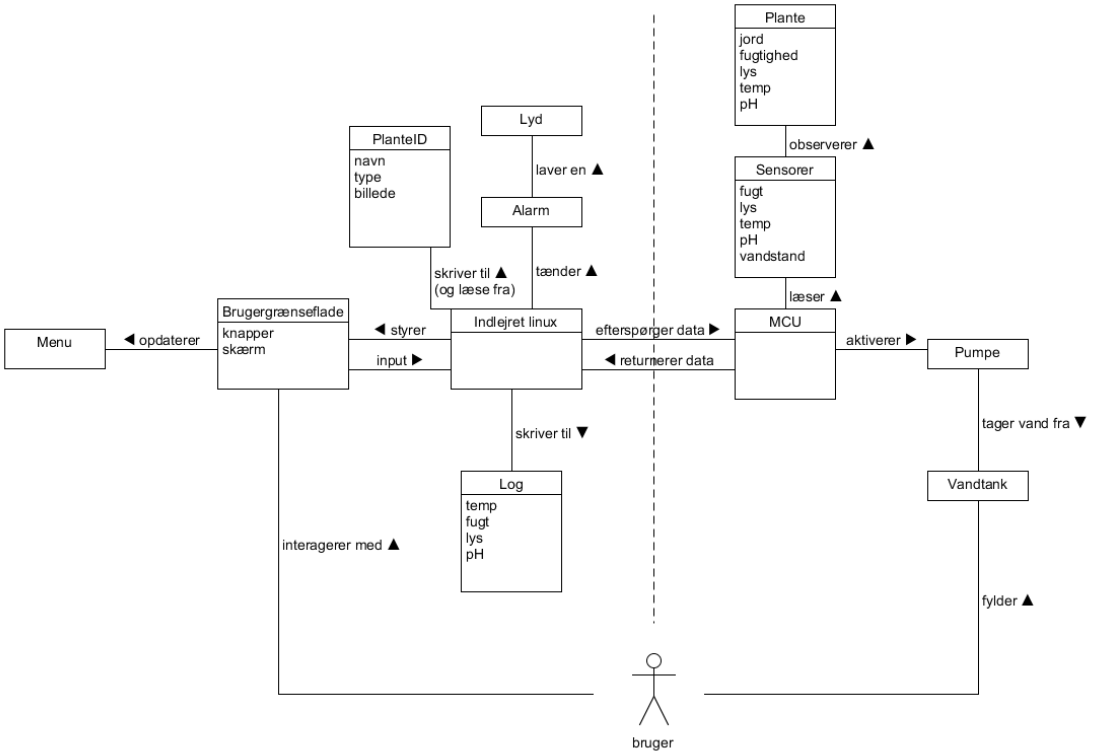
Juni 2023

Studienummer	Navn	Studieretning
20062333	Anette Olesen Lihn	SW
202009421	Jonas Gjørup Eriksen	SW
20002248	Kristian Lund	SW
202008660	Michelle Valentine Petersen	SW
201409521	Szymon Palka	SW
202105139	Asger Ajs Dam	E
201705621	Jan Jakob Agricole Iversen	E
202110470	Rasmus Haugbølle Thomsen	E

Indhold

1 Indledning

Projektets arkitektur er udarbejdet på grundlag af vores domænemodel, med vores kravsspecifikationer - og de Use Cases denne er baseret på - som yderligere krav. Se Bilag J - Domænemodel udvikling, for detaljer om domænemodellen.



Figur 1: Endelig udgave af vores domænemodel.

Bemærk at termerne er ikke overensstemmende med den senere brug. Som man kan se, poll'er brugergrænsefladen potten, og det er her vores model stiller krav til den overordnede arkitektur for både hardware og software. Vi beskriver denne, og undersystemer, i det følgende i en række sysML og UML diagrammer.

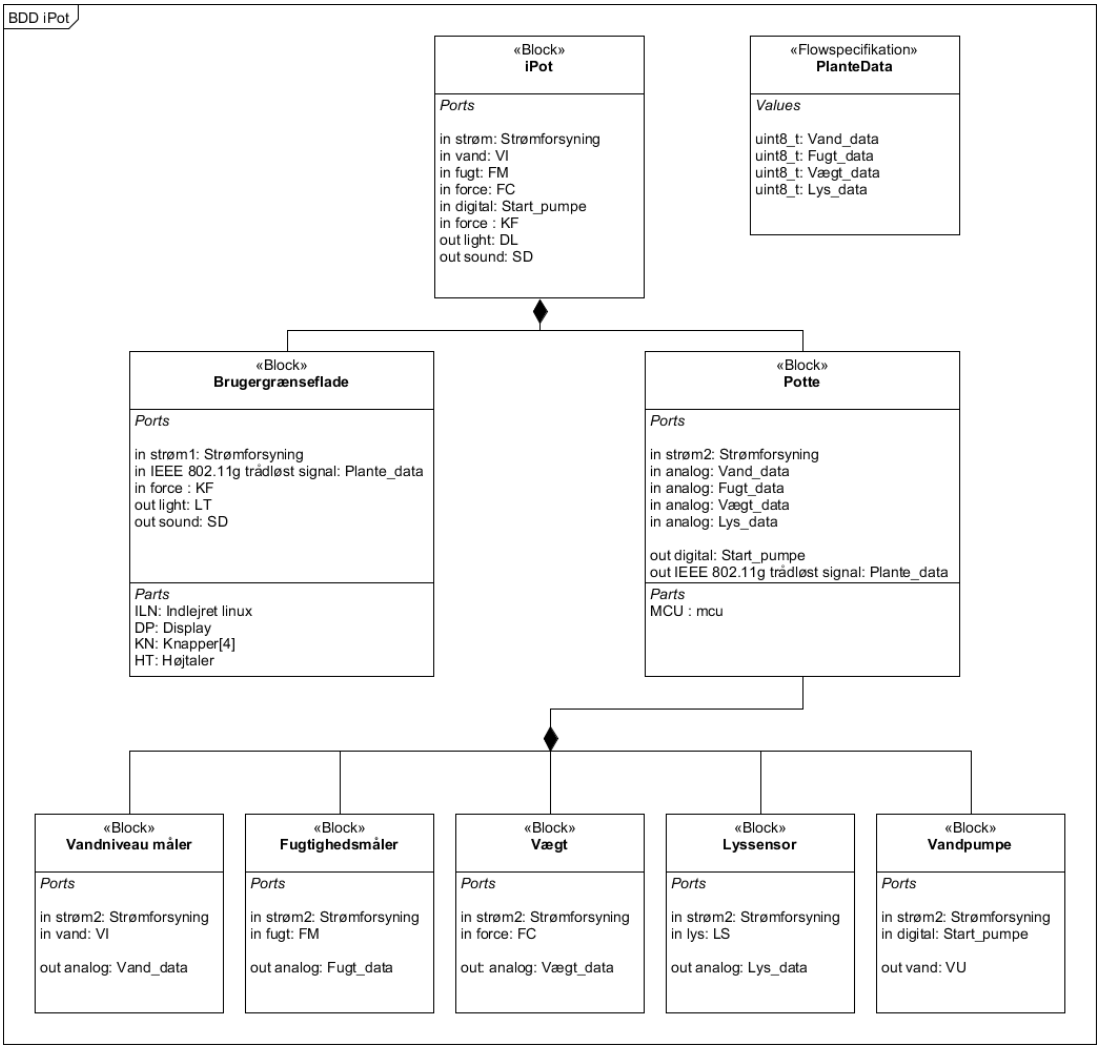
2 Overordnet arkitektur

2.1 Overordnet BDD

Vi har i arbejdet med applikationsmodellen afgrænset projektet til kun at indeholde de 4 mest essentielle sensorer, og altså udeladt:

- Ph måler - hvis data vi ikke har viden nok om forskellige planter til at kunne tolke signalet fornuftigt.
- Temperatur måler - da vi ikke ved nok om temperatur-gradienter i jord af forskellig fugtighed, til at kunne tolke signalet fornuftigt.

Med dette i mente kan vi "oversætte" vores domænemodel til et overordnet BDD-diagram:



Figur 2: Opdateret BDD for iPot

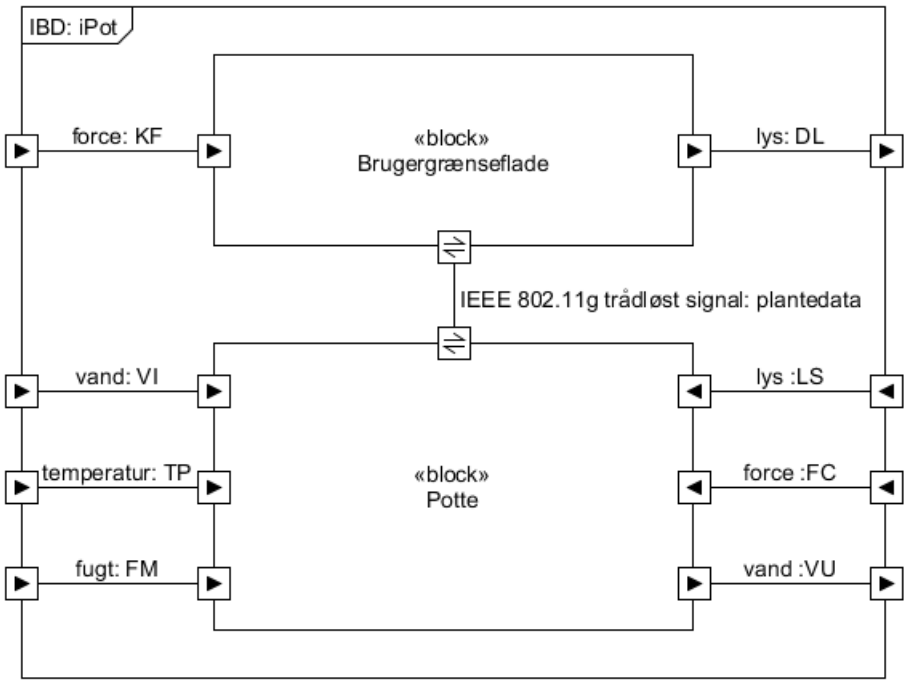
Herunder ses en beskrivelse af de forskellige blokke fra Figur 2.

Bloknavn	Funktionsbeskrivelse
Vandniveaumåler	Vandniveaumåleren er en nødvendig del af systemet, da den vil kunne indikere, når der er et lavt vandniveau i vandtanken, og systemet kan herfra gøre brugeren opmærksom på det via brugergrænsefladen.
Fugtighedsmåler	Fugtighedsmåleren er til for at indikere hvornår plantens jord er blevet for tør og dermed skal vandes. Det er altså denne sensors data, som starter vandingsprocessen.
Vægt	Vægten sidder under potten og vejer løbende ændringer i plantens masse. På denne måde kan brugeren holde øje med plantens udvikling fra planten sættes i potten.
Lyssensor	Lyssensoren måler det ambiente lys i rummet hvor planten står. Meningen er, at brugeren vil kunne få et indblik i, om det valgte sted giver planten nok lys.
Vandpumpe	Vandpumpen sidder i potten, og pumper vand til planten, når fugtighedssensoren får en måling fra jordet, der indikerer, at det er tid til en vanding.

Tabel 1: Udvidet bloktabel for iPot

2.2 Overordnet IBD

For også at vise sammenhængen med systemets omverden, samt planten og jorden, har vi tilsvarende oversat domænemodellen til følgende IBD, som tydeliggør signalerne mellem de overordnede blokke:



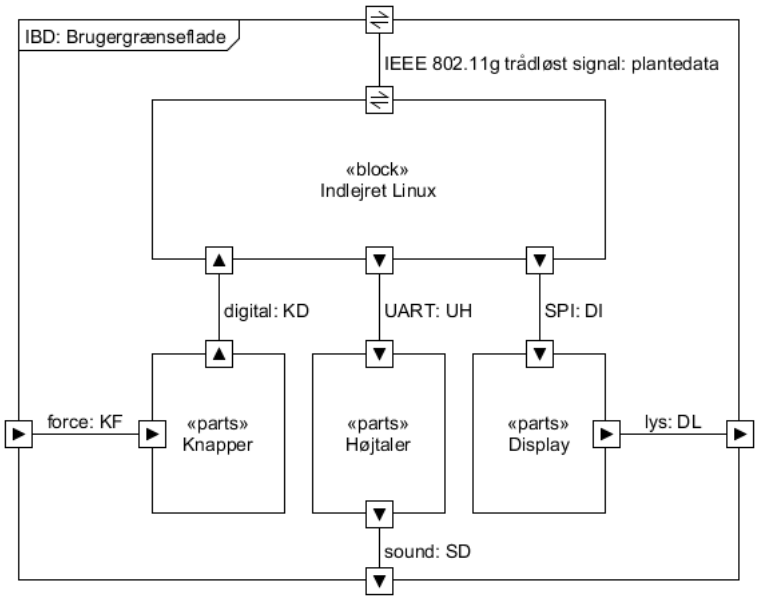
Figur 3: Overordnet BDD for iPot

I det følgende beskrives de to overordnede blokke, Brugergrænseflade og Potte, først ift. til deres hardware og derefter software. Vi har i begge tilfælde, for overskuelighedens skyld, valgt at opdele arkitektur-beskrivelsen i de to blokke. Signalerne fra Figur. 3 vil blive forklaret her.

3 HW arkitektur

3.1 Brugergrænseflade IBD

For at tydeliggøre de interne signaler, men også vise den naturlige deling, der findes i dette projekt, altså mellem brugergrænsefladen og potten, har vi valgt, at lave et IBD for hver af disse to dele. Herunder på figur 5 ses et IBD for brugergrænsefladen, hvor signalerne fra omverden og mellem blokkens dele er vist. Der gøres opmærksom på, at dette er systemarkitektur, og selvom forbindelser som SPI eller UART er mere komplekse, er de her simplificeret til et samlede signaler.



Figur 4: IBD for Brugergrænseflade

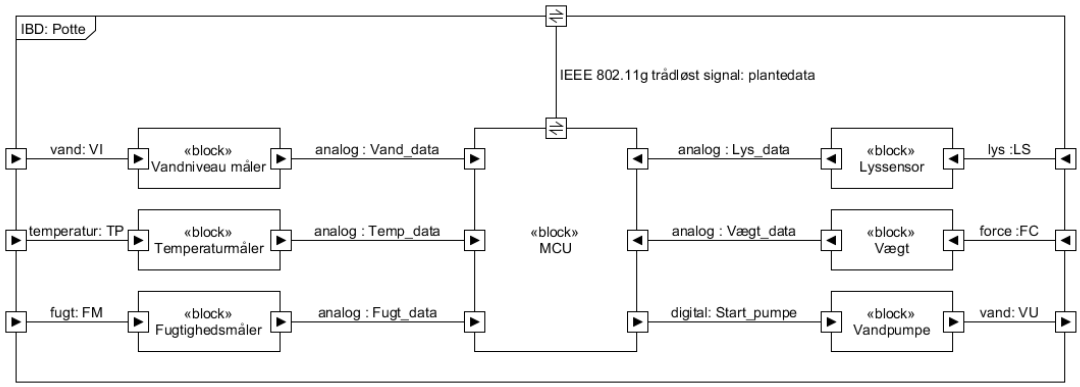
Herunder i tabel 5, ses en oversigt over de forskellige signaler i brugergrænsefladen.

Signaltype	**Funktionsbeskrive**	**Område**	**Port 1 (source)**	**Port 2 (destination)**
force	Fysisk tryk på en knap		Udefra	Knapper
digital	Logisk signal fra knapperne til Indlejret Linux	0-5V	Knapper	Indlejret Linux
lys	Synligt lys, som kommer fra displayet	400-700nm bølgelængde	Display	Bruger
sound	Lyd der spilles fra højtaleren	Hørbart lyd	Højtaler	Bruger
IEEE 802.11g	Kommunikation mellem potten og brugergrænsefladen	2.4GHz mikrobølge	Indlejret Linux	**Potte**
UART	Kommunikation til SOMO og dermed højtaleren	0-3.3V	Indlejret Linux	Højtaler
SPI	Kommunikation til displayet	0-3.3V	Indlejret Linux	Display

Tabel 2: Signaltabel for brugergrænseflade

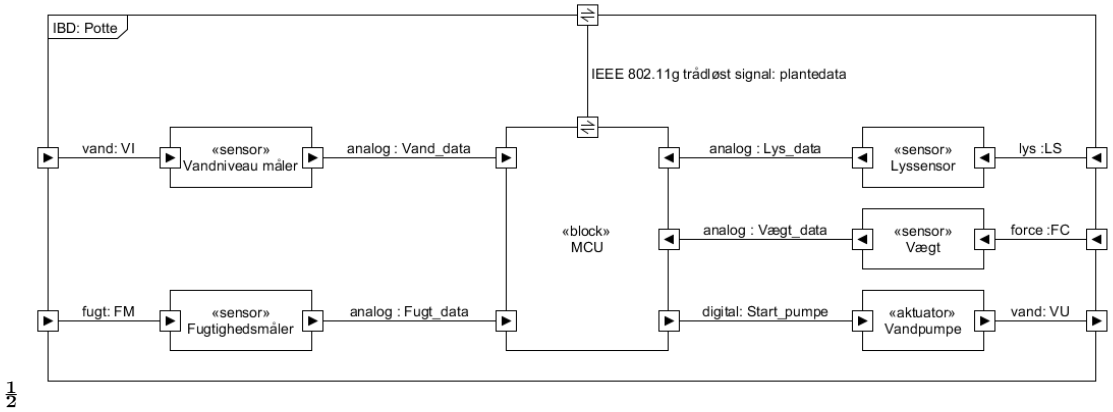
3.2 Potte IBD

Pottens interne signaler er også tydeliggjort i et IBD herunder på figur 6. Mange af disse signaler/forbindelser er sensorenes forbindelse til den virkelige verden, og fra sensorene til MCU-blokken findes de analoge signaler, som er blevet lavet i sensorene.



Figur 5: IBD for Potte

Ligesom det udvidede BDD (figur 3) blev opdateret efter afgrænsning til kun de 4 vigtigste sensorer (figur 4), så er IBD'et for potten også blevet opdateret. Denne nye udgave kan ses herunder på figur 7.



Figur 6: IBD for Potte ny

Herunder i tabel 6 ses en oversigt over de forskellige signaler i potten.

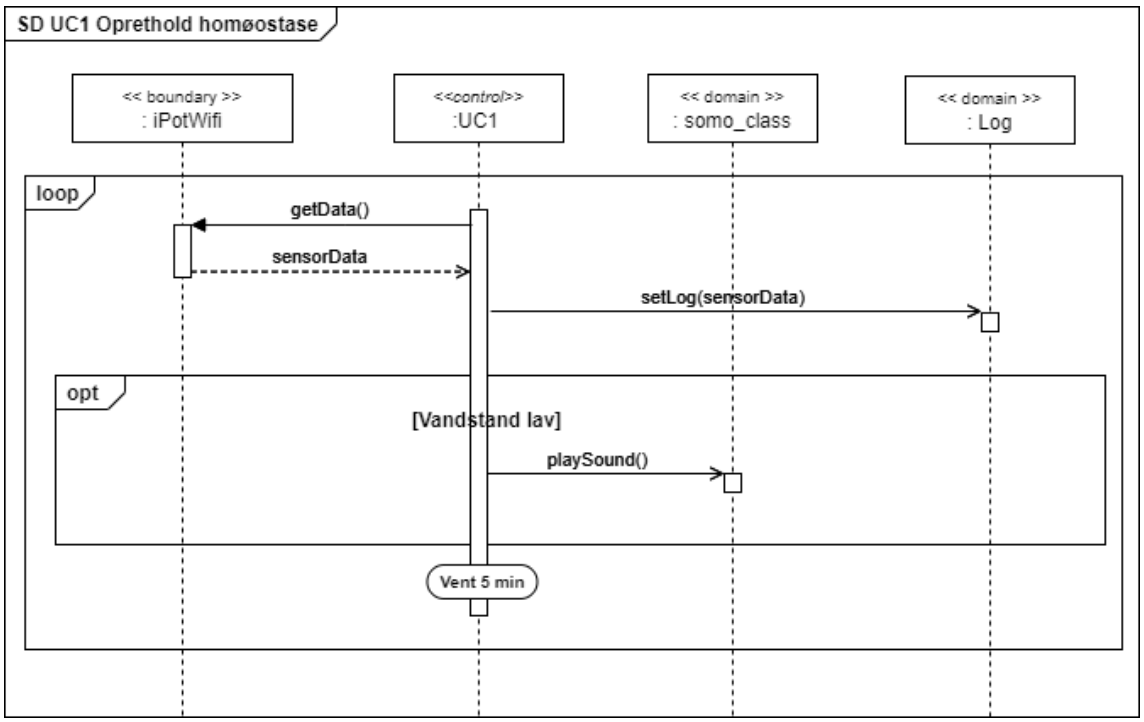
Signaltype	**Funktionsbeskrive**	**Område**	**Port 1 (source)**	**Port 2 (destination)**
analog	Analogt signal fra sensorene til MCU	0-5V	Vandniveaumåler	MCU
			Fugtighedsmåler	
			Lyssensor	
			Vægt	
digital	Logisk signal der tænder pumpen	0-5V	MCU	Vandpumpe
IEEE 802.11g	Kommunikation mellem potten og brugergrænsefladen	2.4GHz mikrobølge	MCU	**Indlejret Linux**
lys	Synligt lys, som opfanges af lyssensoren	400-1100nm bølgelængde	Udefra	Lyssensor
fugt	Fugtighed i plantens jord	0-100%	Udefra	Fugtighedssensor
force	Fysisk pres på vægten	0-6kg	**Potte**	Vægt
vand	Fysisk vand	0-500mL	Udefra	Vandniveaumåler
			Vandpumpe	**Potte**

Tabel 3: Signaltabel for Potte

4 SW arkitektur

4.1 Brugergrænseflade - RPI

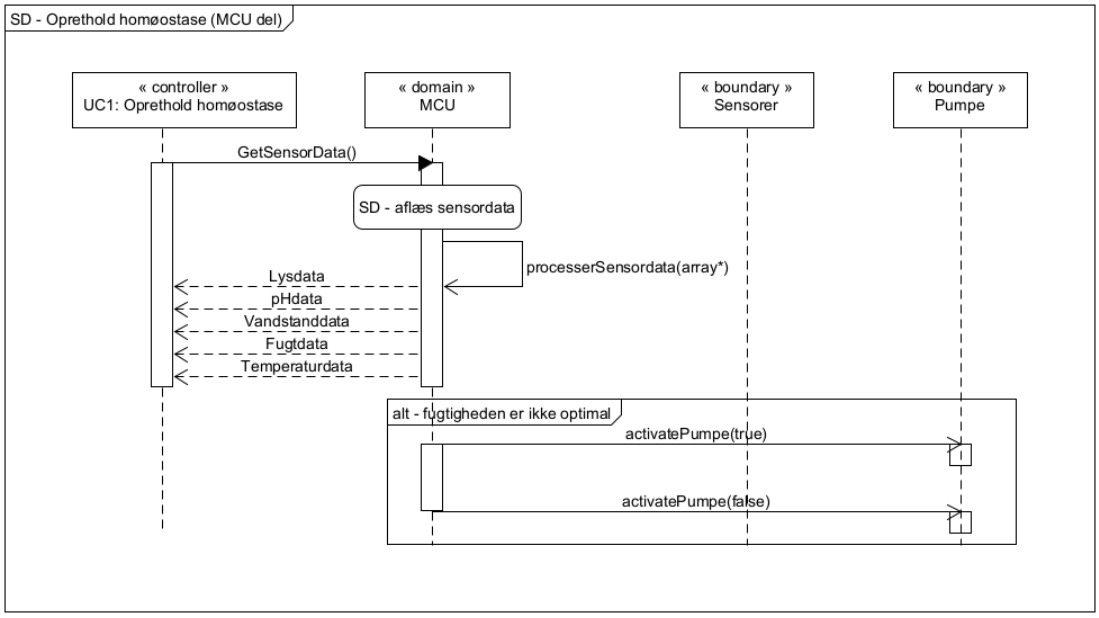
Her begynder arbejdet med vores applikationsmodel. Første beslutning var at benytte Entity-control-boundary software pattern løseligt i vores modellering. Da vores brugergrænseflade indgår i alle tre Use Cases, har vi modeleret med udgangspunkt i denne. Vi har valgt at bruge sekvensdiagrammer, da de ligesom vores Use Case beskrivelser gennemgår et forløb i tid:



Figur 7: Sekvensdiagram for Use Case 1

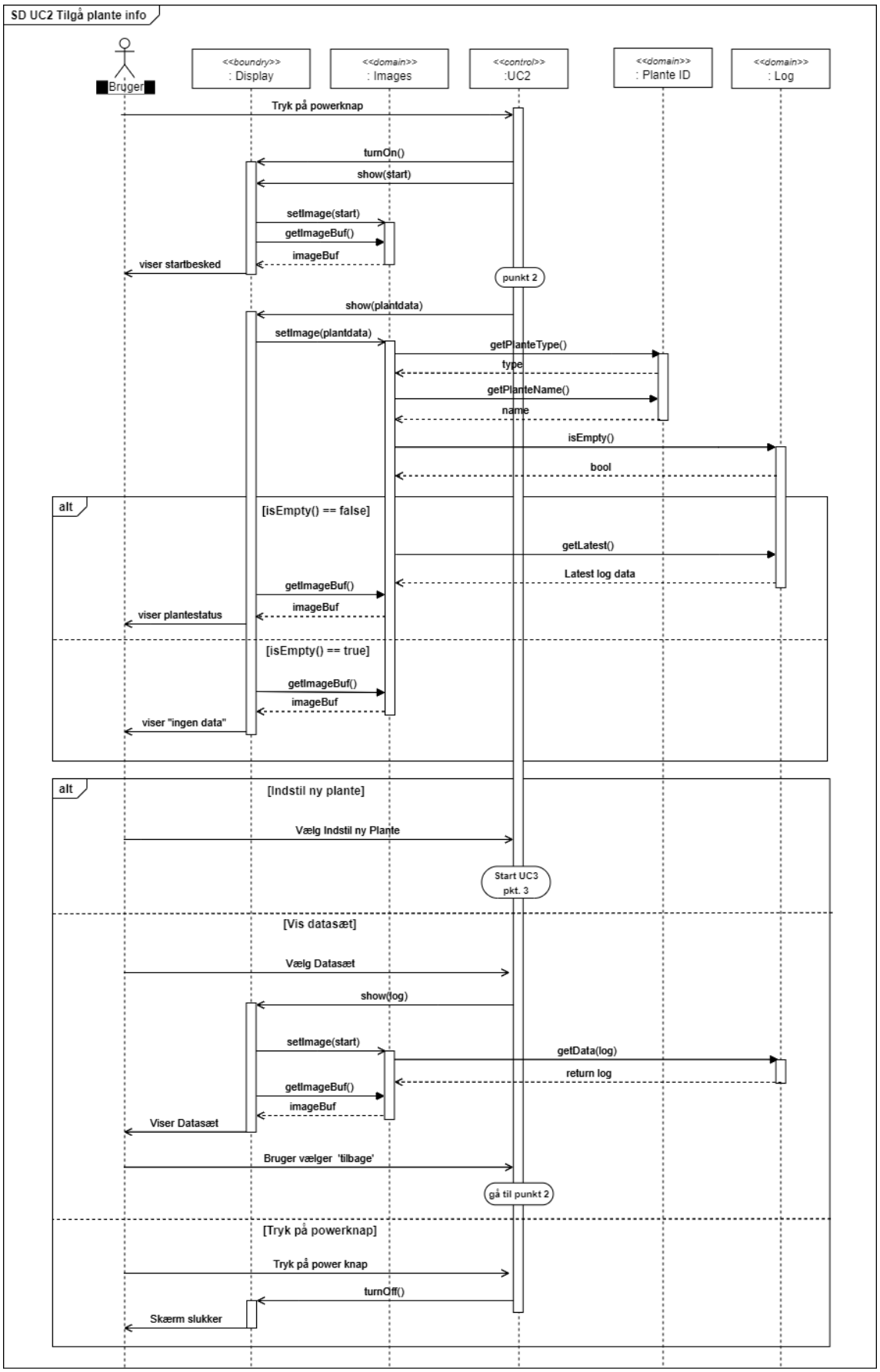
Use Case 1 er det eneste hvor systemet selv skal initiere. Som det ses har vi valgt at det er brugergrænsefladen der står for dette. Det er gjort for at holde alle tre cases så ens som muligt; brugergrænsefladen er også hvor brugeren af systemet starter de andre Use Cases. Samtidig vil vi gerne samle kompleksiteten på vores RPI - det er den vi har bedst kendskab til, og som kan afvikle C++ kode - det stiller os friere og med bedre ressourcer, end alternativet (PSoC'en, som ikke umiddelbart understøtter C++).

Use Case 2 og 3 er svært overskueligt. Derfor har vi valgt at modellere dem først ved hjælp af et State Machine diagram. Et "sted"hvor brugeren befinder sig i menuen svarer herunder til en given state:

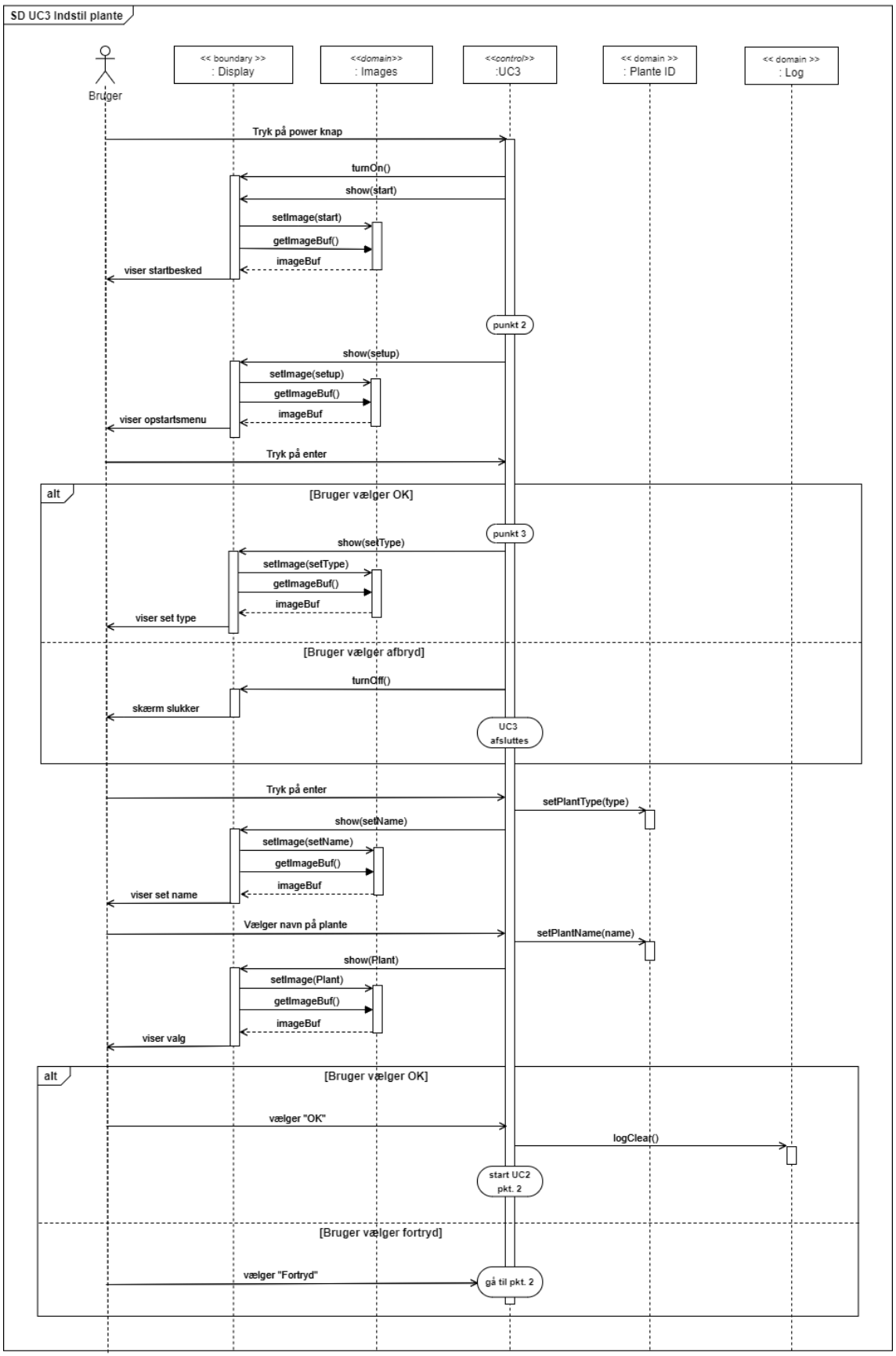


Figur 8: State machine diagram for brugergrænseflade menu

Med dette som guide, kan de mere komplekse sekvensdiagrammer laves:



Figur 9: Sekvensdiagram for Use Case 2

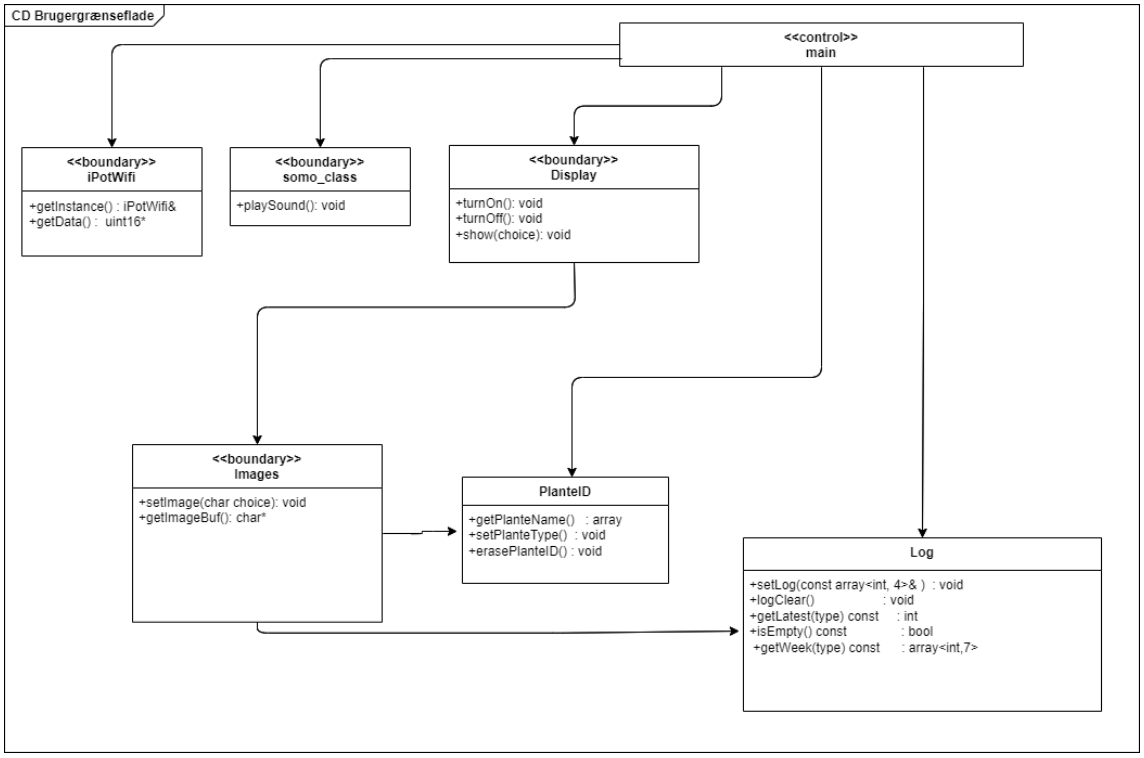


Figur 10: Sekvensdiagram for Use Case 3

Figur 9 og 10 herover er ikke videre anvendelig til at skabe et overblik, men i arbejdet med skabe dem, har vi sikret at vi er igennem alle de tilfælde hvor ét modul har brug for et andet.

Herudfra kan vi skrive vores klassediagrammer (se afsnittet Modul- og kommunikationsdesign). Diagrammerne fungerer også som opslagsværk, hvor den ansvarlig for ét modul kan få en forståelse for den sammenhæng det indgår i med andre, inklusiv formålene med diverse metodekald.

Et samlet, men ukomplet, overblik fås i vores overordnede klassediagram for brugergrænsefladen:

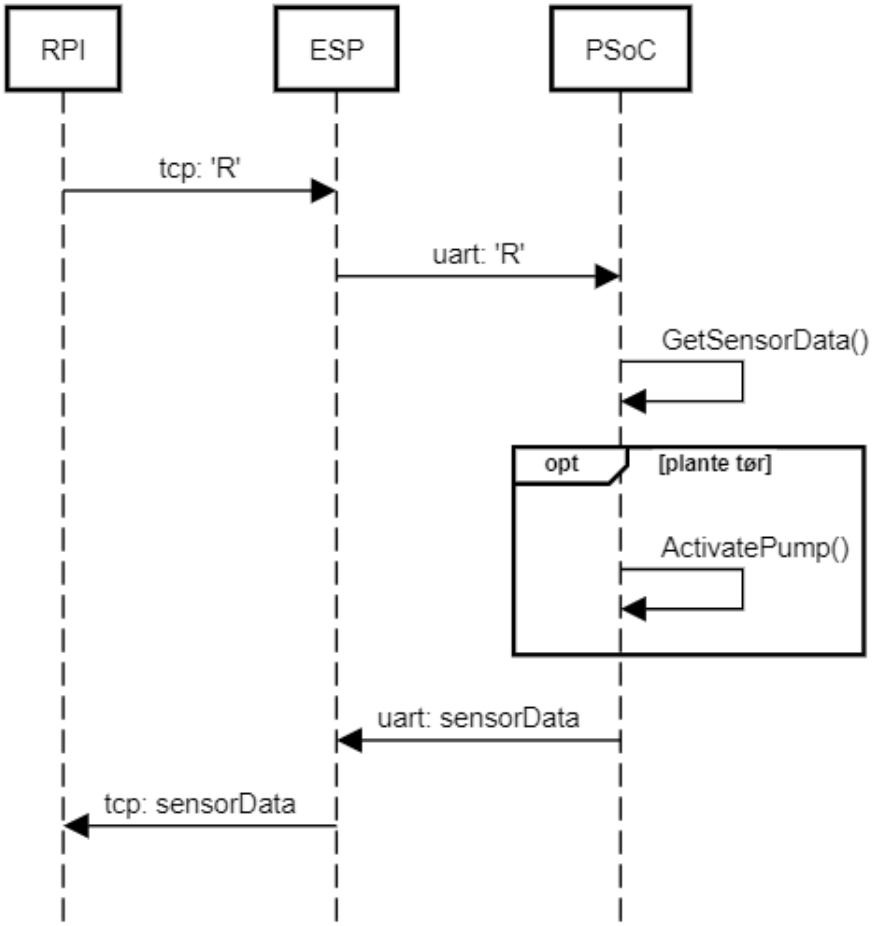


Figur 11: Klassediagram for brugergrænseflade

Som det ses, har vi en klassisk main control klasse, som "serviceres"af en række moduler: somo_class og iPotWifi er simple i deres interaktion, men skjuler kompleksitet omkring et hardware-lag. planteid og log har flere, mere kompleks interaktioner med main, men forholder sig ikke til noget hardware. Kun display (som her også dækker over de dertil hørende knapper) gør begge dele, og er klart vores mest komplekse klasse. Ud fra ovenstående har vi kunnet uddele modulernes design og implementering til gruppens software-medlemmer.

4.2 Potte og kommunikation

I arkitekturen af vores potte, behøver vi kun forholde os til Use Case 1, da det er den eneste som ikke foregår udelukkende på brugergrænsefladen. Til gengæld er det nødvendigt også at forholde sig til det samlede system. Tilsvarende Figur 7 har vi udarbejdet et sekvensdiagram for interaktionen:



Figur 12: Sekvensdiagram for Use Case 1 - kommunikationsdel

Et meget simpelt design, men det indeholder allerede nogle valg:

- **UART mellem PSoC og ESP**: Simpelt og for os velkendt kommunikationsprotokol, som PSoC'ens indbyggede funktioner understøtter direkte.
- **WiFi mellem RPI og ESP**: WiFi passer til vores brugs-situation, som beskrevet i vores Use Cases. Vores bruger antages at bo i et typisk hjem, og skal ikke være begrænset af hvor i hjemmet de to enheder kommunikerer. WiFi er bygget til netop en sådan situation. Bluetooth havde været et muligt alternativ, men da vores RPI understøtter WiFi uden yderligere hardware er dette nemmere. Samtidig er WiFi fremtidssikret, ift. hvis vi ville udvide med app-understøttelse eller en tids-server til vores log-funktioner.
- **TCP mellem RPI og ESP**: TCP har den fordel over UDP at den sikrer levering; ved at bruge denne protokol behøver vi ikke skrive kode der sender diverse ACK beskeder frem og tilbage, for at sikre at en besked er leveret.
- **sensorData**: Fra hardware arkitekturen ved vi at sensoren med det "breddeste"output leverer en 16 bit. Derfor vælges et uint16 array til sensorData; den kan holde den fulde rå data, og dermed begrænses designet ikke.

Udover ovenstående er designet af softwaren ikke begrænset af arkitekturen.