

Rapport  
AutoGreen  
Gruppe 9

3. Semesterprojekt E3PRJ3-02  
Ingeniørhøjskolen, Aarhus Universitet  
Vejleder: Tore Arne Skogberg

27. maj 2015

Navn	Studienummer	Underskrift
Morten Hasseris Gormsen	201370948	
Kristian Thomsen	201311478	
Philip Krogh-Pedersen	201311473	
Lasse Barner Sivertsen	201371048	
Henrik Bagger Jensen	201304157	
David Erik Jensen	11229	
Kasper Torp Samuelsen	201311498	
Kristian Søgaard Sørensen	20115255	

## Abstract

This documentation contains the development of a prototype, which is meant to be installed in a greenhouse. The system, AutoGreen, measures the temperature in the greenhouse and regulates it, by opening/closing windows, using ventilation and a heater. The system is also able to measure soil humidity, with its six soil humidity-sensors, and with the data, able to give the user a notice if it is time to water the plants.

During the first phases of the project, it was set to have air humidity sensors and light sensors in the greenhouse, logging and graphical preview of the logged data, a database with standard and customized plants, an e-mail system to inform the user of important event in the greenhouse, and much more. This was not implemented in the prototype.

During the development of the product, a miniature greenhouse was used. If AutoGreen was to be installed in a real greenhouse, the ventilators, heater and window motor should be scaled up to fit the size of the greenhouse.

AutoGreen's user interface and controller are developed on an Embest Devkit8000 Evaluation board [1]. The Devkit8000 communicates with a I<sup>2</sup>C master, made on a PSoC 4 Pioneer Kit [2], using UART communication. The master unit communicates with its two I<sup>2</sup>C slaves. One of the slaves being the "Actuator" unit, and the other slave being the unit for measuring soil-humidity. The slaves are developed on a PSoC 4 Pioneer Kit. The temperature is measured using a LM75 with I<sup>2</sup>C interface [6].

The system can measure temperatures within the greenhouse with a precision of +/- 0.5 degrees. It is able to regulate the temperature with a precision of +/- 1 degree, up to 10 degrees above the surrounding temperature.

In case of water shortage at one of the soil humidity sensors, a message is displayed on the Graphical interface, and a port on one of the I<sup>2</sup>C slaves switches from low to high.

The soil humidity sensors adds an option to install an automatic watering system to the AutoGreen system. There are much more options to further develop the system. See chapter 1 Projektformulering (danish) on page 1 in the documentation for further information.

The biggest obstacle in the product is the UART communication. A simplified version of UART is used, which means only Tx, Rx and ground reference is used, there are a few miscommunications between the UART and the I<sup>2</sup>C master. The miscommunication never occurs when connecting a UART terminal directly to the I<sup>2</sup>C master. See chapter 8 Accepttest (danish) on page 163 in the documentation for further information on results.

## Resume

Denne rapport omhandler udviklingen af en prototype til et system, der kan installeres i et drivhus. Systemet - kaldet AutoGreen - måler lufttemperatur og regulerer denne i drivhuset ved hjælp af åbning og lukning af et vindue, ventilation og et varmelegeme. Systemet kan desuden måle jordfugtighed med op til seks jordfugtmålere, og give brugeren besked om, at det er tid til at vande ved en given jordfugtmåler.

Gennem de første faser af projektarbejdet er der desuden lagt op til måling af luftfugtighed og lysintensitet i drivhuset, logning og grafisk præsentation af måledata, database med planteinformationer, e-mailnotifikationer med mere. Dette er dog ikke implementeret.

Under udviklingen af prototypen er der anvendt en model af et drivhus på ca. 33 liter. Såfremt AutoGreen skulle anvendes i et rigtigt drivhus, skulle aktuatorer - dvs. ventilatorer, varmelegeme og vinduesmotor - skaleres derefter.

AutoGreen's brugerflade og controller er realiseret på et Embest DevKit8000 Evaluation board [1]. DevKit8000 kommunikerer vha. UART med en I<sup>2</sup>C master, der er realiseret på et PSoC 4 Pioneer Kit [2]. Masterenheden kommunikerer med flere I<sup>2</sup>C slaver, der er koblet til hhv. aktuatorer og analoge sensorer. To af disse disse I<sup>2</sup>C slaver er ligeledes realiseret på et PSoC 4 Pioneer Kit. Måling af temperatur i drivhuset sker vha. en sensor - LM75 - med I<sup>2</sup>C interface [6].

Det realiserede system kan måle temperaturen i drivhuset med en præcision på +/- 0.5 °C. Systemet kan - i området op til 10 °C over den omgivende temperatur - regulere temperaturen med en præcision på +/- 1 grad. Måling af jordfugt fungerer ligeledes på det realiserede system. I tilfælde af manglende vand ved en sensor, gives der besked om dette på brugerfladen, og en port på en af systemets I<sup>2</sup>C slaver går fra logisk lav til logisk høj.

Dette åbner op for muligheden for tilkobling af et vandingssystem, men der er som nævnt flere muligheder for videreudvikling og udbygning af systemet. Se afsnit 1 Projektformulering på side 1 i projektdokumentationen for yderligere information om systemet.

Den største udfordring i det realiserede system ligger i UART kommunikationen mellem DevKit8000 og I<sup>2</sup>C master. Der anvendes en simplificeret UART kommunikation - kun med Tx, Rx og reference - hvorpå der er en del fejlkommunikation. Dette opleves ikke, hvis man fx. kobler en UART terminal direkte på I<sup>2</sup>C masteren. Se afsnit 8 Accepttest på side 163 i projektdokumentationen for yderligere information om resultatet.

# Indhold

<b>Abstract</b>	ii
<b>Resume</b>	iii
<b>Indhold</b>	iv
<b>Arbejdsopgaver</b>	vi
<b>1 Forord</b>	1
<b>2 Indledning</b>	2
2.1 Læsevejledning .....	2
2.2 Ordforklaring .....	3
<b>3 Opgaveformulering</b>	4
<b>4 Projektafgrænsning</b>	5
<b>5 Systembeskrivelse</b>	6
<b>6 Krav</b>	8
<b>7 Projektbeskrivelse</b>	10
7.1 Projektgennemførelse.....	10
7.2 Metoder .....	11
7.2.1 V-Model og ASE-Model .....	11
7.2.2 SysML .....	11
7.2.3 Scrum .....	11
7.2.4 Versionsstyring .....	12
7.2.5 Reviews .....	12
7.3 Specifikation og Analyse .....	13
7.4 Systemarkitektur .....	14
7.5 Hardware Design .....	18
7.5.1 PSoC Master Design .....	18
7.5.2 Slave Aktuator Design .....	22
7.5.3 Slave Jordfugt Design .....	24
7.5.4 Strømforsyning Design .....	25
7.6 Software Design.....	26
7.6.1 Sekvensdiagrammer .....	26
7.6.2 Klassebeskrivelser .....	27
7.6.3 Datastruktur .....	27
7.6.4 Datatransmission .....	28
7.6.5 Intertrådskommunikation .....	28
7.7 Hardware Implementering .....	29
7.7.1 PSoC Master Implementering .....	29
7.7.2 Slave Aktuator Implementering .....	30
7.7.3 Slave Jordfugt Implementering .....	31
7.7.4 Strømforsyning Implementering .....	31

7.8	Software Implementering.....	32
7.8.1	QT Integration .....	32
7.8.2	Timers på DevKit8000 .....	32
7.8.3	UART Implementering.....	32
7.8.4	Regulering .....	33
7.9	Resultater og Diskussion .....	34
7.10	Udviklingsværktøjer .....	35
7.10.1	PSoC Creator.....	35
7.10.2	Multisim .....	35
7.10.3	Ultiboard.....	35
7.10.4	QT Creator .....	35
7.10.5	Git .....	35
7.11	Opnåede Erfaringer.....	36
7.11.1	David .....	36
7.11.2	Henrik .....	36
7.11.3	Kasper.....	36
7.11.4	Kristian S. .....	36
7.11.5	Kristian T.....	37
7.11.6	Lasse .....	37
7.11.7	Morten .....	37
7.11.8	Philip .....	37
7.12	Fremitidigt Arbejde .....	38
<b>8</b>	<b>Konklusion</b>	<b>39</b>
<b>Litteraturliste</b>		<b>40</b>

## Arbejdsopgaver

Under projektarbejdet har arbejdsopgaver været fordelt efter følgende tabel:

	Kristian Søgaard Sørensen	Kasper Torp Samuelsen	David Erik Jensen	Henrik Bagger Jensen	Lasse Barner Sivertsen	Philip Krogh-Pedersen	Kristian Thomsen	Morten H. Gormsen
Projektformulering	X	X	X	X	X	X	X	X
Kravspecifikation	X	X	X	X	X	X	X	X
Systemarkitektur	X	X	X	X	X	X	X	X
HW Design og impl. - I <sup>2</sup> C Protokol				X	X	X	X	X
HW Design og impl. - PSoC Master					X	X	X	X
HW Design og impl. - Slave Aktuator					X			X
HW Design og impl. - Slave Jordfugt					X			X
HW Design og impl. - Strømforsyning					X			X
SW Design	X	X	X					
SW Impl. - Build Tools				X				
SW Impl. - Doubly Linked List				X				
SW Impl. - UART			X					
SW Impl. - Indstillinger				X				
SW Impl. - Monitor	X							
SW Impl. - SystemLog	X							
SW Impl. - Regulator			X					
SW Impl. - Brugerflade	X	X	X					
Accepttest	X	X	X	X	X	X	X	X
Rapport	X	X	X	X	X	X	X	X

## 1 Forord

Da dette projektarbejde skulle påbegyndes, blev der foreslået flere forskellige projekter. Et system, der styrede adgangskontrol til bygninger og lignende var på tale, og der var idéer vedrørende en form for dispensersystem.

Ud af de systemer, der blev overvejet, var der et vandsystem, som var tæt på at blive valgt. Dette system omhandlede mekanisk filtrering, frostsikring, kemikaliehandling mm. af et specifikt drikkevandssystem som findes på Grønland. AutoGreen systemet minder overordnet set meget om et sådant system; det indeholder en række aktuatorer og sensorer, og styringen af disse kunne overordnet set foretages på præcis samme måde.

Valget stod derved mellem to ret ens projekter. AutoGreen systemet blev valgt frem for vandsystemet, da gruppen forudså at der ville blive mindre arbejde med de fysiske rammer. Der var en model af et drivhus tilgængelig fra et tidligere projekt, og arbejdet med luft frem for vand syntes væsentligt nemmere.

## 2 Indledning

Denne rapport omhandler udvikling og realisering af en prototype til et system, der kan installeres i et drivhus. Systemet - AutoGreen - hjælper brugeren med at opnå og fastholde optimale forhold for planterne i drivhuset. Systemets vigtigste funktioner er måling og regulering af temperatur, samt måling af jordfugt. Reguleringen af temperatur sker ved hjælp af et varmelegeme, ventilatorer og åbning/lukning af et vindue. Ved manglende fugtighed i jorden gives brugeren besked herom.

AutoGreen er både for den uerfarne bruger, der har brug for hjælp for at sikre sine drivhusplanters overlevelse, men det er også for den mere erfarne bruger, der ønsker optimerede forhold i sit drivhus. Opvarmning af drivhuset medvirker til forlængelse af vækstsæsonen og rettidig vanding af planterne er med til at sikre optimale vækstforhold.

Controlleren og brugerfladen i AutoGreen - realiseret på et DevKit8000 - kommunikerer via UART med et PSoC 4 Pioneer Kit, der agerer I<sup>2</sup>C master i systemet. Masteren kommunikerer flere I<sup>2</sup>C slaver, der har ansvar for hhv. aktuatorer (varme, vindue og ventilation), måling af temperatur og analoge jordfugtsensorer.

### 2.1 Læsevejledning

Rapporten er, så vel som projektdokumentationen, opbygget kronologisk, dvs. efter samme rækkefølge som arbejdet er udført. Der er dog den undtagelse at accepttestspezifikationen er udarbejdet i umiddelbar forlængelse af kravspecifikationen, men selve testen er naturligvis først gennemført i slutningen af forløbet, hvorfor dette afsnit er behandlet i slutningen af både rapport og projektdokumentation.

Bemærk at afsnittet Arbejdsopgaver (side vi) indeholder en oversigt over hvad de enkelte gruppemedlemmer har haft ansvar for - ikke nødvendigvis hvilke afsnit af rapporten (eller dokumentationen) de har skrevet.

For dokumentationen gælder det, at hvert kapitel har en versionshistorik, hvor der ved indtastning er påført et gruppemedlems initialer. Dette betyder udelukkende at det pågældende gruppemedlem har 'siddet ved tasterne', og giver således ikke gruppemedlemmet nogen form for ansvar eller ejerskab af kapitlet.

Ved første øjekast kan denne rapport synes væsentlig længere end de tilladte 30 normalsider. Se [11] for detaljeret disposition.

## 2.2 Ordforklaring

Begreb	Forklaring
Plantedatabase	Plantedatabasen indeholder information om ideelle forhold for forskellige typer planter, som brugeren kunne tænkes at plante i sit fysiske drivhus. Informationen i plantedatabasen står til grund for udgangsparametre for nye planter i det virtuelle drivhus. Der findes en række systemplanter, som brugeren ikke kan redigere eller slette, men brugeren kan tilføje egne planter.
Datalog	Systemet er udstyret med en log over de indsamlede data fra sensorer i systemet, og der måles og indskrives i loggen hvert minut. Denne er opbygget som en datastruktur, hvor hver logning indeholder information fra sensorerne samt et tidspunkt.
Systemlog	Systemet er udstyret med en log over hvad systemet foretager sig. Dette kunne f.eks. være et indlæg når systemet foretager en mæling, sender en e-mail og regulerer miljøet i drivhuset.
Virtuelt Drivhus	Det virtuelle drivhus er systemets repræsentation af det fysiske drivhus. Brugeren kan tilføje planter fra plantedatabasen i det virtuelle drivhus, og på den måde give systemet indirekte oplysninger om ønskede parametre. Disse informationer lagres i systemets konfigurationsfil.
Fysisk Drivhus	Ved det fysiske drivhus forstås det drivhus, hvori systemet er monteret.
Konfigurationsfil	Dette er en klasse, der er placeret på DevKit8000, som indeholder brugerens konfigurationer om blandt andet notifikationer, e-mailadresser, antallet af fugtsensorer og deres unikke ID mm.
Notifikations e-mail	Dette er en daglig e-mail, som brugeren kan vælge at få tilsendt. Den sendes klokken 12:00, og indeholder informationer om parametrene i det fysiske drivhus.
Advarsels e-mail	Dette er en e-mail, som brugeren kan vælge at få tilsendt. Den sendes, hvis en parameter i det fysiske drivhus kommer uden for tolerancen af den ønskede værdi.

### 3 Opgaveformulering

Herunder er vist en prioritering af funktionaliteter i systemet efter MoSCoW metoden.

Ambitionen for dette projekt var som absolut minimum, at realisere nedenstående punkter under "*skal*". Det forventedes desuden at punkterne under "*bør*" skulle realiseres, men de har haft lavere prioritet. Punkterne under "*kan*" forventedes ikke realiseret, og punkterne under "*vil ikke...*" realiseredes med sikkerhed ikke. Sidstnævnte punkter kan ses som udviklingsmuligheder i forhold til senere versioner af systemet.

- **Systemet skal:**

- Kunne monitorere temperaturen i drivhuset og regulere temperaturen i drivhuset vha. varmelegeme, åbning af vinduer og luftcirculation.
- Give brugeren mulighed for at vælge varmelegeme og/eller luftcirculation fra, hvis en mere økonomisk regulering af temperaturen ønskes.
- Have et grafisk user interface.

- **Systemet bør:**

- Måle jordfugtighed med op til seks sensorer i drivhuset og give brugeren besked på brugerfladen om, at det er tid til at vande.
- Måle lysintensitet og luftfugtighed i drivhuset.
- Indeholde en log over alle målte parametre: Jordfugtighed, temperatur, luftfugtighed og lysintensitet. Dataene præsenteres grafisk for brugeren.
- Indeholde en database over de mest almindelige drivhusplanter, så brugeren kan orientere sig om en plantes optimale forhold.
- Indeholde en systemlog, som noterer vigtige system hændelser.

- **Systemet kan:**

- Sende besked til brugeren via e-mail, om at det er tid til at vande.
- Tilkobles et automatisk vandingssystem, som aktiveres ved behov for vanding.
- Give brugeren mulighed for at tilføje planter i databasen.
- Give brugeren mulighed for at kommunikere trådløst med systemet fra brugerfladen, så denne kan placeres fx inde i brugerens bolig.

- **Systemet vil ikke i denne version:**

- Indeholde et kamera, og tilhørende billedarkiv, som giver brugeren mulighed for at følge planternes udvikling fra dag til dag.
- Give brugeren mulighed for at interagere med systemet via en app på dennes mobiltelefon.

For den fulde tekst se afsnit 1 Projektformulering på side 1 i projektdokumentationen.

## 4 Projektafgrænsning

AutoGreen består af en strømforsyning, en række controller (tre stk. PSoC 4 Pioneer Kits og et stk. DevKit8000), et varmelegeme (en USB strømspareskinne og en 100W 230V AC glødepære), fire 12V DC ventilatorer og en 12V steppermotor - samt tilhørende mosfet drivere. AutoGreen uinneholder desuden sensorer til måling af lufttemperatur, jordfugt, lysintensitet og luftfugtighed.

Selve det fysiske drivhus er således ikke en del af systemet. Under udviklingen af denne prototype er anvendt en model af et drivhus på ca. 33 liter, se evt. Figur 2 på side 7 i Projektdokumentationen. Såfremt prototypen skal monteres i et rigtigt drivhus, skal ventilatorer og varmelegeme dimensioneres derefter.

I de indledende faser af projektarbejdet - Projektformulering, Kravspecifikation, Accepttestspezifikation og Systemarkitektur - omhandler projektdokumentationen det fulde system. Grundet tidsnød og forskellige komplikationer, er der flere dele af det samlede system, som kun er delvist eller slet ikke implementeret. Prioriteringen af hvad der skulle skæres væk undervejs har taget udgangspunkt i MoSCoW prioriteringen, se afsnit 3 Opgaveformulering på side 4.

Alle punkter under "Systemet skal" er fuldt implementeret.

Under "Systemet bør" er det første punkt, vedrørende jordfugtsensorer, fuldt implementeret; øvrige punkter er kun delvist eller slet ikke implementeret.

Punkter under "Systemet kan" og "Systemet vil ikke i denne version" er ikke implementeret.

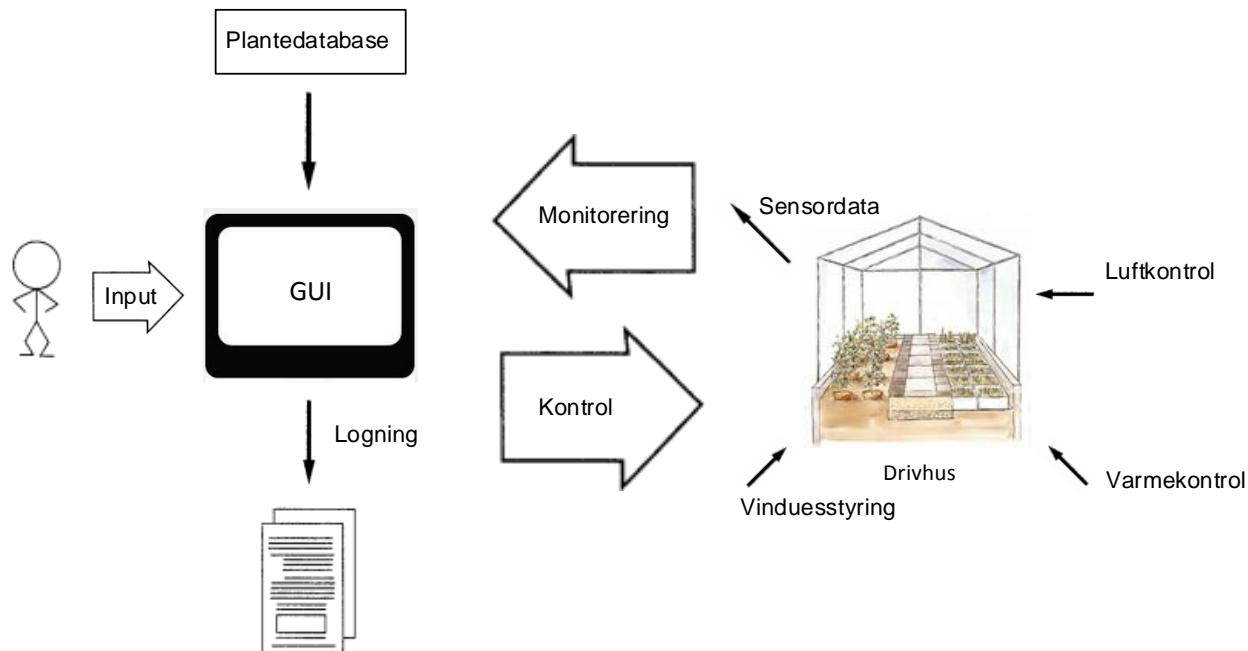
Det er med fuldt overlæg at denne løbende prioritering har fundet sted. Gruppen ønskede fra starten et system med mange muligheder for udvikling og udbygning; derfor blev der fra begyndelsen beskrevet et system, som var væsentligt mere omfattende, end hvad gruppen regnede med at kunne nå at realisere.

Den gennemførte accepttest (af hele det beskrevne system) giver detaljeret information om hvad der er realiseret, hvad der er delvist realiseret og hvad der ikke er realiseret, se afsnit 8 Accepttest på side 163 i projektdokumentationen.

## 5 Systembeskrivelse

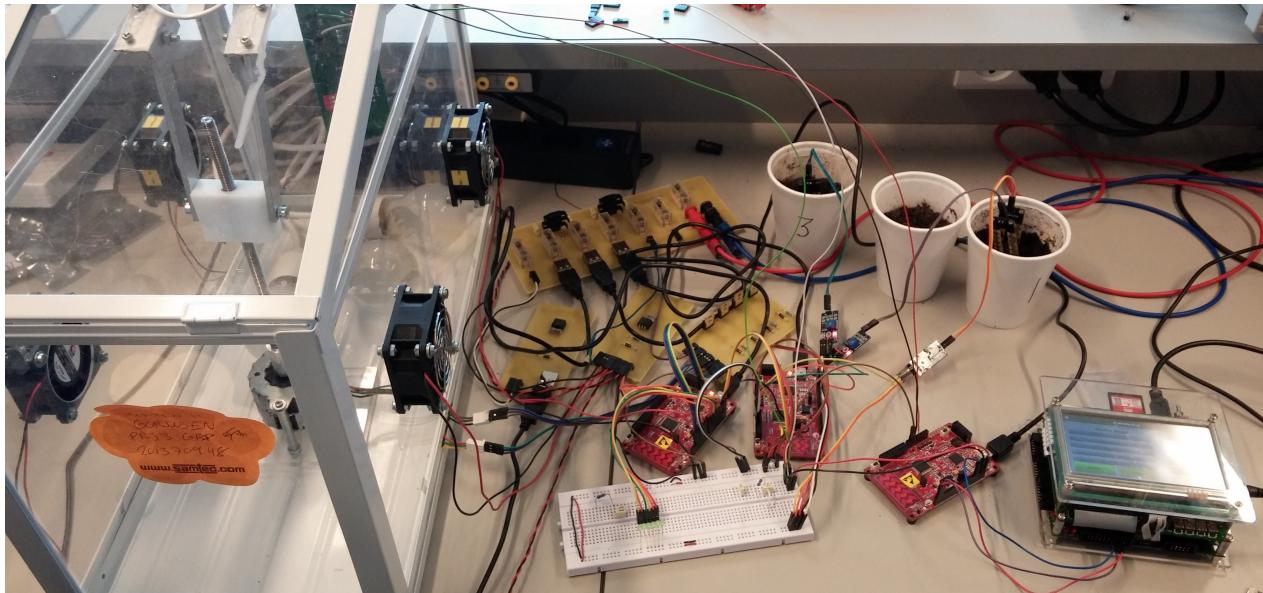
Det system, der er tænkt realiseret i dette projekt, er en skalamodel af et drivhus med steppermotor til åbning af et vindue og blæsere til udluftning samt et varmelegeme til at varme drivhuset op. Til at måle på drivhuset var det tiltænkt at implementere en temperatursensor, jordfugtmålere samt sensorer til måling af lysintensitet og luftfugtighed. De to sidstnævnte er dog ikke implementeret grundet komplikationer i implementeringsfasen.

Selv systemet styres af et DevKit8000, som brugeren af systemet kan interagere med. På denne platform kører - samtidigt med det grafiske miljø - processer til regulering og monitorering af miljøet i drivhuset. Der er udover dette også mulighed for at tilgå forudindstillede planter i en plantedatabase samt at tilføje og fjerne eksisterende planter i et virtuelt drivhus, som systemet anvender til at afspejle de planter, der eksisterer i det fysiske drivhus. Al aktivitet omkring styringen af systemet logges i en systemlog.



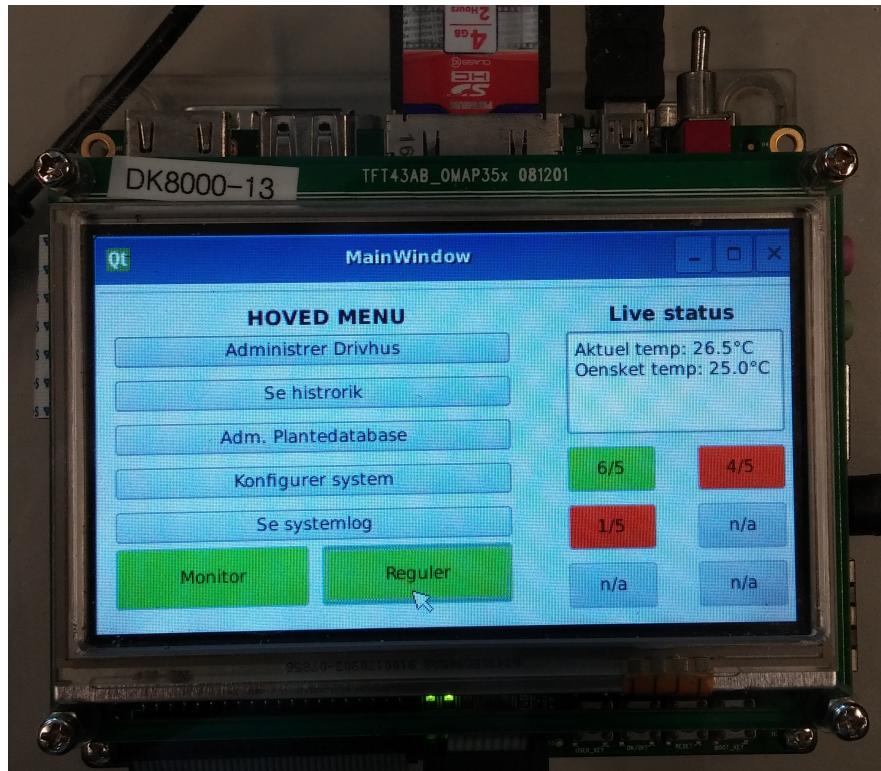
Figur 1: Rigt billede af systemet

Figur 1 viser et rigt billede af systemet; der ses hvordan det fysiske drivhus påvirkes ved kontrol af varme og luft samt kommunikationen med GUI'en.



Figur 2: Billede af det færdige system

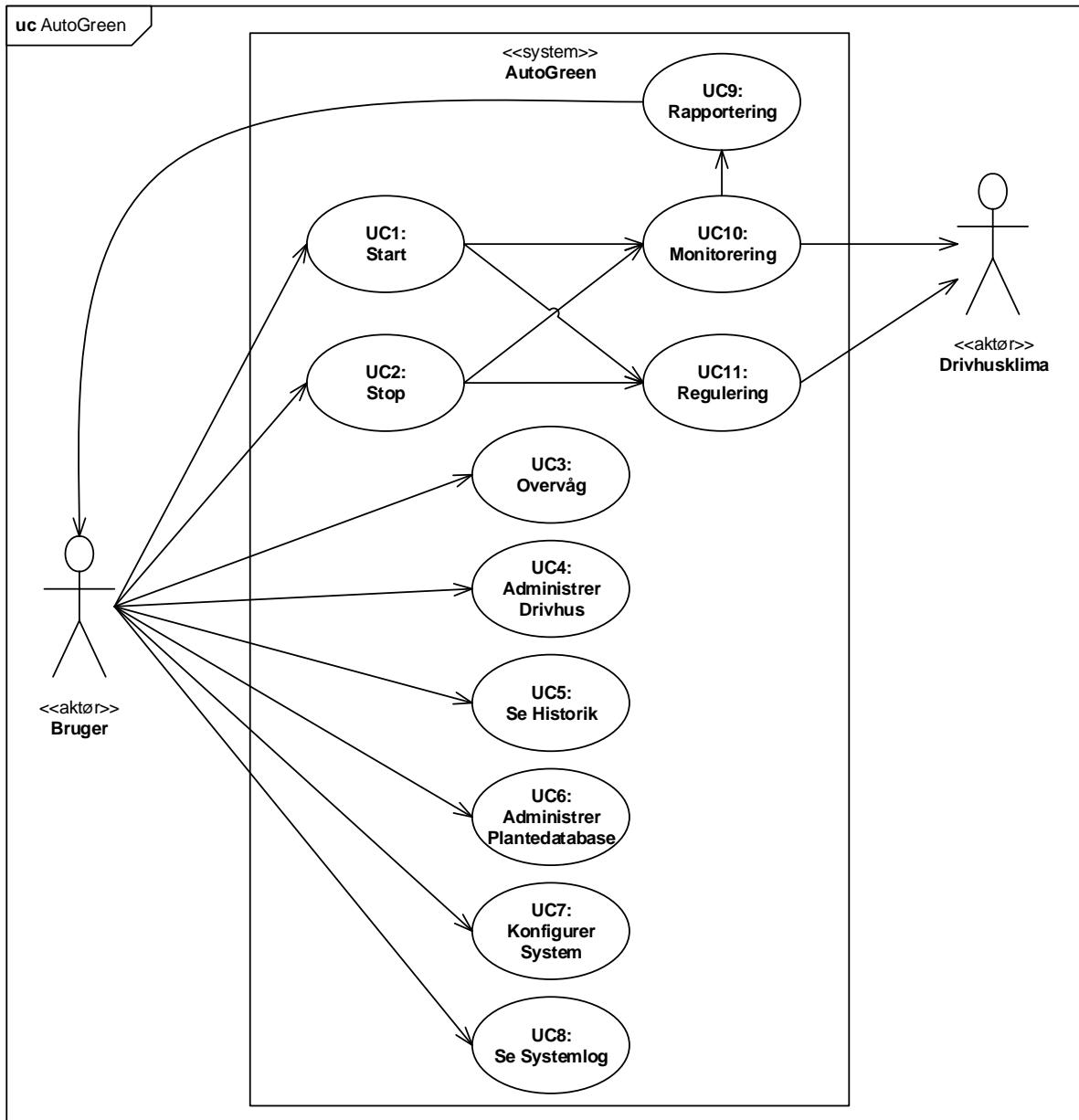
På Figur 2 ses et billede af den færdige prototype, og på Figur 3 ses et billede af hovedmenuen på systemets brugerflade.



Figur 3: Billede af brugerfladen

## 6 Krav

I dette afsnit beskrives optillede krav for AutoGreen, som er opstillet ud fra opgaveformuleringen. På Figur 4 ses Use Case diagram over systemet. Dette giver et overblik over de funktionelle krav, der er formuleret i dokumentationen på side 5.



Figur 4: Use Case Diagram for AutoGreen

Use Cases på billedet er kort beskrevet herunder:

- UC1: Start

Denne UC giver brugeren mulighed for at starte systemet, dvs. monitorering og regulering af drivhusklimaet.

- UC2: Stop  
Denne UC giver brugeren mulighed for at stoppe systemet, dvs. monitorering og regulering af drivhusklimaet.
- UC3: Overvåg  
Når UC10 Monitorering er startet, vises der på brugerfladens hovedmenu alle de aktuelle måleværdier. Hvis UC11 Regulering er startet, kan værdierne for lufttemperatur og jordfugtighed være røde, hvis de ikke passer med de ønskede værdier.
- UC4: Administrer Drivhus  
Giver brugeren mulighed for at informere systemet om hvilke planter der er i drivhuset.
- UC5: Se Historik  
Giver brugeren mulighed for at se en grafisk historik over de fire målte parametre i drivhuset.
- UC6: Administrer Plantedatabase  
Giver brugeren mulighed for at se på planter i databasen, samt tilføje og fjerne egne planter i databasen.
- UC7: Konfigurer System  
Giver brugeren mulighed for at rette i systemindstillinger.
- UC8: Se Systemlog  
Giver brugeren mulighed for at se en liste over systemhændelser.
- UC9: Rapportering  
Rapporterer til brugeren ud fra de indstillinger brugeren har valgt. Dette sker ved afsendelse af e-mail til den eller de adresser som er valgt.
- UC10: Konfigurer System  
Lagrer målinger af lufttemperatur, jordfugtighed, luftfugtighed og lysintensitet i en data log.
- UC11: Regulering  
Regulerer temperaturen i drivhuset, vha. vinduesåbner, varmelegeme og luftcirculation, med mindre brugeren har slået varmelegeme og/eller luftcirculation fra.

Systemet skal have en grafisk brugerflade, der giver brugeren mulighed for at konfigurere og monitere drivhuset. På brugerfladen skal brugeren have mulighed for at overvåge den aktuelle temperatur og bør desuden kunne se jordfugt, luftfugtighed og lysintensitet. Disse data logges og bør kunne aflæses på en graf, der viser historik for samtlige parametre. Systemet skal kunne regulere temperaturen i drivhuset på baggrund af de aktuelle parametre.

Baseret på brugeres præferencer kan systemet advare brugeren via e-mail, hvis systemet fejler eller klimaforholdene bliver kritiske. Til at regulere systemet skal brugeren kunne indstille systemet til at anvende varmelegeme og/eller ventilatorer til at justere klimaet.

AutoGreen bør have en plantedatabase, der indeholder foruddefinerede planter. Planterne kan indsættes i det virtuelle drivhus, og herefter skal systemet kunne regulere klimaet i det fysiske drivhus, så passer bedst til de(n) valgte plante(r). Brugeren skal kunne tilføje, fjerne og redigere planter, som er indsat i det virtuelle drivhus efter behov.

## 7 Projektbeskrivelse

### 7.1 Projektgennemførelse

Gruppen, som er en videreførelse fra 2. semesterprojekt, har løftet opgaven med fornyet engagement og endnu større handlekraft end tidligere. Fra begyndelsen af projektperioden har arbejdet med projektet været relativt uden problemer. Det har hele tiden været gruppens mål, som på foregående semester, at holde sig foran tidsplanen [10], men samtidig have en fornuftig tilgang til arbejdet. Dette har for gruppen betydet en forøget arbejdsindsats i form af anvendelsen af alle fritimer, der var mulige at bruge. Forskellen på dette semesterprojekt og gruppens tidligere, er en meget mere klar opdeling i hardware og software. Opdelingen er kommet som en naturlig konsekvens af opdelingen i uddannelserne E/EP/IKT, og har samtidig betydet øget fokus på de relevante dele af projektet for de individuelle medlemmer.

I projektet er anvendt en kombination af udviklingsmodellerne V-model, Scrum og ASE-modellen, som i en stor blanding, gav muligheden for udarbejdelsen af gruppens projektdokumentation og rapport. Se afsnit 7.2 Metoder på side 11 for nærmere beskrivelse. Modellerne er ikke nødvendigvis fulgt fuldstændigt, men gruppen har efterhånden udarbejdet sin egen fortolkning, som er meget velfungerende. Gennem projektperioden er hvert overemne blevet kørt som et sprint, men det er først i forbindelse med design og implementering at der konkret kan tales om reelle sprint. Eftersom gruppen har valgt at fortsætte fra et tidligere semester, er mange af tingene som blev udarbejdet tidligere, fx samarbejdsaftale, mødeskabeloner og opgaver blevet genbrugt. Genanvendelsen af delelementer har gjort opstarten af projektet en del nemmere, end hvis der skulle startes fra bunden. Rollerne der er blevet fordelt i projektet ser ud som følger:

- Koordinator

Morten har haft det overordnede ansvar for administrative opgaver, som mødeindkaldelser, referater og logførelse.

- Ordstyrer

Philip har været ordstyrer igennem gruppens vejleder- samt arbejdsmøder.

- Dropbox Ansvarlig

Kristian T. har stået for orden og udlægning af deletjenesten.

- GitHub ansvarlig

David har været ansvarlig for kildekodedelingen over GitHub.

- SCRUM ansvarlig

David har været overordnet ansvarlig for anvendelsen af SCRUM.

- Lokale booking

Kristian S. har været ansvarlig for at booke lokaler når det var nødvendigt.

## 7.2 Metoder

Under Metoder vil de forskellige arbejdsmetoder, der er blevet brugt under dette projekt blive beskrevet. Disse metoder er hhv. V-model, SysML, Scrum, Reviews og Versionsstyring.

### 7.2.1 V-Model og ASE-Model

Under projektets forløb er V-Modellen fulgt, som en vejledning til udførelsen af projektet. Modellen er dog ikke fulgt fuldstændigt, da der ikke er blevet defineret flere testscenarier ud over den vigtige Accepttest. Dette skyldes, at gruppen har fundet det mere hensigtsmæssigt at lave løbende tests, da der i mange tilfælde har været en vigtig læringsproces i hvilke funktionaliteter, der har kunne lade sig gøre. Derved har det været svært at fastsætte mindre tests imellem de forskellige enheder i tidlige stadier. Det vil sige at tests såsom modultests blev beskrevet og bearbejdet sideløbende med design- og implementeringsfasen.

Ud over V-Modellen ??, er ASE-Modellen ?? taget i brug som en vejledning til udførelse af projektet. Der er hovedsageligt lagt fokus på at gøre det muligt for HW- og SW-grupper at dele sig op under design og implementering. På baggrund af dette er der lagt stor fokus på at forklare systemets ønskede funktionalitet og kommunikationsveje under systemarkitektur. Opdelingen har gjort arbejdet mere effektivt, da det har givet den enkelte mulighed for mere fordybelse til at arbejde med et specifikt område.

### 7.2.2 SysML

SysML har været medvirkende til at give overblik over projektet, da systemet har kunnet deles op i blokke, og det herefter var muligt at arbejde med disse individuelt. Ud fra disse blokke var beskrivelsen af parts og ports nemt. BDD-diagrammer har givet overblik over komposition af blokkenes relationer, som er specificeret i diagrammet. IBD-diagrammer har givet mulighed for at holde styr på signaler og kommunikationsveje mellem de forskellige blokke. Signalerne, der går imellem blokkene, gav mulighed for at lave detaljerede grænseflader på systemets elementer. UART protokollen er systemets vigtigste grænseflade, da den definerer grænsen mellem HW og SW gruppen. Use Cases har givet mulighed for at designe det ønskede scenarie, og tage højde for de faldgrupper, der kan opstå undervejs i scenariet. Use Cases er blevet anvendt til at fremstille sekvensdiagrammer, så de stemmer overens med udførelsen af de enkelte steps i use casen.

### 7.2.3 Scrum

Scrum er primært brugt af SW gruppen. Scrum blev brugt til uddeling af opgaver under design af SW til DevKit8000. En webbaseret løsning er brugt som scrum-board, i stedet for et fysisk scrum-board, da et fast grupperum ikke har været til rådighed. Der blev i SW gruppen brugt en form for daglige scrum-møder, hvor der hurtigt kunne gennemgås status på individuelle opgaver, og om der var forekommeth nogle problemer, der kunne være svære at løse. Herefter kunne scrum-boardet opdateres med evt. nye opgaver. Det gav et godt overblik, og alle havde altid adgang til at kunne se, hvad der kunne laves som det næste, når en opgave var løst. Det blev dog valgt i SW gruppen ikke at bruge scrum-boardet under implementeringsfasen, da der blev holdt daglige møder, og det ikke føltes som en nødvendighed at skulle holde styr på alle opgaverne vha. Scrum, når der kun var 3 personer i teamet. Der blev dog fortsat holdt fast på de daglige scrum-møder.

### 7.2.4 Versionsstyring

Versionerhistorik på dokumenter i projektdokumentationen er blevet opdateret løbende bla. i forbindelse med kommentarer fra vejleder og reviews. Væsentlige ændringer i fx design har givet anledning til versions-ændring, hvilket hjælper med at holde styr på hvilke ændringer projektet har gennemgået.

### 7.2.5 Reviews

De reviews der er modtaget igennem projektet, [8] og [9], har været en stor hjælp til retning og tilføjelser til dokumentationen. De afgivne reviews har været med til at give ideer til ændringer af dokumentation og eget projekt. De afleverede review er rettet med henblik på, hvad der er svært at forstå eller dårligt beskrevet. Dette gør reviewet objektivt. Der medtages ikke forslag til, hvordan man kunne ændre projektet, da det ikke er reviewernes opgave at komme med løsninger til modtagers problemer. Ved modtagelse af review er der holdt en neutralt tilgang. Fokus er lagt på at få så meget som muligt ud af de kommentarer, der modtages. Disse kommentarer har herefter kunnet diskuteres på et efterfølgende internt møde.

### 7.3 Specifikation og Analyse

Dette afsnit omhandler specifikation og analyse i forbindelse med projektets opstillede krav, samt bearbejdelsen og tankerne bag udarbejdelsen af kravspecifikationen (se 2 Kravspecifikation side 5 i projektdokumentationen).

Der blev foretaget undersøgelser af de forskellige krav, der blev opstillet til projektet, for at sikre at kravene faktisk kunne opfyldes. Der blev efterfølgende diskuteret hvorvidt forskellige sensorer og aktuatorer skulle implementeres i systemet. Det resulterede i, at systemet skulle indeholde en temperatur-, jordfugt-, luftfugt- og en lysintensitetssensor.

Til styring af systemet blev det valgt at bruge DevKit8000 som embedded system, og PSoC 4 Pioneer Kits til at styre hardwaren med. Det blev bestemt at bruge en PSoC Master, som skulle have forbindelse til DevKit8000 igennem UART. UART blev valgt som kommunikationsvej, da der allerede var kendskab til UART. Valget af UART gjorde fejlfinding nemt, da det var muligt at teste ved brug af en PC. PC'en kunne læse, hvad der blev sendt, og skrive tilbage ved brug af tastaturet på PC'en. Til kommunikation mellem PSoCs blev det besluttet at bruge I<sup>2</sup>C jf. projektoplægget. Beslutningen om at buskommunikationen skulle være af typen I<sup>2</sup>C blev taget på baggrund af flere forskellige faktorer. Modsat SPI er I<sup>2</sup>C i stand til at sende data over relativt lange afstande. I<sup>2</sup>C interfacet har desuden indbygget sikkerhed i standarden.

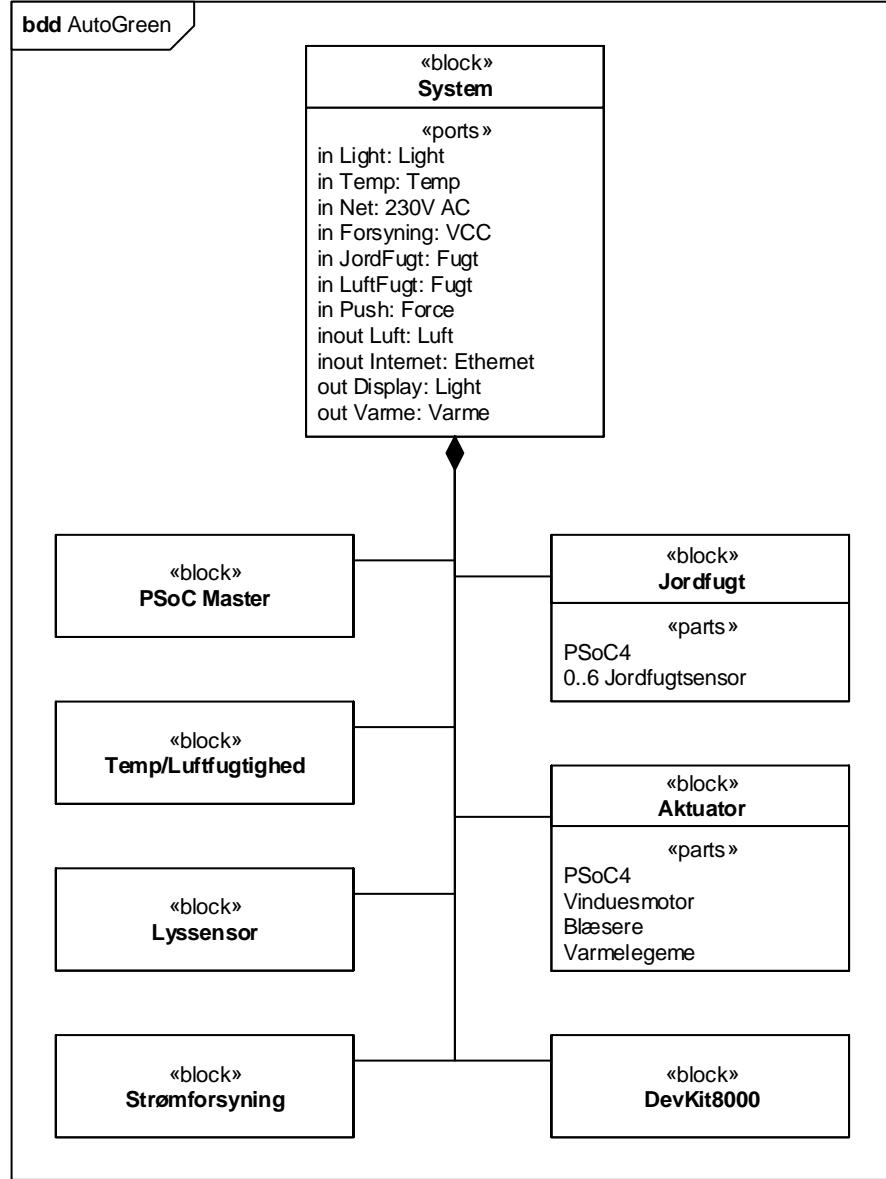
Til regulering af temperatur i drivhuset blev det valgt, at et vindue skulle kunne åbne og lukke, og fire ventilatorer skulle udskifte luften i drivhuset. Formålet med disse er at køle drivhuset ned. Der blev til opvarmning valgt en glødepære, som en simpel måde at opvarme drivhuset på.

Efter de fleste overordnede hardware beslutninger var blevet taget, blev der lavet en overordnet plan for den generelle funktionalitet og hvordan den grafiske brugerflade skulle se ud.

Der kunne efter de generelle beslutninger, skrives use case diagrammer over de ønskede processer og funktionaliteter, og derved give et godt overblik til at opstille endelige krav. Disse krav blev inddelt i funktionelle og ikke-funktionelle krav, således at de valgte arbejdsmodeller blev fulgt, og det var muligt at opstille en endelig kravspecifikation under processen. Dette endte ud i, at en accepttestspecifikation (se afsnit 8 Accepttest side 163 i projektdokumentationen) var mulig at udarbejde, dermed blev V-modellen fulgt.

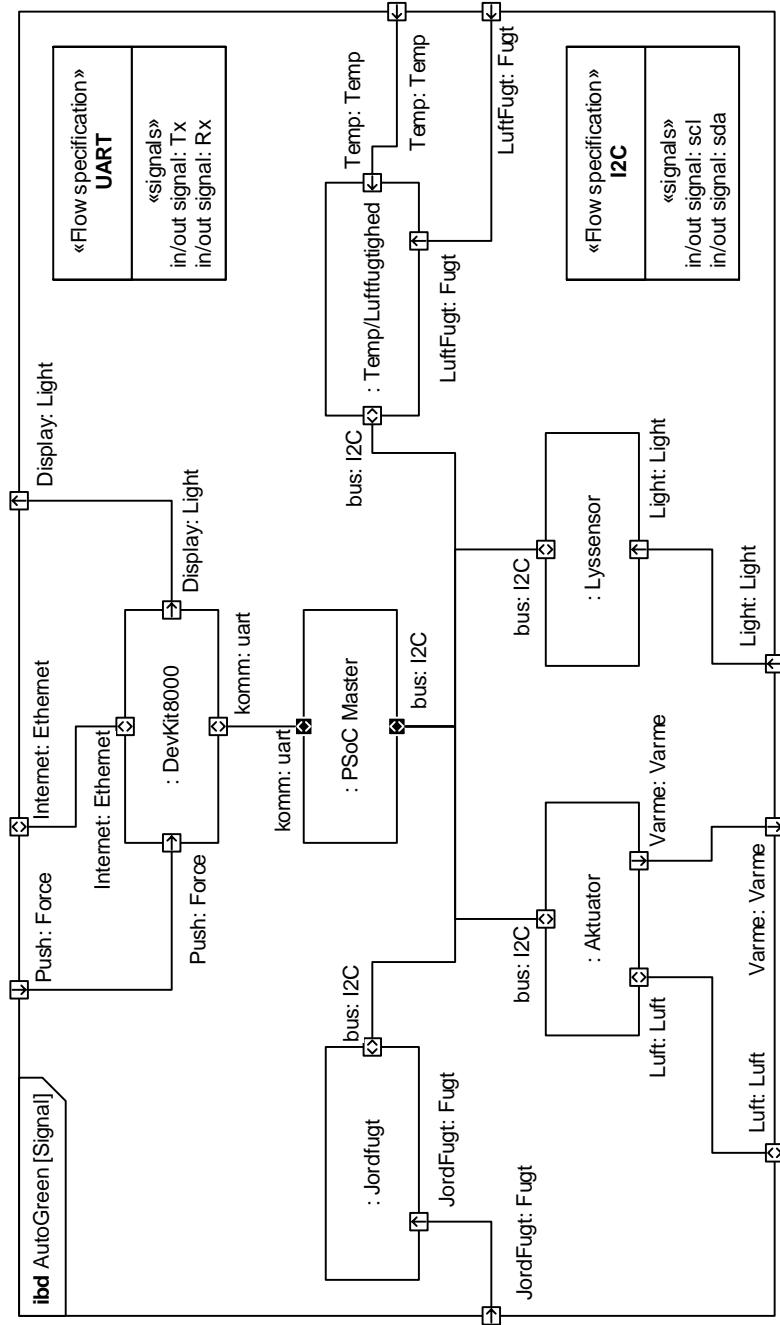
## 7.4 Systemarkitektur

I systemarkitekturen beskrives grænseflader for systemet og hvilke blokke det består af. Til at beskrive dette er der anvendt en række BDD'er og IBD'er. Nedenfor er de vigtigste af disse vist. For mere detaljeret beskrivelse se afsnit 3 Systemarkitektur på side 27 i projektdokumentationen.



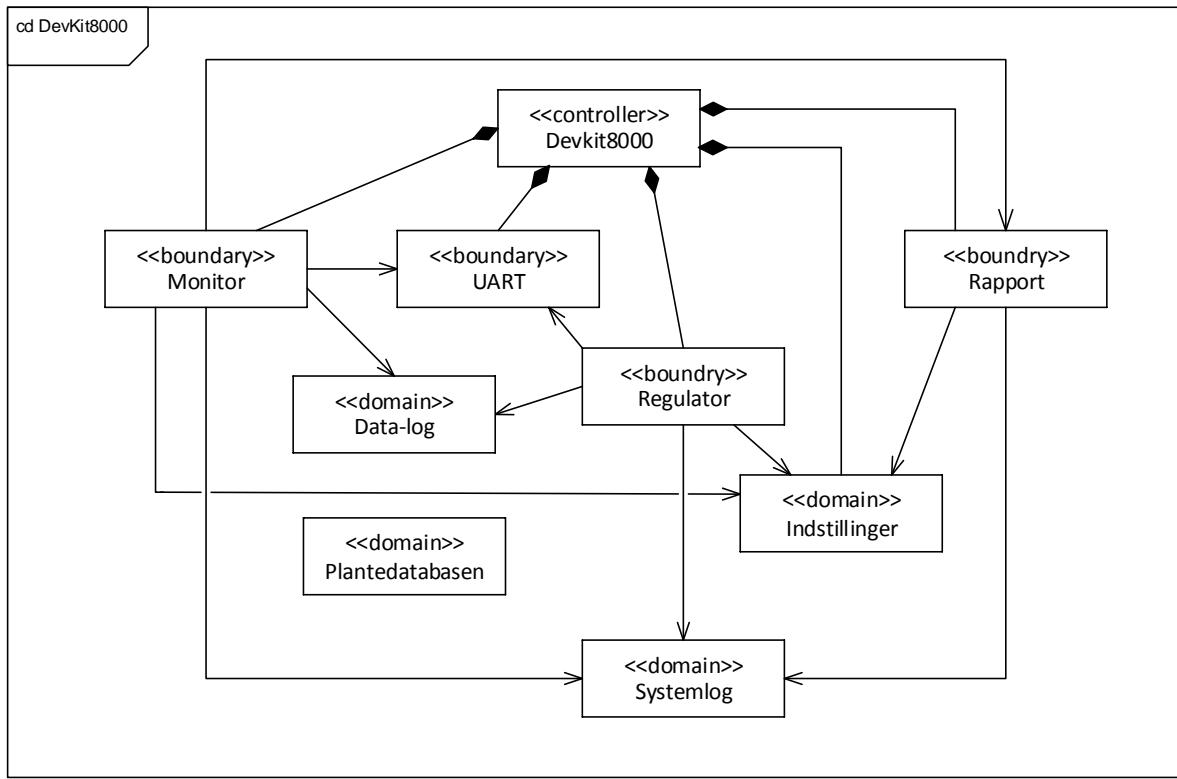
Figur 5: BDD for System.

På Figur 5 kan der skabes et overblik over systemet og hvilke underblokke det består af. Det kan ses, at systemet består af syv underblokke, bl.a. PSoC Master, DevKit8000 mm. I blokken System, der består af alle de andre underblokke, vises de porte som hele systemet har, dvs. grænsefladen til omverdenen.



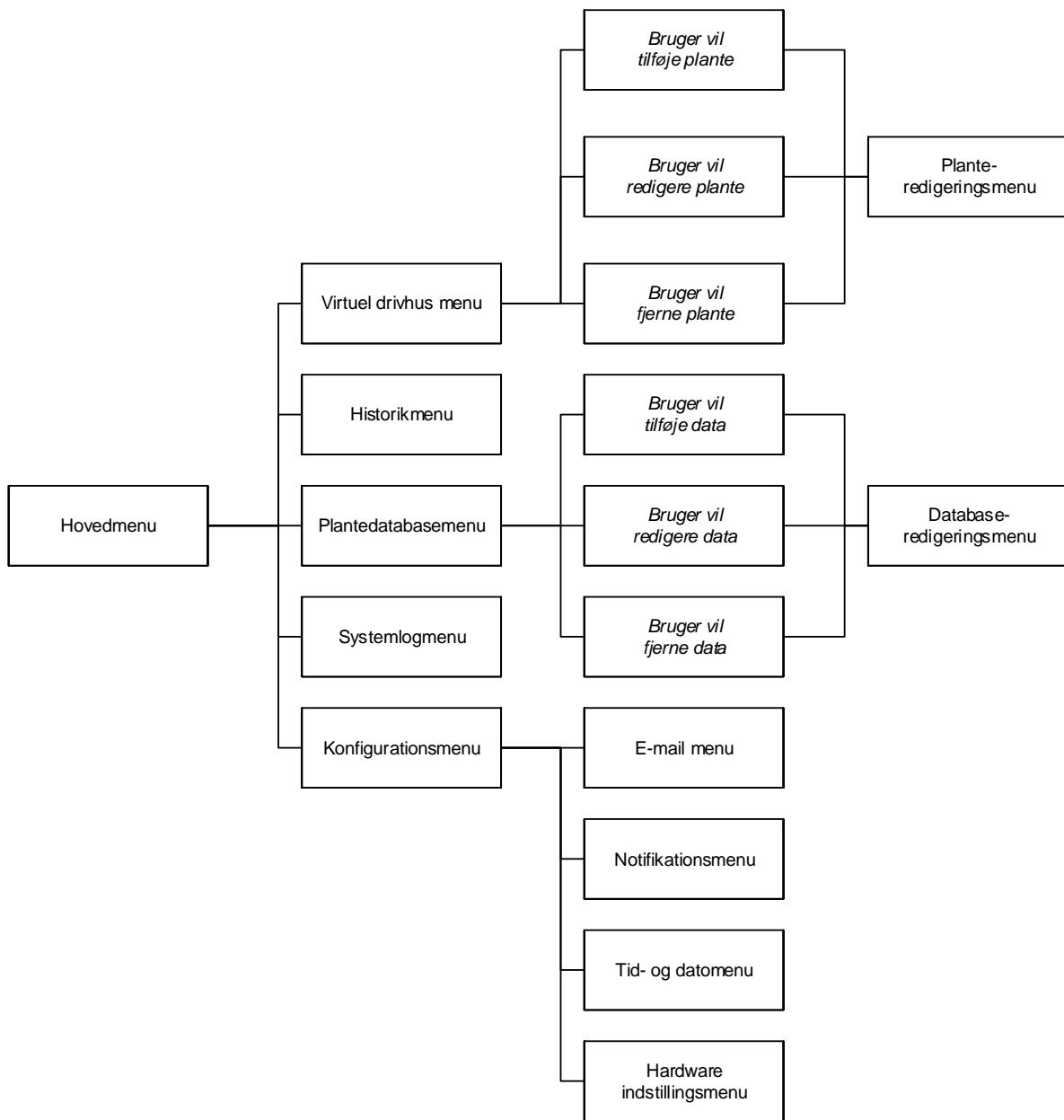
Figur 6: IBD for signaler i systemet.

På Figur 6 vises alle signalerne i systemet, dvs. alle spændingsforsyninger og referencer er udeladt for overskuelighedens skyld. For at beskrive de interne signaler, tages der udgangspunkt i DevKit8000. DevKit8000 spørger løbende PSoC Master om temperatur, luftfugtighed, lysintensitet samt jordfugtighed over UART, denne er detaljeret beskrevet på side 39 i projektdokumentationen. Kommunikationen mellem PSoC Master, Jordfugt, Temp/Luftfugtighed, Lyssensor og Aktuator foregår over en I<sup>2</sup>C bus. Via denne kommunikationsvej kan PSoC Master'en efterspørge alle sensorværdierne og aktivere aktuatorer, hvis DevKit8000 ønsker at regulere klimaet i drivhuset med varmelegemet, vinduet og/eller blæserne. Der kan ses en signalbeskrivelse, hvor alle signaler mellem hver blok er beskrevet i Tabel 12 på side 34 i dokumentationen.



Figur 7: Klassediagram for DevKit8000

På Figur 7 ses et UML klassediagram, som viser relationer mellem klasserne på DevKit8000. Der anvendes to relationstyper; komposition og association. Domain klassen datalog har til opgave at gemme de sensordata som monitor opsamler, jf. Listing 7.1 på side 138 i dokumentationen. Regulatoren anvender denne information og bruger den til at afgøre, om forholdene i drivhuset er som ønsket. DevKit8000 er som angivet en controller klasse, og derfor binder den de øvrige klasser sammen og har den overordnede styring.



Figur 8: Oversigt over AutoGreen's menuer

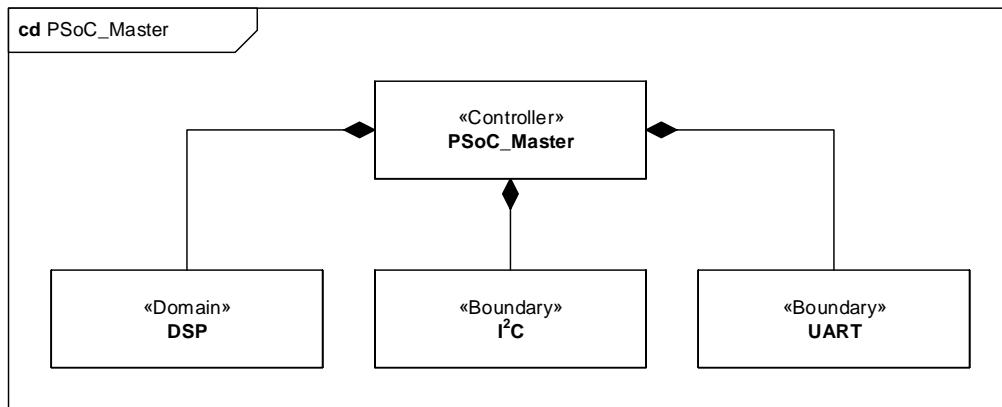
Menuoversigten, der ses på Figur 8, gives et samlet overblik over hvordan de forskellige menuer tilgås igennem systemet. Systemet viser altid hovedmenuen, når systemet starter op. Herfra er det muligt at få et overblik over drivhusets aktuelle klima. Hovedmenuen viser desuden fem knapper, hvor man kan få adgang til undermenuerne: virtuel drivhus-, historik-, plantedatabase-, systemlog- og konfigurationsmenu.

## 7.5 Hardware Design

### 7.5.1 PSoC Master Design

PSoC Master blokken er hardwaremæssigt bestående af et Cypress PSoC 4 Pioneer Kit [2]. Dokumentation af designet for PSoC Master findes i projektdokumentationen side 48.

PSoC Master blokken interfacer med DevKit8000 via UART og samtlige I<sup>2</sup>C enheder på I<sup>2</sup>C bussen. Der er derfor oprettet klasser til at håndtere hhv. UART og I<sup>2</sup>C samt yderligere en klasse til at udføre digital signalbehandling. Disse er vist på Figur 9.

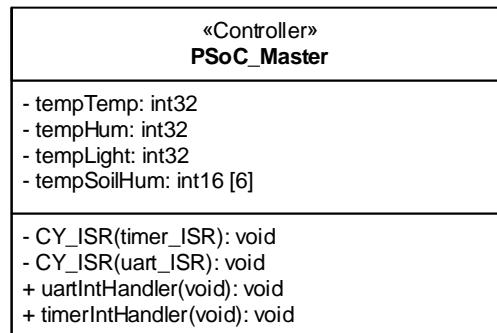


Figur 9: Klassediagram over PSoC Master.

Under designet af PSoC Master blev det nøje overvejet hvilke datatyper, der skulle returneres og bruges som parametre i de forskellige metoder.

Ydermere er det generelt i designfasen overvejet, hvordan de forskellige værdier er repræsenteret i binære mønstre og decimalværdier. Dette er gjort med henblik på at de forskellige klassers metoder skal kunne bruges sammen, så at eksempelvis returværdien fra `getLight()` i I<sup>2</sup>C klassen kan sættes direkte ind i `inputLight()` i DSP klassen.

#### PSoC\_Master Klassen



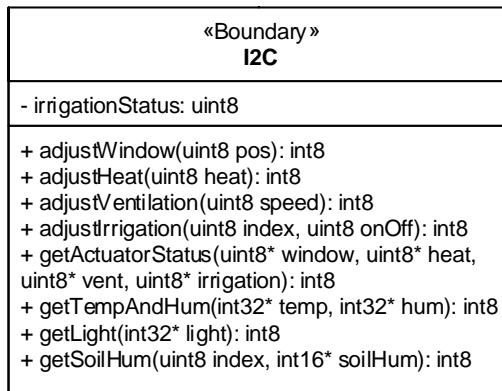
Figur 10: PSoC\_Master klassen.

På Figur 10 ses et diagram for `PSoC_Master`-klassen. Formålet og tanken bag klassen er, at denne skal styre al funktionalitet i PSoC Master blokken. Som udgangspunkt var det tænkt, at al funktionalitet skulle ligge i de to interrupt service rutiner (`timer_ISR` og `uart_ISR`). Dette viste sig sidenhen

at være uhensigtsmæssigt, da der ofte opstod konflikter mellem de to interrupt service rutiner. Der blev derfor lavet to yderligere metoder, nemlig `uartIntHandler()` og `timerIntHandler()` til at håndtere funktionaliteten for hhv. UART og timer. Dette viste sig at løse en del af problemerne. Der er senere erfaret, at samhørigheden kunne være større ved at lade interrupt service rutinerne ligge i deres respektive klasser, da de er tæt knyttet til hardwaren. Dog blev de lagt i PSoC\_Master klassen, da de oprindeligt var tiltænkt at også indeholde al funktionaliteten.

## I2C Klassen

Der er designet en I<sup>2</sup>C protokol (side 44 i dokumentationen), som dækker over samtlige kommandoer der kan sendes. Protokollen er designet med rige udvidelsesmuligheder, således er kommandoerne mulige at ændre, hvis der stødes på problemer med nogle af dem, eller hvis der er behov for at udvide funktionaliteten.

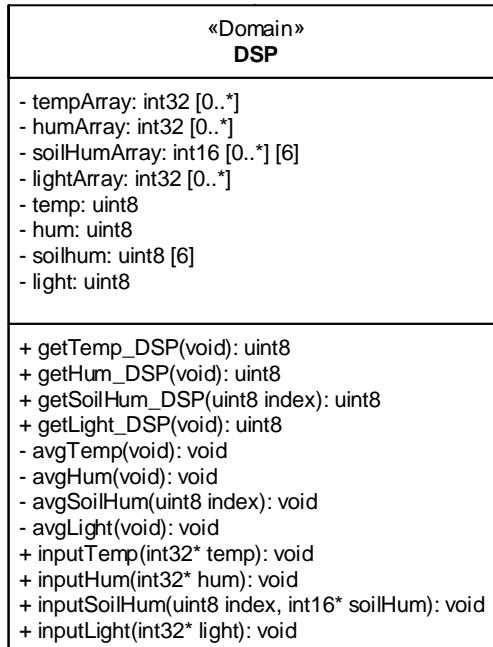


Figur 11: I<sup>2</sup>C klassen.

Klassen I<sup>2</sup>C står for indhentning og afsending af data fra/til hhv. sensorer og aktuatorer. Som det kan ses i klassediagrammet på Figur 11 er der designet metoder, hvis navne svarer til kommandoer. Formålet med klassen er, at varetage kommunikation på I<sup>2</sup>C bussen. Eksempelvis varetager `adjustHeat()` metoden den kommunikation til Aktuator-slaven, når der skal slukkes eller tændes for varmelegemet. Metodens returværdi afspejler om kommunikationen gik godt, og at slaven har forstået beskeden.

## DSP Klassen

DSP klassen har til formål at indsamle og behandle den data, som alle sensorerne leverer. Klassen er designet så den er meget skalerbar, da den kan opsættes til at gemme de alt imellem én og flere hundrede seneste målinger for hver sensor. Ligeledes kan den justeres til hvor mange af de gemte målinger der skal være valide, for at måleresultatet er gyldigt og kan sendes til DevKit8000 frem for en fejlmeddelelse. Ud over denne vurdering er klassen designet, så den foretager et gennemsnit af alle de gemte datapunkter for en bestemt sensor, og gemmer den i en variabel (fx `temp`) der er klar til at blive hentet ud af klassen og sendt direkte til DevKit8000.



Figur 12: DSP klassen.

På Figur 12 ses et klassediagram over DSP klassen. Der er private datamembers både i form af arrays og almindelige variable til at gemme hhv. rá sensordata og klargjorte data. Det kan ses, at `soilHumArray` er et to-dimensionalt array, da det herved er nemmere at implementere behandlingen af jordfugt-data ved for-løkker.

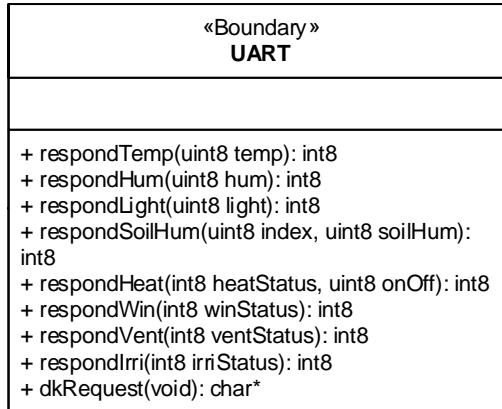
Vejen for et datapunkt ind og ud af klassen kan beskrives ved et par punkter. Her er et eksempel med temperaturen:

- Den målte temperaturdata hentes direkte fra sensoren og gemmes i DSP klassen ved at kalde `inputTemp()` metoden fra PSoC Master klassen (styret af en timer).
- Når `inputTemp()` kaldes, gemmes datapunktet i `tempArray` og den private metode `avgTemp()` kaldes herefter automatisk.
- Metoden `avgTemp()` kontrollerer, om der er nok valide datapunkter, tager gennemsnittet over alle punkterne i `tempArray`, konverterer data'en til formatet der passer til UART protokollen og gemmer resultatet i `temp`.
- Når DevKit8000 anmoder om temperaturen, kan `getTemp_DSP()` blot kaldes for at få den færdigbehandlede temperatur.

Det detaljerede design med klassebeskrivelser m.m. kan findes i afsnit 4.2.1 Domainklasse DSP på side 58 i dokumentationen.

## UART Klassen

Grænsefladen mellem DevKit8000 og PSoC Masteren blev valgt som UART, grundet tidligere erfaringer med denne type af kommunikation. Der blev designet en UART-protokol (nærmere beskrevet i afsnit 3.4 Protokol for UART på side 39 i dokumentationen), som beskriver kommandoer der er nødvendige for systemet. UART klassen virker således som en driver, der håndterer kommunikationen med DevKit8000.

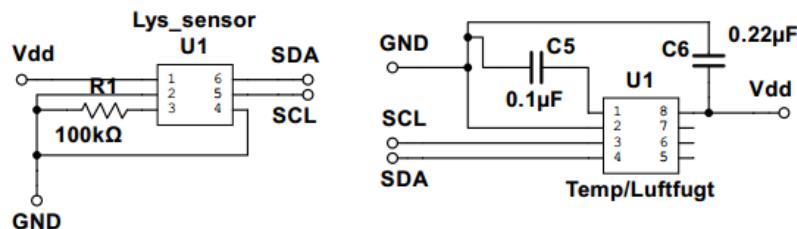


Figur 13: UART klassen

På Figur 13 ses klassediagrammet for UART. Der er designet en public metode, `dkRequest()`, der sørger for at læse den data, der måtte stå i UART bufferen. Tanken er, at denne metode kaldes fra PSoC Master klassen, når der registreres et interrupt. Alle `respond`-metoderne kaldes med en parameter, der valideres af metoden og hvis denne er gyldig (ikke nul), sendes en passende besked til DevKit8000 jf. UART-protokollen. Eksempelvis sendes '`X`'+'`T`', hvis der er en ugyldig temperatur i DSP-klassens' `temp` variabel og '`T`'+'80 hvis der er målt 20°C med temperatursensoren.

## Breakoutboards til I<sup>2</sup>C sensorer

Som temperatur- og luftfugtighedssensor er valgt *HONEYWELL S&C HIH6030-021-001*, der kan måle både temperatur og luftfugtighed og har en I<sup>2</sup>C grænseflade. Lyssensoren hedder *Intersil ISL29010IROZ* og har ligeledes I<sup>2</sup>C grænseflade. Følsomheden på denne kan indstilles til forskellige intervaller, fx 0-128000 lux, hvilket er passende for vores behov. Begge sensorer er forholdsvis små og vi har derfor behov for at lave breakoutboards til dem. Designet af disse er vist på Figur 14.

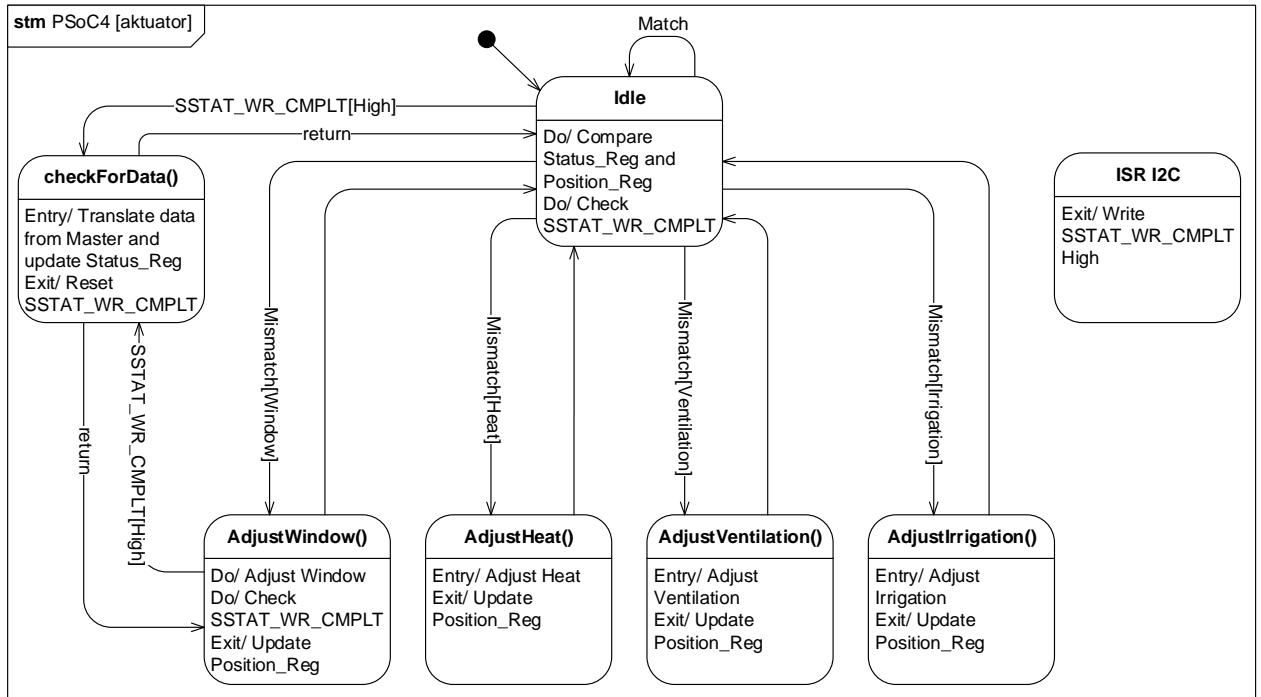


Figur 14: Design af temp/luftfugtsensor og lyssensor breakoutboards.

Størrelserne på modstande og kondensatorer er fundet i de respektive datablade [3] og [4].

### 7.5.2 Slave Aktuator Design

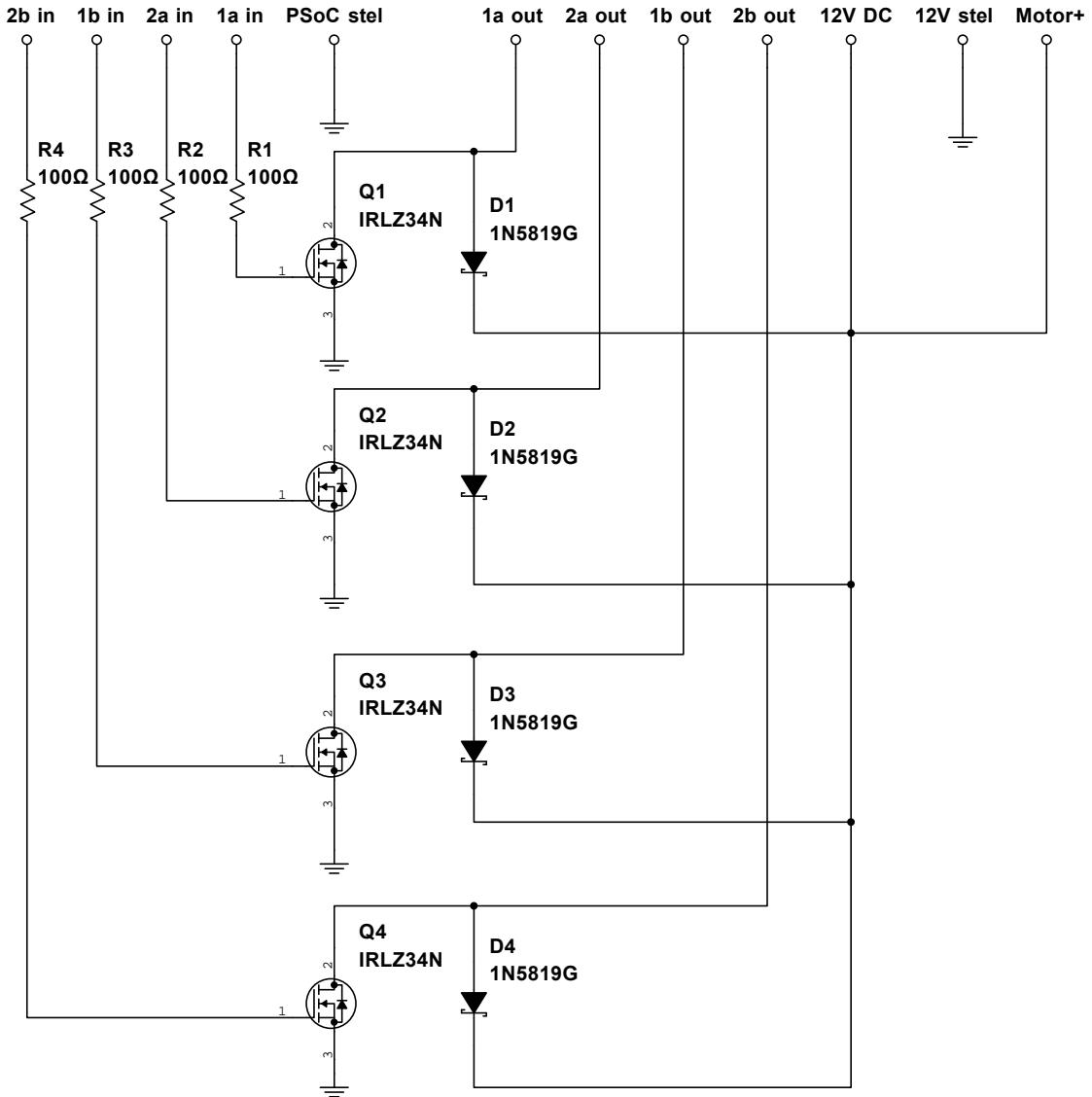
På Figur 15 vises virkemåden for SW på PSoC 4 i blokken Slave Aktuator. Grundprincippet er, at slaven hele tiden er klar til at modtage data på I<sup>2</sup>C bussen, da dette sker vha. et interrupt. Når den ikke er i gang med at modtage data, opdaterer den et status register - hvis der er modtaget data - og sammenligner det med et positionsregister. Hvis de to registre ikke er ens, indstiller den respektive porte for de forskellige aktuatorer og opdaterer positionsregistret.



Figur 15: State Machine for software på underblokken PSoC 4 i Aktuator

De seks porte, som styrer vanding ved tilhørende jordfugtsensorer, er ikke koblet til noget, de giver blot mulighed for at tilkoble et vandingssystem til AutoGreen.

Portene til styring af varmelegeme, ventilatorer og steppermotor er koblet sammen med tre forskellige Mosfetdrive. De tre drive er grundlæggende ens designet; på næste side er designet for steppermotoren vist.



Figur 16: Kredsløb for Mosfet Driver i underblokken Vinduesmotor

Mosfetdriveren på Figur 16 indeholder fire mosfet transistorer, da de fire indgange på motoren ikke skal åbne og lukke samtidigt. Motoren får hele tiden 12V DC, og transistorerne åbner og lukker for stel. De fire dioder er beskyttelsesdioder, som sikrer mod spændingspeak fra spolerne i motoren, når de afbrydes. De fire modstande er også beskyttelse af PSoC'en; såfremt mosfet'erne fejler, vil der afsættes effekt i modstandene frem for i PSoC'en.

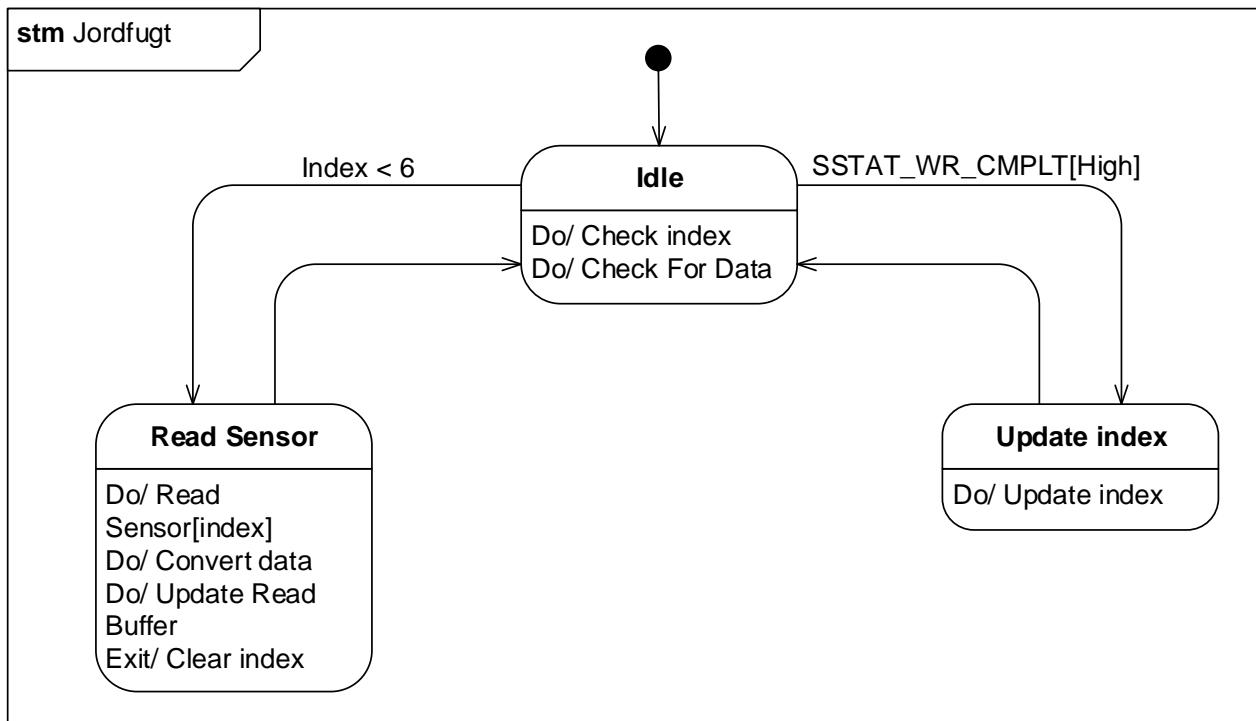
For den fulde beskrivelse, se afsnit 4.4 Aktuator Design på side 65 i projektdokumentationen.

Som udgangspunkt var tanken at gøre designet fuldstændigt interruptbaseret for at effektivisere afvikling af koden. Dette voldte dog en del problemer under implementeringen, derfor blev designet som beskrevet ovenfor.

Der blev desuden eksperimenteret med brug af PSoC'ens flash hukommelse, så aktuel status på aktuatorer ikke gik tabt ved strømafbrydelse. Dette voldte desværre også problemer, hvorfor dette ikke er en del af designet.

### 7.5.3 Slave Jordfugt Design

Slave Jordfugt er tilkoblet op til seks analoge jordfugtsensorer, og den har til opgave at indsamle data fra disse og kommunikere med PSoC Master på I<sup>2</sup>C bussen. Figur 17 viser en state machine for SW på PSoC 4 i Slave Jordfugt. ADC'en i PSoC 4 står hele tiden og opdaterer seks registre, uanset hvad PSoC'en ellers laver. Koden er derfor designet sådan, at når der modtages besked om at læse på en given sensor, opdateres en variabel (`index`), hvorefter data fra den pågældende sensor lægges i read bufferen.

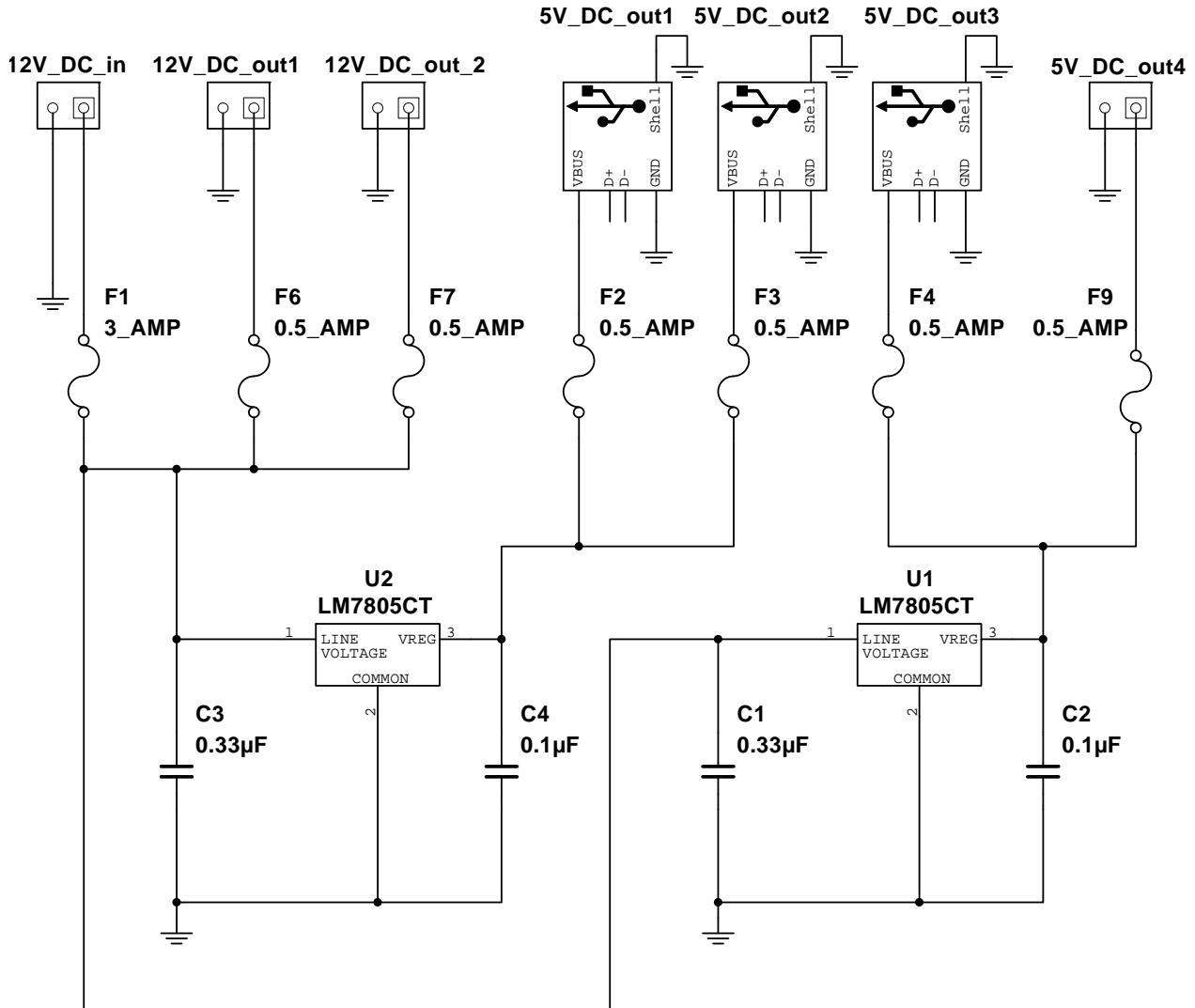


Figur 17: State Machine for SW på PSoC 4 i blokken Jordfugt

De anvendte sensorer er indkøbt, men der foreligger ikke datablad eller lignende. Under MSE øvelse 6, blev der lavet en undersøgelse af sensorernes virkemåde, herunder primært støjmåling [7].

### 7.5.4 Strømforsyning Design

Strømforsyningen skal levere 12V DC til steppermotoren og de fire ventilatorer og 5V DC til PSoC 4 Pioneer Kits og Mosfetdriver til Varmelegeme. Et Multisim diagram for designet er vist på Figur 18.



Figur 18: Diagram for blokken Strømforsyning

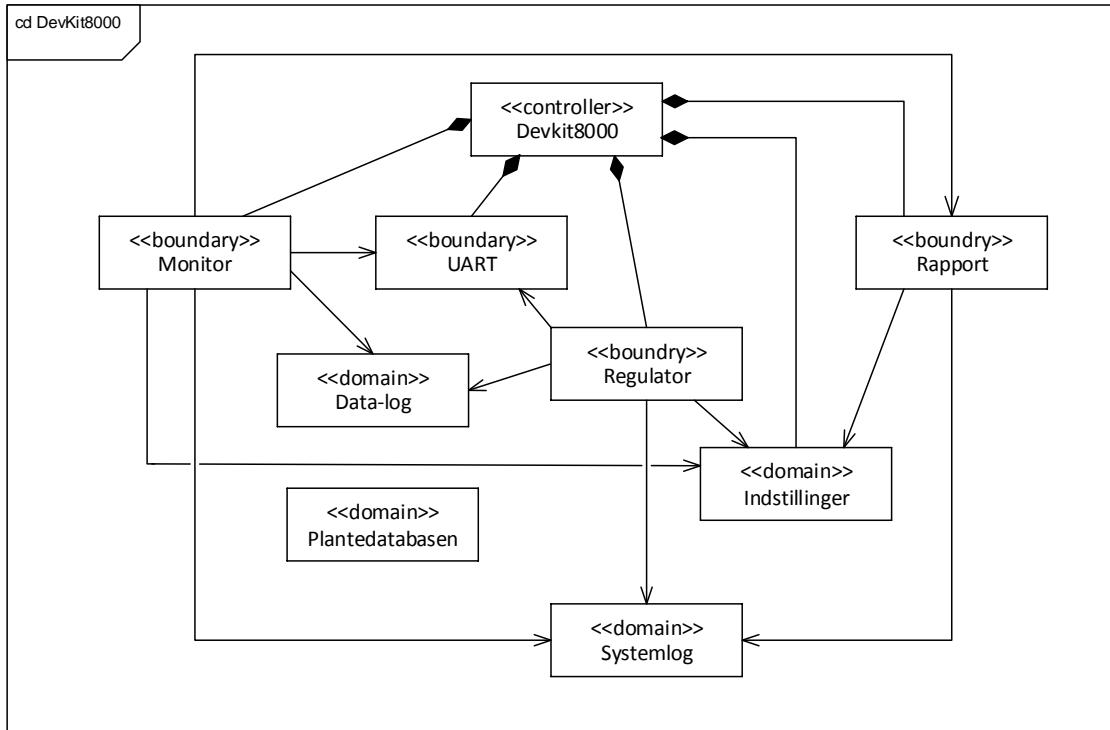
Designet er forsynet med sikringer jf. 3.2.5 Signalbeskrivelser på side 34 i projektdokumentationen. 5V DC forsyningerne laves vha. en spændingsregulator - LM7805 - som er anvendt jf. standard applikationen i databladet [5]. Der kan afsættes op til 7 W i hver af de to spændingsregulatorer, derfor monteres de med køleplade.

For yderligere info se afsnit 4.5 Strømforsyning Design på side 73 i projektdokumentationen.

## 7.6 Software Design

Gennem designprocessen har fokus for softwaredesignet været at parallelisere de vigtigste opgaver i tråde, samt holde ressourceforbruget nede, da memory på DevKit8000 er begrænset. Softwaredesignet er lavet på baggrund af sekvensdiagrammer for relavante use cases på side 17 i projektdokumentationen, så omfanget af softwarearbejdet var nogentlunde kendt på forhånd.

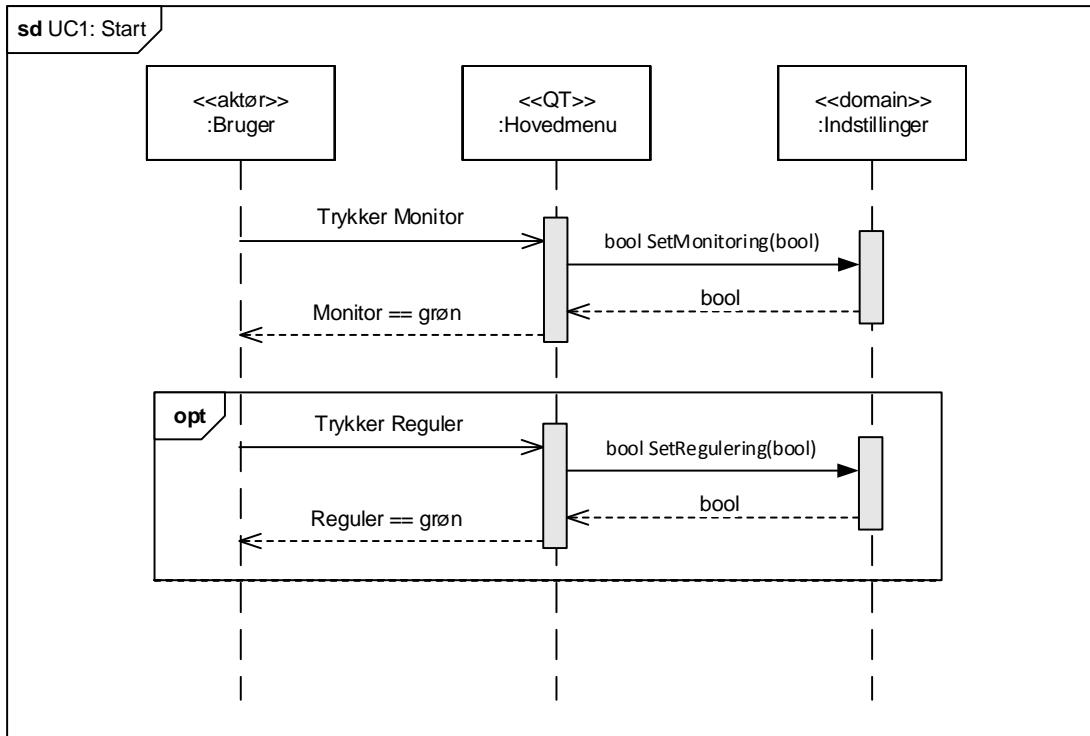
### 7.6.1 Sekvensdiagrammer



Figur 19: Klassediagram for DevKit8000

På Figur 19 ses klassediagram fra systemarkitekturen, som ligger til grundlag for konstruktionen af sekvensdiagrammernes funktion er, at klarlægge de enkelte klassers metoder og associationer, samt at give et overblik over klassernes funktionalitet.

### 7.6.2 Klassebeskrivelser



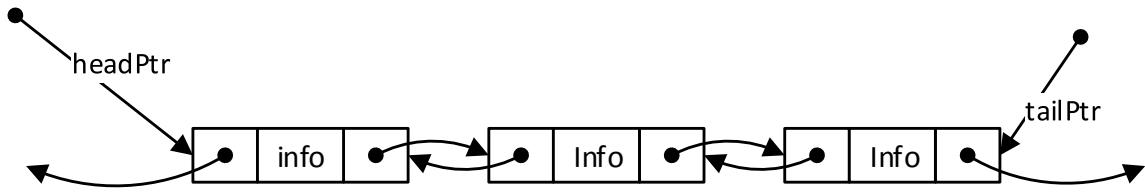
Figur 20: Eksempel for sekvensdiagram.

Under arbejdet med sekvensdiagrammerne (se Figur 20), var det nødvendigt med mindre tests af funktionaliteten og virkemåden i QT [14]. Ud fra disse tests, blev QT set som et bindelede mellem grundsystemets klasser, som fx monitor og regulator. Dette lagde grund for klassebeskrivelser, men ikke beskrivelser for QT-klasserne, da gruppen valgte at udskyde dette til implementationsfasen. Det store fokus har været på trådhåndtering i monitor og regulatoren, samt valg af datastrukturer. Det endelige design omfatter grundsystemets opbygning og de interne forbindelser mellem klasserne.

### 7.6.3 Datastruktur

Valget af datastruktur (se DoublyLinkedList på side 95 i dokumentationen) til lagring af data forskellige steder i systemet, er valgt af hensyn til simplicitet. Det blev bestemt at bruge doublylinkedlist, da den primære brug af datastrukturen skulle være at tilgå det senest tilføjede element. Dette gør doublylinkedlist til et optimalt valg, da man kan tilgå både første og sidste element, igennem head- og tailpointere. For ikke at overkomplicere systemet med forskellige datastrukturer, blev det vedtaget, at doublylinkedlist skulle bruges som datastruktur i alle andre lagringsområder i systemet, foruden indstillinger.

Indstillinger blev implementeret som en klasse, da den havde flere forskellige slags information der skulle lagres. Dataloggen skulle indeholde de samme data i alle noder (se Figur 21), mens indstillinger kun skulle indeholde én af hver information, fx om varmelegeme skal anvendes eller ej.



Figur 21: Eksempel på en DoublyLinkedList

Der blev besluttet at køre flere tråde på systemet for at give mulighed for softwaren at arbejde parallelt. Da det blev bestemt at køre regulator og monitor i hver deres tråd, var det vigtigt at sikre de tilgående klasser som fx UART. Idet både monitor og regulator bruger UART'en til at hente og indsamle data fra, er det vigtigt at disse ikke kommer i konflikt over hvem der skal bruge den. Problemet blev løst ved brug af "mutual exclusion" også kendt som "mutexes", så kun en tråd kan tilgå klassens funktionalitet ad gangen. Til at oprette og køre tråde, blev pthread biblioteket anvendt, eftersom der allerede var kendskab til brugen af pthread fra undervisningen. Det var desuden sikkert, at pthreads fungerede på DevKit8000, da tråde er blevet testet på platformen gennem hele semesteret.

#### 7.6.4 Datatransmission

Til transmission af data imellem DevKit8000 og PSoC Master, blev det besluttet at anvende UART. Denne beslutning blev taget pga. tidlige erfaringer med UART og dermed ville det være oplagt at bruge denne kommunikationsform. DevKit8000 har 3 RS-232 porte, og der er mulighed for at mappe UART'en til andre ben på addon-boardet (se UART opsætning på side 148 i projektdokumentation). I systemet var der brug for at kende tidsstemplingen på det indsamlede data, som hentes fra drivhuset. Der kunne være brugt en tråd, som hele tiden tæller tiden op i system, men sandsynligheden for præcisionsproblemer ville være stor. På DevKit8000 kører der en Linux-distribution der allerede har en timer, som holder fint styr på tiden. Derfor blev det besluttet at bruge denne timer, for gøre systemet mere simpelt.

#### 7.6.5 Intertrådkommunikation

For at få en bedre ide om hvad systemet laver og hvilke systemhændelser som finder sted, blev der lavet en systemlog. I forbindelse med dette, skulle der oprettes en ekstra tråd, som modtager beskeder fra de klasser, der har relevante tilbagemelding om hændelser i systemet. Et eksempel er, fx når regulator vurderer at der er for varmt i drivhuset. For at realisere dette, blev der oprettet en messagequeue, som beskeder kan sendes til. Beskederne bliver taget ud og gemt i DoublyLinkedList'en.

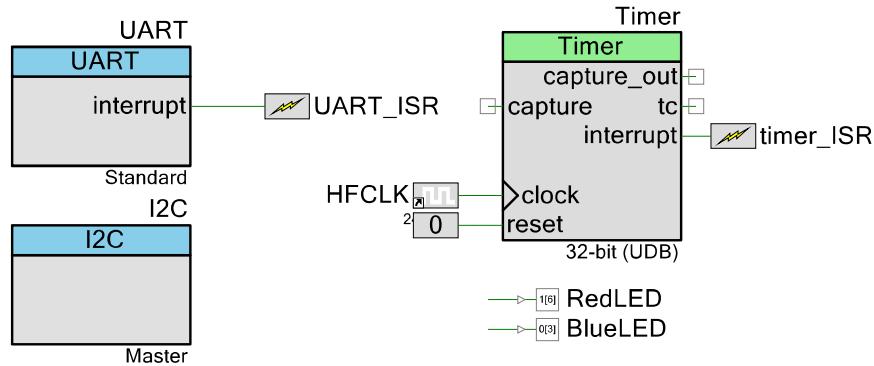
Der skulle også være mulighed for at få tilsendt e-mails fra systemet. Der defineres to typer e-mails: En advarsels e-mail, som bliver sendt, hvis klimaet i drivhuset når et kritisk niveau, og en daglig e-mail, som beskriver hvad der er forgået inden for de sidste 24 timer. Brugeren kan selv styre hvilke e-mails der ønskes. Der er plads til at e-mails kan sendes til maksimalt tre e-mailadresser, som der indstilles i e-mailmenuen (Se beskrivelse i E-mailmenu på side 38 i dokumentation.)

I AutoGreen er der en plantedatabase, som skal indeholde prædefinerede planter, som kan indsættes i det virtuelle drivhus. Denne database skal også bruge DoublyLinkedList'en til at gemme de prædefinerede planter. Ideen bag plantedatabasen er, at kunne samle en række prædefinerede planter og en række selvdefinerede planter, som brugeren har mulighed for at tilføje til det virtuelle drivhus.

## 7.7 Hardware Implementering

### 7.7.1 PSoC Master Implementering

På Figur 22 ses topdesignet for PSoC Master projektet. Det ses hvordan der i det færdige design er udnyttet interrupts fra både UART og Timer. Disse har til formål at fortælle systemet når det er tid til hhv. at hente data fra UART og opdatere data fra sensorer.



Figur 22: Top Design for PSoC Master

Som udgangspunkt var der valgt en temperatur- og luftfugtighedssensor med navnet *HONEYWELL SENS HIH6030-021-001* [3]. Arbejdet med denne sensor har ikke været problemfrit, da værdierne der kunne læses fra sensoren ikke var valide. Det var tydeligt at sensoren ikke afgav de korrekte værdier, da de lå fast på to forskellige størrelser, der i hvert fald ikke afspejlede virkeligheden. Om dette evt. skyldes en kortslutning eller at sensoren var brændt af, vides ikke.

Der blev valgt en ny temperatursensor (LM75 [6]) som erstatning. Denne er dog uden mulighed for at måle luftfugtighed, men dette er acceptabelt, da det kun er temperaturen der er vigtig for systemets virkning. Lyssensoren [4], blev ligeledes nedprioriteret grundet tidsmangel og er derfor ikke implementeret.

Gruppen har under udvikling af UART-delen til PSoC Master draget stor nytte af at UART kan kobles direkte på en PC. PSoC 4 Pioneer Kit giver via den monterede hjælpe-chip (PSoC 5) mulighed for at køre UART direkte fra USB og ud på PSoC 5'ens header. Der er ligeledes under implementering af det færdige system brugt UART fra en PC til at sikre sig, at de givne input var korrekte i forhold til et svar fra systemet.

Ved implementering af UART klassen og I<sup>2</sup>C klassen er der taget udgangspunkt i de respektive protokoller, der er beskrevet i dokumentationen i hhv. afsnit 3.4 Protokol for UART på side 39 og afsnit 4.1 I<sup>2</sup>C Protokol på side 44.

Ved integrationstest med resten af systemet blev det valgt at timeren laver interrupt hvert andet sekund, hvilket betyder at alle sensordata også opdateres hvert andet sekund. Dette viste sig at give færrest problemer i kommunikationen mellem DevKit8000 og PSoC Master'en. Der kan i enkelte tilfælde stadig opstå en situation hvor kommunikationen her brister, hvilket resulterer i en fejlmeldelse på systemets interface. Dette rettes dog allerede næste gang DevKit8000 efterspørger sensorværdier.

DSP klassen er implementeret, så der er mulighed for at variere hvor mange datapunkter, der skal midles over og dermed hvor lang tidsforsinkelse, der er på fx temperaturændringer. Her er der valgt, at der skal tages gennemsnittet af to punkter, hvilket giver en kort tidsforsinkelse, og derved er det lettere at regulere temperaturen i drivhuset.

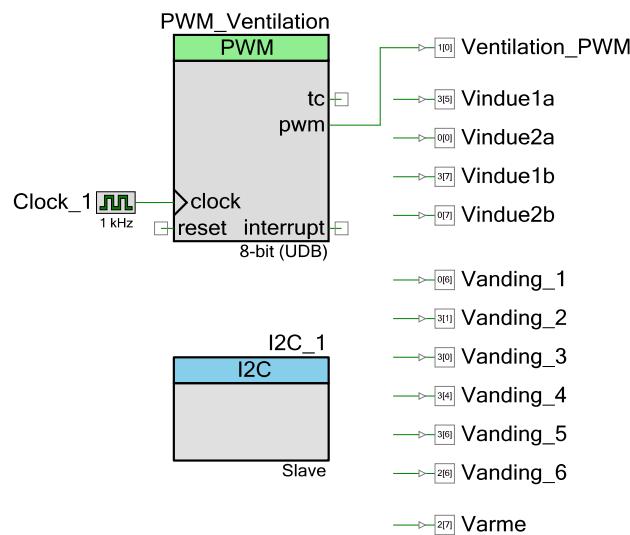
### 7.7.2 Slave Aktuator Implementering

Figur 23 viser topdesignet fra PSoC Creator.

Der er porte, som styrer hhv. varmelegeme, vanding, vindue og ventilation.

Porten til ventilation styres vha. en PWM komponent. Det er egentlig ikke nødvendigt, men derved er blokken forberedt til PWM styring af ventilatorerne, ved senere implementering af dette; fx i forbindelse med PID regulering af temperaturen. Dette udnyttes samtidigt til at køre med en dutycycle på 50%, når ventilatorerne er tændt, for at undgå for voldsom ventilation. Ventilatorerne startes altid med en dutycycle på 100% i et kort øjeblik, før der skrues ned til de 50%. Det sker for at sikre at ventilatorerne opnår tilstrækkeligt højt omdrejningsmoment til at sætte i gang.

Det er umiddelbart opagt at styringen af varmelegemet også foretages vha. PWM, men det er ikke muligt, da varmelegemet tændes og slukkes med et mekanisk relæ, der ikke kan håndtere høje frekvenser.



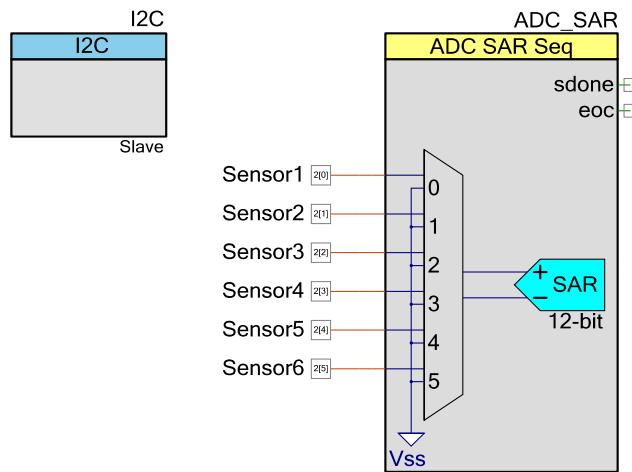
Figur 23: TopDesign.cysch for PSoC 4 i Aktuator

For detaljeret beskrivelse af afvikling af koden, der er anskueliggjort i state machinen under design i denne rapport (7.5.2 Slave Aktuator Design Figur 15, side 22), se afsnit 6.2.2 SW PSoC4 på side 122 i projektdokumentationen.

Print til de tre mosfet drivere til hhv. varme, ventilation og stepper motor, udlægges i Ultiboard og komponenter loddes på. Se fx afsnit 6.2.5 HW Vinduesmotor på side 132 i projektdokumentationen for nærmere info derom.

### 7.7.3 Slave Jordfugt Implementering

Den - ganske simple - HW der syntetiseres i blokken Slave Jordfugt er vist på Figur 24. Der er en I<sup>2</sup>C slave komponent, der varetager kommunikationen på I<sup>2</sup>C bussen og en analog til digital converter, der læser på de seks jordfugtsensorer. Når ADC'en startes, opdaterer den løbende seks registre, uanset hvad PSoC'en ellers laver.



Figur 24: TopDesign.cysch for PSoC 4 i Jordfugt

For detaljeret information om den kode, der afvikles jf. state machinen under design i denne rapport (Figur 17, side 24), se afsnit 6.4 Jordfugt på side 135 i projektdokumentationen.

### 7.7.4 Strømforsyning Implementering

Designet (Figur 18 side 25) udlægges i Ultiboard, printet bestilles, og komponenter loddes på. For yderligere information derom, se afsnit 6.3 Strømforsyning på side 133 i projektdokumentationen.

## 7.8 Software Implementering

### 7.8.1 QT Integration

Implementeringen af SW på DevKit8000 begyndte med en opdeling af klasser, således at hver klasse kunne testes individuelt med teststubs for manglende associationer. Derefter blev hele grundsystemet koblet sammen, en klasse af gangen, før den endelige integration med QT. Selve sammenkoblingen forløb problemfrit, men QT-integrationen tog længst tid af integrationsfasen.

Første problem softwaregruppen havde, var deling af ressourcer mellem de forskellige QT-klasser. Den grundlæggende løsning var først at gøre alle grundsystemsklasser statiske, så de kunne ses globalt - dette virkede dog ikke med pthreads. I stedet blev der lavet en referencestruct, som indeholdt pointers til relevante grundsystemsklasser. Den er oprettet i main og structen bliver sendt med i hver QT constructor.

Et af de største problemer var live opdatering af UI'en på hovedmenuen. Det første forsøg på at løse problemet var, at bruge slots og signaler, hvilket er QT's foreslæede løsning. Dette indebar at monitor-klassen skulle nedarves fra `QObject` [17] og at der skulle implementere et signal, som udsendte GUI data'en for alle klasser, der abbonerede på dette signal. Dog kunne signaler ikke blive sendt fra en klasse kørt i en pthread, så som en alternativ løsning blev der oprettet en `QTimer`, der opdaterede GUI'en med faste intervaller, ved at hente informationen direkte fra monitor.

### 7.8.2 Timers på DevKit8000

For at indstille tiden på DevKit8000, skulle der bruges en C++ API, som kunne sætte tiden. De metoder der kan bruges til at sætte tiden med, er linux's `gettimeofday()` og `settimeofday()`. Måden det virker på, er ved at starte tiden fra 1. januar 1970. Dette betyder, at dags dato svarer til den tid, der er gået siden 1970. Men dette API er obsolete derfor kunne det ikke bruges. Det blev i stedet valgt at få koden til at manipulere tiden ved at køre et script med et mere stabilt interface. Når først tiden er sat, anvendes et normalt C++ API til at få fat i den aktuelle tid på systemet.

QT skabte problemer (se afsnit 7.11 GUI - QT på side 158 vedrørende løsning) angående brugen af pthreads, da QT har implementeret sin egen trådprotokol kaldet Qthread. Måden problemet blev løst på, var at starte pthreads i QT main.cpp filen. Et andet problem var, at monitor og regulator konfliktede om adgangen til UART. Dette gjorde, at regulator var langsom mht. at sende kommandoer til PSoC Masteren, da regulatoren lavede flere små kald til UART'en, mens monitor brugte et større kald til UART. Siden monitor og regulator kører i hver sin tråd kan monitor gå ind i mellem hver kald regulator laver. For at undgå dette satte vi monitor og regulator i samme tråd, hvilket heldigvis ikke gav nogen problemer mht. systemets ydeevne, da monitor skal hente nye data før regulator kan regulere fra dem.

### 7.8.3 UART Implementering

UART'en er implementeret ud fra et open-source bibliotek [15], hvor koden er nedskåret til at kunne bruge simple funktionskald til at sende data ud på UART'en. Der kan hentes data ind med et pull, så det data der er blevet sendt til RX-benet på DevKit8000, bliver hentet ind i en buffer. Når data er hentet ind i buffer, kan der tjekkes på det data der befinner sig i bufferen og ud fra dette, afgøre hvad der skal returneres eller lagres. Hvis fx bufferen er tom på den første plads, vil UART'en i de fleste tilfælde, sende en kommando igen og vente på et nyt svar. Dette tilføjer noget fejlhåndtering (se afsnit 7.8 UART på side 148 i projektdokumentationen), hvilket er smart, da en kommando kan gå tabt i kommunikationen mellem PSoC Master og DevKit8000.

Systemet gav fejl på nogle enkelte beskeder sendt via UART'en, hvilket umiddelbart skyldes QT timeren. QTtimeren opdaterer GUI'en mens UART'en venter på data fra PSoC Masteren. PSoC

Masteren venter fx med at svare på UART forspørgsler, mens den kommunikerer med Slave Aktuator. Disse problemer med kommunikation på UART kunne ikke helt løses, men UART'en er lavet til at foresørge op til tre gange med lidt delay, afhængigt af hvilken kommando der er sendt, hvis PSoC Masteren giver en fejlbesked tilbage.

UART'ens standard udgange på DevKit8000 forekommer på J1, J2 og J3 [16], men da der her udsendes op til 15V, var det en nødvendighed at mappe UART'ens RX og TX til andre udgange, som kun udsender 3.3V. Udgangen blev skiftet af hensyn til PSoC Masteren, som arbejder inden for 3.3V. Dette gjorde også, at der ikke skulle bruges et RS-232 standardkabel, men i stedet kun tre forbindelser; Tx, Rx og reference.

#### 7.8.4 Regulering

Ikke alle QT-menuer er blevet implementeret, da der blev lagt større vægt på optimering af regulatoren. Regulatoren blev i første omgang implementeret med, at når der skulle fortages en regulering vil den altid sende en kommando om fx at åbne vinduet, selvom vinduet allerede var åbent. Dette blev undgået ved at sætte interne bools ind i regulatoren, så den selv har styr på om vindue, varmelegeme og blæsere er tændte eller slukkede. Denne ændring gav mindre data, der skulle sendes over UART'en og gjorde derfor regulatoren hurtigere til at komme igennem dens kode. En anden stor ændring der kom igennem implementeringen af regulatoren var, at ændre reguleringsområderne så regulatoren tænder for varmen, når der var 1 grad for koldt, istedet for 4 °C, som var udgangspunktet. Regulering når der bliver for varmt sker også inden for 1 grad, hvor vindue og blæsere bliver hhv. åbnet og tændt. Dette er med til at systemet godt kan tænde og slukke for blæsere, varmelegeme eller vindue relativt ofte. For at gøre regulatoren en smule hurtigere, blev beskeder til systemloggen fra regulatoren også fjernet. UART skriver i forvejen til systemloggen, hvorfor det ikke var nødvendigt for Regulering også at skrive i denne.

Efter implementeringen blev det konstateret, at mange af de udfordringer, der var undervejs kunne være undgået, hvis alt kode var skrevet med QT syntax fra begyndelsen - men på grund af tidspres, valgte softwaregruppen at fortsætte implementering af grundsystemet direkte ind i QT miljøet.

## 7.9 Resultater og Diskussion

Overordnet er alle "skal" krav blevet opfyldt for AutoGreen projektet. Det vil sige, at systemet kan overvåge temperaturen, samt regulere den på baggrund af måleværdier. Systemet giver mulighed for at vælge om varmelegeme og/eller blæsere skal være aktive under regulering, og AutoGreen indeholder en grafisk brugerflade udviklet i QT. AutoGreen indeholder desuden nogle "bør" funktionaliteter, da brugeren har mulighed for at overvåge op til seks planters jordfugtigthed i hovedmenuen.

Lysintensitet- og luftfugtighedsensorer er blevet droppet sent i forløbet, da det viste sig at der var problemer med at få dem til at fungere. Systemet har en fungerende datalog over alle måleværdier, dog er der ikke implementeret en grafisk fremstilling af dem. Ligeledes er plantedatabasen ikke implementeret. Systemloggen virker, men kan kun vise sidste hændelse, dvs. den er ikke implementeret færdig. System opfylder kun et enkelt "kan" krav, nemlig mulighed for tilslutning til et automatisk vandingssystem, ellers opfylder AutoGreen ikke flere "kan" krav.

En af de ting som virker overraskende godt i AutoGreen, er selve reguleringen af temperaturen. Vi havde forventet at få problemer med at kunne regulere temperaturen med en præcision på  $+/- 2^{\circ}\text{C}$ , men det er på grænsen til at det lader sig gøre med  $+/- 0,5^{\circ}\text{C}$ , og så er det pludselig oplosningen på temperatursensoren, der sætter begrænsningen.

Det gode resultat grunder i et sammenfald af flere ting. For det første er aktuatorerne passende dimensioneret i forhold til drivhusets størrelse. Der ligger ikke dybe tanker og en masse beregninger bag dette. Der var monteret fire ventilatorer i drivhuset da vi overtog det, hvilket syntes en smule voldomt; derfor kører de med en duty cycle på 50% for at undgå overshoot i forbindelse med køling. De 50% var intet mere end et gæt, som viste sig at være fornuftigt. Vi indkøbte i starten af forløbet 3 stk. 100W glødepærer til at bruge som varmelegeme, men det viste sig hurtigt, at en enkelt var passende, for at undgå overshoot i forbindelse med opvarmning.

Der var fra begyndelsen desuden lagt op til at udvide med PWM styring og PID-regulering af varmelegeme og ventilation under hhv. opvarmning og afkøling af drivhuset. Dette blev bla. pga. tidsnød ikke implementeret, så vi regnede fx med at varmelegemet ville komme til at stå og tænde og slukke, når temperaturen i drivhuset nåede det ønskede niveau. Dette er også tilfældet, men det sker meget langsommere end vi havde forventet, da glødepæren ikke bliver kold i det samme øjeblik den slukkes; derfor falder temperaturen kun langsomt.

UART kommunikationen mellem DevKit8000 og PSoC Master kom desværre til at fungere dårligere end forventet. Der er en del fejlkomunikation, som primært kommer til udtryk ved udfald på jordfugtsensorerne. Vi mener dette kan skyldes problemer med timing i SW på hhv. DevKit8000 og PSoC Master, da meget af koden er interruptbaseret, der afvikles flere strenge samtidigt og der skal ventes på retursvar, når der spørges efter en sensorværdi eller sendes en kommando til en aktuator. Problemet kan også skyldes decideret fejl på selve den fysiske UART kommunikation. Problemet kan muligvis afhjælpes ved at anvende skærmet kabel mellem DevKit8000 og PSoC Master, eller mere sandsynligt ved at anvende en komplet UART med alle 9 forbindelser i stedet for AutoGreen's mere skrabede model, der kun indeholder Tx, Rx og en reference. Alternativt skal design af SW på PSoC Master og/eller DevKit8000 skrives helt om. Under alle omstændigheder ligger her et oplagt udviklingspotentiale for systemet.

## 7.10 Udviklingsværktøjer

I dette afsnit vil der blive gennemgået de forskellige udviklingsværktøjer, som er blevet anvendt under dette projekts design-, implementerings- og integrationsproces.

### 7.10.1 PSoC Creator

Til programmering af projektets PSoC 4 Pioneer Kits blev PSoC Creator udnyttet. Programmet er anvendt til kodning og debugging af PSoCs under implementering af software. Dette udviklingsmiljø har et stort bibliotek af klargjorte komponenter og kode hertil, hvilket gør processen af programmering af PSoCs effektiv.

I dette projekt er der valgt at bruge PSoC 4 processoren på de tre Pioneer Kits, det er dog muligt at bruge PSoC 5 processoren i stedet, hvis dette ønskes, da denne processor har mange flere funktionaliteter. Grunden til at PSoC 5 komponenten ikke er valgt i dette projekt er, at der ikke har været brug for den funktionalitet PSoC 5'eren har, frem for PSoC 4'eren. Ved at bruge PSoC 4, kan PSoC 5 processoren ligeledes bruges til at debugge direkte på et PSoC 4 projekt, hvilket har gjort fejlfinding af PSoCs effektiv.

### 7.10.2 Multisim

Gruppen har valgt Multisim til design af hardware. Styrkerne ved Multisim er at skabe et overblik og muligheden for at simulere forskellige hardware moduler, med de ønskede komponenter, fra et rigt bibliotek. Svaghederne ved Multisim er, at det nogle gange ikke er muligt simulere et kredsløb korrekt samt i værste tilfælde, at Multisim ikke har mulighed for at simulere en specifik hardware komponent. I dette projekt er Multisim anvendt under design og implementeringsprocessen for hardware.

### 7.10.3 Ultiboard

Ultiboard blev brugt til at tegne og konstruere printplade layouts. Ultiboard kan, i forbindelse med Multisim, uddrage de komponenter der findes i designs fra Multisim og konstruere printplade layouts ud fra disse.

### 7.10.4 QT Creator

QT Creator er brugt til designe GUI'en til DevKit8000. Programmet har en del funktionalitet, der gør at de GUI programmer man kan lave i programmet også kan køre på andre OS platforme fx Windows eller Linux. Dette er dog også problematisk, da denne funktionalitet kommer på bekostning af den normale implementering. Dette ses i deres interne primitive typer, som fx er string kaldet en QString, så alle normale strings skal castes til QString for at kunne bruges i deres UI klasser.

### 7.10.5 Git

Git er et versionstyringsværktøj til at vedligeholde kildekode. Git er et stærkere værktøj end SVN, der blev brugt i sidste semester projekt. Det er dog mere kompliceret at bruge pga. de flere valgmuligheder som medfølger. Som repository host er der valgt Github grundet stabilitet, da gruppen oplevede problemer med RiouxSVN sidste semester.

## 7.11 Opnåede Erfaringer

### 7.11.1 David

På dette semester har der været rigtigt meget nyt at lære, hvilket er fedt. Jeg synes, at jeg blevet bedre til lave programmering og Idoms og pattens og ikke mindst datastrukturer og forstå dem, så man vælger den rigtige. En ny ting er også at få et program til at køre kode i hver sin tråd, så man lave flere ting på en gang. QT har både vært svært og sjovt at lære. Efter at have brugt det, har jeg fundet ud af, at det har sin egen implementering af stor set alle klasser i standard C++. Hvis jeg skulle bruge et system som QT igen, vil jeg helt klart bruge QT's egen implementering af det meste, da det nok ville have sparet en del problemer på længere sigt.

### 7.11.2 Henrik

Dette semesterprojekt har været lærerigt på mange områder. Fagligt føler jeg, at jeg har fået rigtig meget ud af dette forløb. Jeg har hovedsageligt været en del af hardwaregruppen og har især haft stort ansvar inden for Slave Aktuator og Slave Jordfugt. Det er andet semester jeg har været en del af denne gruppe, og jeg mener at gruppen bliver stærkere og stærkere jo mere vi arbejder sammen. Det er en gruppe fuld af mange stærke individer, hvilket hovedsageligt gør at når der arbejdes, arbejdes der effektivt. Alt i alt har det været fedt med et projekt, hvor man har haft mere løse tøjler og dermed mulighed for, at lave mere eller mindre personlige løsninger til et problem.

### 7.11.3 Kasper

Dette semesterprojekt har været meget interessant, men har også været ret frustrerende. Starten af projektet, hvor jeg har arbejdet sammen med gruppen fra det tidligere semester var god, og det var tydeligt at gruppen havde udviklet sig i forhold til det samme forløb på sidste semester. Da gruppen delte sig op i hardware of software, var jeg en del af softwaregruppen. I forløbet med softwaregruppen ville jeg gerne have lavet en del om. Vi troede vi var smarte at konstruere et grundsystem først, hvor vi lavede de fleste funktionaliteter, og derefter sammensætte det med QT, hvilken endte med at give os problemer, da vi kom til at skulle arbejde grundsystemet ind i QT. Dette gav dog viden til, at der i fremtiden skal tages hensyn til hvis jeg igen skal bygge et system op, og så ikke bygge et grundsystem op først, men ihvertfald finde ud af hvordan det skal bygges sammen med GUI'en. Jeg har desuden fået en masse ud af at arbejde med tråde og QT i større omfang, da det har givet indsigt i hvordan, det er at arbejde med større systemer, hvor flere funktionaliteter skal køres samtidig, og der skal være en brugerflade på. Generalt har projektet været meget lærerigt at arbejde på.

### 7.11.4 Kristian S.

Gennem projektet har jeg opnået en god erfaring med Linux, og har personligt godt kunne lide forløbet gennem semesteret. Men når det er sagt, så er der også en del ting, som jeg ville have gjort om. For eksempel troede softwaregruppen, at vi var smarte, da vi lavede grundsystemet først, men det viste sig, at vi havde bygget et hus før vi havde lagt fundamentet (QT). Ellers har min fokus været mest på integrering af de forskellige klasser i QT, specielt med alle de problemer, der fulgte med hver integration. Af tekniske erfaringer vil jeg sige at håndtering og sikring af tråd har fyldt meget, og det kan fx ses i systemloggen, der bygger videre på en ISU opgave, som er lavet til at kunne modtage beskeder fra forskellige tråde, uden at der sker segmentation faults.

### 7.11.5 Kristian T.

Under dette projektforløb har jeg primært dannet mange erfaringer omkring projektopbygning og design på PSoC. Vi er under projektforløbet stødt på mange små og nogle store problemer, som umiddelbart ikke var synlige ud fra et arkitektur- og designperspektiv. Disse problemer har tvunget mig og gruppen ud i at lære mere omkring opbyggelsen af PSoC, og hvordan den er stykket sammen ift. fx interrupts, noget som vi ikke rigtig er kommet ind på i de øvrige kurser dette semester. Jeg har også opnået en del erfaringer omkring fejlsøgning af en opstilling, og hvordan man undgår eventuelle faldgrupper, fx har vi haft dårlig erfaring med at sætte projektet sammen med enkelte harwin-stik, da det var besværligt at sætte sammen igen uden at have dokumentationen foran sig.

En anden væsentlig ting jeg føler, jeg har lært, er at samarbejde bedre med projektgruppen. Dette er forhåbentlig en del oplevelse, men jeg synes den er værd at tage med, da samarbejde på et højere plan også hjælper ved fremtidige projektgrupper.

### 7.11.6 Lasse

AutoGreen har for mig været et spændende og særdeles udfordrende projekt. Selvom projektet virkede simpelt nok, er gruppen alligevel undervejs stødt på en lang række problemer. De problemer der var mulige at løse er via fokus og samarbejde blevet løst tilfredsstillende for det endelige system. Jeg er positivt overrasket over gruppens udvikling, hvilket har gjort sig meget bemærket i form af måden der samarbejdes på nu vs. måden vi samarbejde på i forrige semesterprojekt. Overordnet set er jeg meget tilfreds med forløbet.

### 7.11.7 Morten

Arbejdet med dette projekt har for mig været meget lærerigt. Jeg har efter min egen mening fået et højt fagligt udbytte, særligt i design- og implementeringsfaserne i HW gruppen, herunder primært arbejdet med Aktuator blokken og Jordfugt blokken.

Den største læring ligger dog, efter min mening, i selve processen, selv om læringsmålene på 3. semester fokuserer mere på fagligt indhold frem for processen i forhold til tidligere semestre.

Det har også været interessant selv at skulle formulere projektet, hvor der tidligere har været et endnu mere bundet oplæg.

Gruppen har draget stor fordel af at have arbejdet sammen på forrige semester, og det har været særdeles lærerigt at opleve, hvordan samarbejdet nærmest automatisk optimeres, efterhånden som gruppen er blevet mere sammentømret.

### 7.11.8 Philip

Dette semesterprojekt har været meget lærerigt, både fagligt og procesorienteret. Jeg har været en del af hardwaregruppen, hvor jeg hovedsagligt har arbejdet på PSoC Master. Her har jeg bl.a. fået meget erfaring med udviklingsværktøjet PSoC Creator, PSoC 4 Pioneer Kit og I<sup>2</sup>C interfacet. Jeg har også arbejdet meget med sensorerne, der er anvendt i systemet, og er blevet i stand til hurtigt at sætte mig ind i en sensors interface og funktionalitet. Igennem arbejdet med PSoC Masteren har vi ramt mange små og store problemer, og jeg er derved blevet bedre til systematisk at fejlsøge et system. Det er tydeligt, at gruppen er blevet stærk igennem det sidste år, hvor vi har arbejdet sammen, og derfor har vi arbejdet effektivt igennem hele projektet. Vi er gode til at hjælpe hinanden, og jeg føler, at jeg bidrager godt til samarbejdet, hvilket også giver gode erfaringer.

## 7.12 Fremtidigt Arbejde

Umiddelbart er de resterende krav, som ikke blev implementeret i projektforløbet, oplagte kandidater til fremtidigt arbejde. Dette inkluderer måling af lysintensitet og luftfugtighed, mulighed for at tilføje planter i det virtuelle drivhus, interaktion med plantedatabase, grafer for samtlige måledata, e-mail notifikationer samt fuld systemlog. For at e-mail notifikationer skal fungere ville det også være nødvendigt at implementere internetadgang på DevKit8000, hertil kan det tilføjes at mulighed for trådløst netværk ville være oplagt, da systemet skal sidde i brugerens hjem og ikke nødvendigvis være i nærheden af router el. lign., og man ville slippe for at trække ethernet kabel til sit drivhus. Man kunne også overveje en eller anden form for trådløs kommunikation mellem DevKit8000 og PSoC 4 Master; derved undgår man at brugerfladen skal placeres i drivhuset.

Udover disse opgaver er oplagte muligheder integration med et vandingssystem, så drivhuset er selvstyrende mht. vanding. Dette kunne gøres ved at åbne/lukke magnetventiler for hver plante i drivhuset. For at det ville fungere, bør der også implementeres en form for kalibrering af vandtilførslen i forhold til systemet, så en eventuel vandingsalgoritme ville kunne justere vandmængden fornuftigt.

Under projektforløbet har gruppen indset, at PSoC Master-blokkens funktionaliteter kunne integreres i DevKit8000 og herved spare en hel hardwareblok, da denne har mulighed for I<sup>2</sup>C direkte på kippet. Der er ligeleds også fordele i at lade den digitale filtrering foregå på DevKit8000, da denne har langt flere resourcer tilgængelige i forhold til digital filtrering i software end PSoC 4. En anden mulighed ville være at lade den digitale signalbehandling foregå på PSoC 5, der også forefindes på PSoC 4 Pioneer Kit. Denne er dog upraktisk at udvikle på, da den ikke tilbyder samme debugging funktionaliteter som PSoC 4 processoren.

Gruppen har under forløbet også luftet tanken om at implementere en mobilapp til monitorering og interaktion med systemet, denne ville fx kunne implementeres til Android og iOS. En sådan app ville kunne give brugeren information og mulighed for interaktion med systemet når brugeren ikke er hjemme i en længere periode, som fx på ferie. Brugeren ville ligeledes også kunne anvende appen udelukkende i stedet for touchskærmen på DevKit8000.

Med hensyn til selve hardwaren i drivhuset ville det være oplagt at integrere aktuator og jordfugtmåler på én PSoC 4 og placere denne på samme print som strømforsyning. Dette ville reducere prisen på produktet væsentligt samt skabe mere stabilitet ift. udefrakommende påvirkninger (elektromagnetisk stråling og mekanisk påvirkning). Ligeledes ville det være en fordel at indpakke alle hardwareenheder i passende kabinetter.

## 8 Konklusion

Hvis man kun ser på afsnit 8 Accepttest på side 163 i projektdokumentationen, kan det umiddelbart se ud som de opstillede krav for projektet langt fra er opfyldt, men hvis man sammenholder accepttesten med MoSCoW prioriteringen i afsnit 1 Projektformulering på side 1 i projektdokumentationen, ses det, at de vigtigste krav er opfyldt.

At de mange uopfyldte krav overhovedet blev formuleret i de tidlige faser af projektarbejdet, var med fuldt overlæg. Gruppen ønskede at arbejde med et stort og realistisk projekt, og ville hellere skære arbejdsopgaver væk undervejs end løbe tør for stof at arbejde med. Der er gennem projektarbejdet løbende lavet bagudgående rettelser i dokumentationen efterhånden som arbejdet skred frem, men gruppen har valgt at fastholde alle krav i projektformuleringen.

En stor del af de krav som ikke er opfyldt, er delvist opfyldt. Det skyldes at gruppen som udgangspunkt har sat ambitionerne højt, og så har vi skåret ned undervejs. Der er fx blevet arbejdet med både lysintensitets- og luftfugtighedssensorer, men komplikationer med disse betød, at de blev droppet sidst i forløbet. Der er dog designet og implementeret SW for sensorerne, så der kræves ikke meget mere arbejde for at kravene vedrørende dette er opfyldt. Af andre eksempler kan nævnes systemlog og plantedatabase, der er delvist implementerede, men af hensyn til tidsplanen ikke blev gjort fuldstændig færdige.

Det implementerede systems største udviklingspotentiale ligger i UART kommunikationen mellem DevKit8000 og PSoC Mater, se 7.9 Resultater og Diskussion på side 34 for nærmere diskussion af dette.

I de sidste faser af projektarbejdet er der i gruppen internt blevet talt en del om, at det nærmest er ærgerligt at forløbet er slut. Der er mange funktionaliteter som kunne færdigimplementeres og/eller optimeres, hvis der havde været mere tid til rådighed.

Gruppen har undervejs været meget tilfreds med projektarbejdets forløb; de erfaringer gruppen gjorde sig på sidste semester har gavnet forløbet, og gruppen har formået at udvikle sig yderligere. Der er ingen tvivl om at der er store fordele ved at arbejde sammen i en "gammel" gruppe, der ikke først skal til at lære hinanden at kende både socialt og fagligt.

Gruppen er meget tilfreds med det realiserede system, men der er bred enighed om at gruppens største styrke ligger i planlægning og koordinering af arbejdet. Der blev - som på sidste semester - lagt en stram tidsplan fra start; der var lagt op til en periode på tre uger til skrivning af denne rapport. Tidsplanen kom - som forventet - til at skride undervejs, men gruppen som helhed har undervejs formået at have overblik over arbejdet og rette i tidsplanen og kravene for projektet. Derved har vi undgået at skulle lave makværk og lappeløsninger i slutningen af forløbet.

## Litteraturliste

- [1] Timll Technic Inc: *DevKit8000 brugermanual*.  
"Bilag 001 - DevKit8000 User Manual En". 2009.
- [2] Cypress Semiconductor: *PSoC 4 Pioneer Kit Guide*.  
"Bilag 002 - CY8CKIT-042 PSoC 4 Pioneer Kit Guide". 2013.
- [3] Honeywell International Inc.: *Datablad for Temperatur-/Luftfugtighedssensor*.  
"Bilag 003 - Datablad for Temp og Luftfugtsensor". August 2013.
- [4] Intersil: *Datablad for lyssensor*.  
"Bilag 005 - Datablad for lyssensor". Februar 2008.
- [5] Motorola, Inc.: *Three-Terminal Positive Voltage Regulators*.  
"Bilag 006 - Datablad for LM7805". Maj 1996.
- [6] Maxim: *Digital Temperature Sensor and Thermal Watchdog with 2-Wire Interface*.  
"Bilag 008 - Datablad for LM75". 2009.
- [7] PRJ3 F15 Gruppe 9: *MSE Øvelse 6*.  
"Bilag 009 - MSE Øvelse 6". Maj 2015.
- [8] Projektdokumentations Review: *Kravspecifikation, Accepttestspezifikation og Tidsplan*.  
"Bilag 010 - Projektdokumentation review 1". 27. Februar 2015.
- [9] Projektdokumentations Review: *Systemarkitektur*.  
"Bilag 011 - Projektdokumentation review 2". 20. Marts 2015.
- [10] PRJ3 F15 Gruppe 9: *Overordnet tidsplan*.  
"Bilag 012 - Tidsplan". Maj 2015.
- [11] PRJ3 F15 Gruppe 9: *Disposition for rapport*.  
"Bilag 013 - Disposition Rapport". Maj 2015.
- [12] IHA: *Vejledning til semesterprojektrapporter*.  
"Bilag 014 - Vejledning for semester projektrapporter 2014". Feb 2014.
- [13] Forskellige forfattere: *I2ISE Compendium*.  
"Bilag 015 - T-006 ISE kompendium". 2012.
- [14] The QT company: *QT dokumentation*.  
<http://doc.qt.io/>. Feb 2015.
- [15] Beelen, Teunis van: *RS232 UART-protokol*.  
<http://www.teuniz.net/RS-232/>. Jan 2015.
- [16] Devkit8000 wiki, IHA: *DevKit8000 wiki*.  
[https://redmine.ihadk.devs/projects/devkit8000/wiki/Addon\\_board](https://redmine.ihadk.devs/projects/devkit8000/wiki/Addon_board). Dec 2013.
- [17] The QT company: *Signals dokumentation*.  
<http://doc.qt.io/qt-4.8/signalsandslots.html>. Feb 2015.

