

# Projekt 3 - Vejrstationen



- **Institution:** Aarhus Universitet
- **Institut:** Institut for Elektro- og Computerteknologi
- **Dato for aflevering:** 17/12-2021
- **Navn på vejleder:** Søren Hansen

## Projektgruppe 8

Navne	Forkortelser	Studieretning	Studie Nummer
Peter Michael Brødsgaard Moore	PMBM	SW	202003680
Mads Holtum Nielsen	MH	SW	202008601
Mathias Schjødt-Bavngaard	MSB	SW	201806662
Rasmus Baunvig Aagaard	RBAA	SW	201510642
Andreas Steen Sørensen	ASS	SW	20042418
Oscar Dennis	OD	SW	202009975
Bahoz Waisi	BW	E	201909362

## Versionshistorik

Version	Dato	Initialer	Ændring
1.0	2-12-2021	MSB	Oprettet dokument og Besluttet opbygning
1.1	9-12-2021	MSB	Indledning skrevet
1.2	10-12-2021	MH	Skrevet opgaveformulering.
1.3	11-12-2021	MH, MSB	Lavet krav-afsnit, Arkitektur
1.4	12-12-2021	OD, RBAA	Lavet systembeskrivelsen, og udviklingsværktøjer. Læringer og design af data modtager(uart)
1.5	13-12-2021	BW	Skrevet lidt til udviklingsværktøjer samt resultat/diskussion-afsnit.
1.6	14-12-2021	MH, BW	Start på designafsnit for vejrviser og vejrmåler
1.7	14-12-2021	ASS, MH	Ændre Fremtidigt arbejde afsnittet, og tilføjet implementering af StepperMotor implementering. Design og implementering.
1.8	15-12-2021	MH, PMBM, ASS, MSB, BW, OD, RBAA	Design og implementering. Lave rettelser
1.9	16-12-2021	MH, PMBM, ASS, MSB, BW, OD, RBAA	Rettelser overalt.
2.0	17-12-2021	MH, PMBM, ASS, MSB, BW, OD, RBAA	Sidste rettelser.

# Resume

Denne rapport beskriver arbejdsprocessen i udvikling af projektet vejrstation.

Vejrstationen er defineret ud fra kravsspecifikationen, som forklarer brugen af systemet. Projektet kræver kun interaktion fra brugeren når systemet skal tændes. Derfra fungerer det automatisk, men har interaktionsmuligheder.

Systemet består af 2 dele, en Vejrmåler og en Vejrviser.

Vejrmåleren er et delsystem, fysisk placeret udendørs, der måler på vejret vha. 4 sensorer. Disse sensorer kommunikerer med en PSoC, der har en trådet forbindelse til en Raspberry Pi, der kan sende vejrdata videre til Vejrviseren over en trådløs forbindelse (WiFi).

Vejrviseren består af en Raspberry Pi med en webserver, der hoster en hjemmeside med visning af vejrdata. Denne kan tilgås over samme trådløse forbindelse som Vejrmåleren benytter. Vejrviseren består også af 4 vejrskiver som afbilder de aktuelle lokale vejrforhold.

Udviklingsprocessen er inspireret af Scrum, således at roller er veldefineret og arbejdet sker i iterative sprints. Den iterative del sker i forhold til en løbende ændring af kravspecifikationen og arkitekturen samt design og implementering. Når der arbejdes iterativt, sikres det, at problemer løses løbende, og at systemet kan optimeres i takt med at man, som udvikler, tilegner sig ny viden.

# Abstract

This report describes the work process of the development of the weather station project. The weather station is based on 6 use cases that explain the usage of the system. The project only demands user interaction to switch on the system. From then on the system is completely automatic, with optional user interaction possibilities.

The system consists of two main parts, a 'Vejrmåler' and a 'Vejrviser'.

The Vejrmåler is a subsystem, physically placed outside, that measures the weather through 4 sensors. These sensors communicate with a PSoC, which has a wired connection to a Raspberry Pi that can pass weatherinformation on to the Vejrviser, through a wireless connection (WiFi).

The Vejrviser consists of a Raspberry Pi with a webserver that hosts a website showing the current weatherconditions. The website is available through the same wireless connection that the Vejrmåler uses. The Vejrviser also consists of 4 weathermeters that similarly show the current weatherconditions

The process is inspired by Scrum, so that roles are well defined, and the work is done in iterative sprints. The iterative part takes place in relation to an ongoing change in the requirements specification, architecture, design, and implementation. By working iteratively, we ensure that problems can be solved dynamically and that the system can be optimized while the developer acquires new information.

# Ansvarsfordeling

X = Hovedansvar

(X) = Delansvar

Afsnit	Ansvarsområder	Mathias	Mads	Bahoz	Rasmus	Oscar	Andreas	Peter
	Resume	X						(X)
	Abstract	(X)						X
1	Indledning	X						(X)
2	Opgaveformulering		X					
3	Projektafgrænsning				X			
4	Systembeskrivelse					X		
5	Krav		X					
6	Projekt proces							X
7	Systemarkitektur	X						
8	Design og Implementering							
8.1	Subsystem vejrmåler	X			X			
8.1.1	Dataindsamler	(X)		X		X		
8.1.2	Datasender	X			X			
8.2	Subsystem Vejrviser	(X)	X					
8.2.1	Klassediagram		X					
8.2.2	Sekvensdiagram		X					
8.2.3	Datamodtager	X						
8.2.4	Database		X					(X)
8.2.5	Hjemmeside							X
8.2.6	Vejrskiver						X	
8.2.7	MinMax							X
8.2.8	Integrationstests af klasserne		(X)				(X)	(X)
9	Resultater og diskussion			X				
10	Udviklingsværktøjer		(X)	(X)		(X)	(X)	(X)
11	Nye erfaringer		(X)	(X)	X	(X)		(X)
12	Fremtidigt arbejde						X	
13	Konklusion			X				
14	Litteraturliste	(X)	(X)	(X)	(X)	(X)	(X)	(X)
15	Bilagsliste	(X)	(X)	(X)	(X)	(X)	(X)	(X)

# Ordforklaringer

Betegnelse	Beskrivelse
Systemet	Består af to dele. En udendørs vejrmåler og en indendørs vejrvise.
Vejrvise	Indbefatter det indendørs system, som skal vise vejrdata til brugeren. Det drejer sig om en CPU med tilhørende webserver, samt fire steppemotorer/vejrskiver, der bruger fysiske visere til at vise vejrdata til brugeren. Derudover er der tilknyttet to knapper til at vise min- og max-værdier.
Vejrmåler	Indbefatter det udendørs system, der skal måle vejrdata og sende det til vejrviseren. Vejrmåleren består af en dataindsamler, der samler data fra fire sensorer, samt en datasender, som modtager data fra dataindsamleren og sender det videre til Vejrviseren.
Vejrskiver	Det er fysiske urskiver, der viser forskellige vejrforhold. Hver vejrskive har en viser, der peger på den måle-værdi, som er sendt fra vejrmåleren.
RPI	En forkortelse for "Raspberry Pi Zero W"
MinMax	Modulet der står for håndtering af minimum og maksimum knapperne. Knapperne bør ændre visning af data på vejrskiverne til de henholdvist laveste og højeste målte værdier for det seneste døgn.
TCP	Transmission Control Protocol – En standard kommunikations protokol til internettet.
telnet	Pre installeret program der fungerer som client ved at skrive ip og portnummer, herefter sendes beskeder der skrives i terminalen til den connectede server.
BDD	Block Definition Diagram
IBD	Internal Block Diagram
UC	Use case
streng	Det refereres til en std::string, som er et char array med indbyggede funktionalitet.
UART	(universal asynchronous receiver-transmitter) er en HW kommunikationsprotokol som benyttes til at sende bytes imellem eksempelvis mikrocontrollere.
Boot/Bootload	Med boot forstås som "ved opstart" eller når der tændes. Bootload betyder i den henseende start af en applikation eller et script når systemet startes/tændes.
WLAN	"Wireless Local Area Network." - Er et netværk der gør det muligt for devices at connect og kommunikere over WiFi.
TSL2561	Sensor der gør det muligt at måle lysstyrken i lux
LM75	Temperatur sensor der gør det muligt at måle temperaturen
Tipping bucket	Regnmåler der gør det muligt at måle nedbør
NRG40 Anemometer	Vindmåler der gør det muligt at måle vindstyrken

# Indholdsfortegnelse

1	Indledning .....	7
2	Opgave formulering .....	8
3	Projekt Afgrænsning .....	9
4	Systembeskrivelse .....	10
5	Krav .....	11
5.1	Aktør beskrivelse .....	11
5.2	Use-case beskrivelser .....	12
5.2.1	Use case 1: Tænd systemet .....	12
5.2.2	Use case 2: Indsaml vejrdato .....	12
5.2.3	Use case 3: Opdatér vejrviseers skiver .....	12
5.2.4	Use case 4: Opdatér hjemmeside .....	12
5.2.5	Use case 5: Skift format for temperaturdata .....	12
5.2.6	Use case 6: Aflæs max/min værdier .....	12
6	Projekt process .....	13
6.1	Ase-modellen .....	13
6.2	V-modellen .....	14
6.3	Scrum .....	15
6.3.1	Product owner og Scrum master .....	15
6.3.2	Scrum board .....	15
6.3.3	Den iterative proces .....	16
6.4	SysML .....	17
6.5	Projektstyring .....	18
7	Systemarkitektur .....	19
7.1	Blokidentifikation af Vejrmåler .....	19
7.1.1	Allokering af Blokke i Vejrmåler .....	20
7.2	Blokindikation af Vejrvise .....	20
7.2.1	Blokbeskrivelse for Vejrvise .....	21
7.2.2	Allokering af Blokke i Vejrvise .....	21
8	Design og Implementering .....	22
8.1	Subsystem Vejrmåler .....	23
8.1.1	Dataindsamler .....	23

8.1.2	Datasender Modul .....	31
8.2	Subsystem Vejrviser.....	36
8.2.1	Klassediagram .....	37
8.2.2	Sekvensdiagram.....	38
8.2.3	Datamodtager.....	39
8.2.4	Database .....	41
8.2.5	Hjemmeside .....	43
8.2.6	Vejrskiver .....	44
8.2.7	MinMax .....	47
8.2.8	Integrationstests af klasserne .....	48
9	Resultater og diskussion .....	49
10	Udviklingsværktøjer .....	51
11	Nye erfaringer .....	52
12	Fremtidigt Arbejde .....	53
13	Konklusion.....	54
14	Litteraturliste .....	55
15	Bilagsliste .....	56

# 1 INDLEDNING

---

Som forældre er der utroligt mange udfordringer og problemstillinger som melder sig på en almindelig og travl hverdag. Man bliver nødt til at planlægge sin tid ordentligt så man ikke bruger hele dagen på logistiske og praktiske opgaver.

Når man om morgenen står op og skal forberede sig til den nye dag, er det vigtigt at vide, hvordan det lokale vejr ser ud, så man kan få sendt sine børn afsted i institution med den rigtige påklædning. Ofte vil man være nødt til at tjekke en hjemmeside, som [vejret.tv2.dk](http://vejret.tv2.dk), for at kunne få overblik over vejsituationen. Det tager tid, og problemet er, at disse hjemmesider viser vejret indenfor et område på en kvadratkilometer<sup>1</sup>. Det kan man knapt nok kalde lokalt.

For at imødekomme dette problem har projektet til formål at udvikle et produkt, som gør det nemt for brugeren at få indblik i vejsituation med det samme, helt lokalt. Produktet vil gøre dette muligt via en hjemmeside og en række vejrskiver, som vil fortælle om vejrforholdene lokalt, hvor brugeren er. Brugeren vil dermed lynhurtigt kunne danne et overblik af vejsituationen på den specifikke placering, ved at besøge hjemmesiden eller kaste et enkelt blik på vejrskiverne og hermed være klar over, hvilket tøjvalg der er optimalt til dagen.

På vejrskiverne vil der være en lille viser der indikerer, hvor meget det regner, hvor meget det blæser, hvor lyst der er og hvad temperaturen er. Brugeren vil desuden kunne interagere med systemet via knapper, som ændrer visningen på vejrskiverne til de højeste eller laveste målinger for det seneste døgn. Dette kan eksempelvis stille brugerens nysgerrighed i forhold til nattefrost eller stormvejr. Både visernes positioner og hjemmesiden bliver opdateret én gang i minuttet, så der altid er dugfrisk data til brugeren.

For at produktet skal kunne alle disse ting er der opstillet en række use cases, som skal beskrive systems funktioner og interne interaktioner. Disse use cases og andre opstillede krav skal løses af systemets moduler. Disse moduler udgør arkitekturen af systemet, hvor der i designet og implementeringen udvikles det færdige produkt.

---

<sup>1</sup> (Deutscher Wetterdienst, 2021) (Deutscher Wetterdienst, 2021)



## 2 OPGAVE FORMULERING

---

Denne opgave handler om at designe en vejrstation, som skal kunne opsættes i brugerens private hjem. Systemet skal bestå af en indendørs vejrvise og en udendørs vejrmåler. Hver del skal kunne tilsluttes et lokalt WiFi, så delene kan kommunikere. Vejrmåleren skal styres af en Datasender-CPU og en Dataindsamler-CPU. Sidstnævnte skal snakke med en række sensorer, der skal måle den aktuelle vejr-situation og give målingerne til CPU'en. Denne Dataindsamler-CPU skal efterfølgende lave beregninger på vejrdaten, så der tages højde for fejlmålinger og så vejrdaten bliver mere sigende. Dataindsamler-CPU'en skal én gang i minuttet sende den beregnede vejrdato til Datasender-CPU'en, som skal videresende dataene til vejrvise indenfor.

Systemets vejrvise skal også bestå af en CPU, der modtager vejrmålinger. Derudover skal den have en webserver, en database og en række tilknyttede vejrskiver. Disse vejrskiver skal fungere som barometre, der afspejler hver sin type af vejrdato. Derudover skal CPU'en have tilsluttet to fysiske knapper - en min- og en max-knap. Når én af disse trykkes ned, skal vejrskiverne skifte fra at vise den seneste måling til at vise den højeste/laveste måle-værdi for hver vejrtype, som er indsamlet i løbet af det seneste døgn.

Webserveren skal hoste en hjemmeside, der kan vise de seneste vejrdato. På den måde kan brugeren se vejr-situationen, selvom vedkommende ikke er i nærheden af vejrskiverne. Hjemmesiden skal ovenikøbet kunne ændres til at vise temperaturen i fahrenheit frem for celsius.

Databasen skal kunne indeholde vejrmålinger fra det sidste døgn. Blandt disse målinger skal der kunne findes min- og max-værdier for alle måle-typer, som kan sendes til stepper-motorerne, når knapperne trykkes ned. Herudover skal databasen udskrive sine målinger til en ekstern fil én gang i døgnet. Denne fil skal bruges til at føre historik over vejret.

Når vejrvise's CPU modtager vejrdato fra vejrmåleren, skal vejrdaten lægges i databasen, webserveren skal opdateres og det samme skal alle vejrskiverne. Denne ring af begivenheder skal fortsætte, så længe systemet er tændt.

### 3 PROJEKT AFGRÆNSNING

---

Der er givet følgende krav til indhold, metode og proces for dette projekt<sup>2</sup>

- Systemet **skal** via sensorer/aktuatorer interagere med omverdenen
- Systemet **skal** have et brugerinterface
- Systemet **skal** indeholde faglige elementer fra semesterets andre fag
- Systemet **skal** anvende en indlejret Linux platform og en PSoC platform
- Der **skal** anvendes Scrum som arbejdsmetode

Ud fra ovenstående krav til systemet, vil der blive fremstillet en prototype af Vejrstation.

Der tages et forbehold for at enkelte delelementer for prototypen kan implementeres med henblik på at vise ”proof of concept” samt den ønskede funktionalitet og ikke med tanke på scalability eller vedligeholdelse af systemet.

Der er i projektet afgrænset fra at teste systemet under specifikke, relevante og realistiske vejrforhold, som kraftig regn eller i temperatur områder svarende til ekstremiteterne for det dansk vejr. Dette skyldes en manglede adgang til sådanne testfaciliteter, samt et fokus på at udvikle en funktionsdygtig prototype inden for tidsrammen til projektet.

---

<sup>2</sup> (MIKKELSEN, 2020)

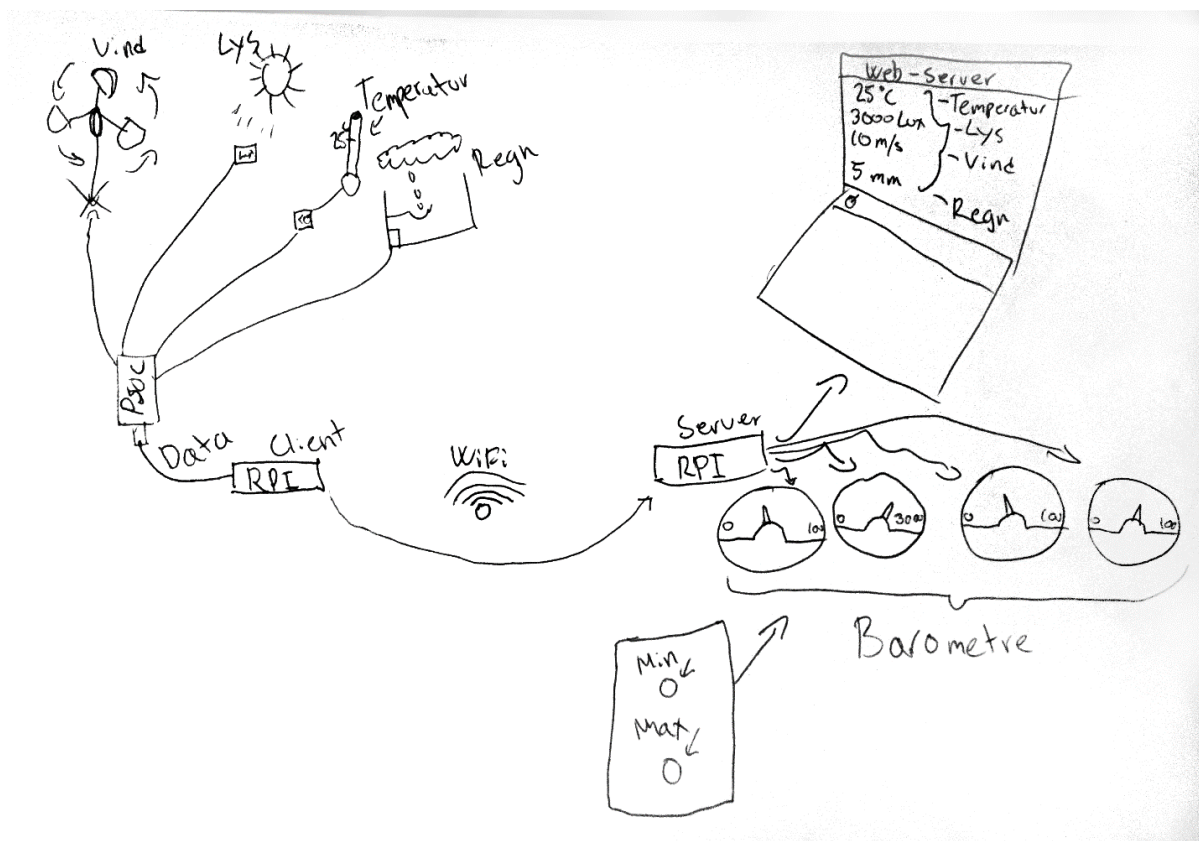
## 4 SYSTEMBESKRIVELSE

I dette projekts samlede problemformulering, som kan findes i bilag<sup>3</sup>, beskrives der en lokal vejrstation for den travle familie. Pga. dette laves et passivt system der kan fungere uden bruger input når det er i gang, men som stadig er i stand til at vise minimum og maksimum værdi for vejr data, hvis bruger ønsker det.

Herfra er der blevet valgt et system som består af en Vejrmåler og en Vejrviser.

Først i systemet er vejrmåleren. Denne vejrmåler består af to led, hvor det første led er en mikrocontroller, som fungerer som vores data indsamler, til måling af det aktuelle vejr vha. de valgte sensorer. Det andet led er en data sender, som modtager dataene.

Derefter er der systemets Vejrviser, som består af server, som modtager data fra data senderen, en database som vil gemme denne data i en .csv fil, fire vejrskiver som vil illustrere dataene til brugeren på en let og overskuelig måde, en minimum værdis knap, en maksimum værdis knap, som vil give brugeren mulighed for at se henholdsvis mindst målt værdi og max målt værdi indenfor det sidste døgn og til sidst en webserver. Denne webserver bruges til at vise det målte data, som er det aktuelle vejr.



Figur 1 : Sketch af samlet system

Brugeren er også i stand til at ændre mellem fortolkning af den målte data såsom; at temperatur vises i fahrenheit i stedet for celsius.

<sup>3</sup> Bilag 1 – Dokumentation\1.1 - Problemformulering

På Figur 1 ses et billede af systemet. Heri ses der vejrmåleren til venstre, som står for indsamlingen af dataene fra de fire sensorer. Dataene sendes videre til en client, der står for at sende dataene til serveren i vejrviseren. Vejrviseren sørger både for data til web-server, database og vejrskiverne.

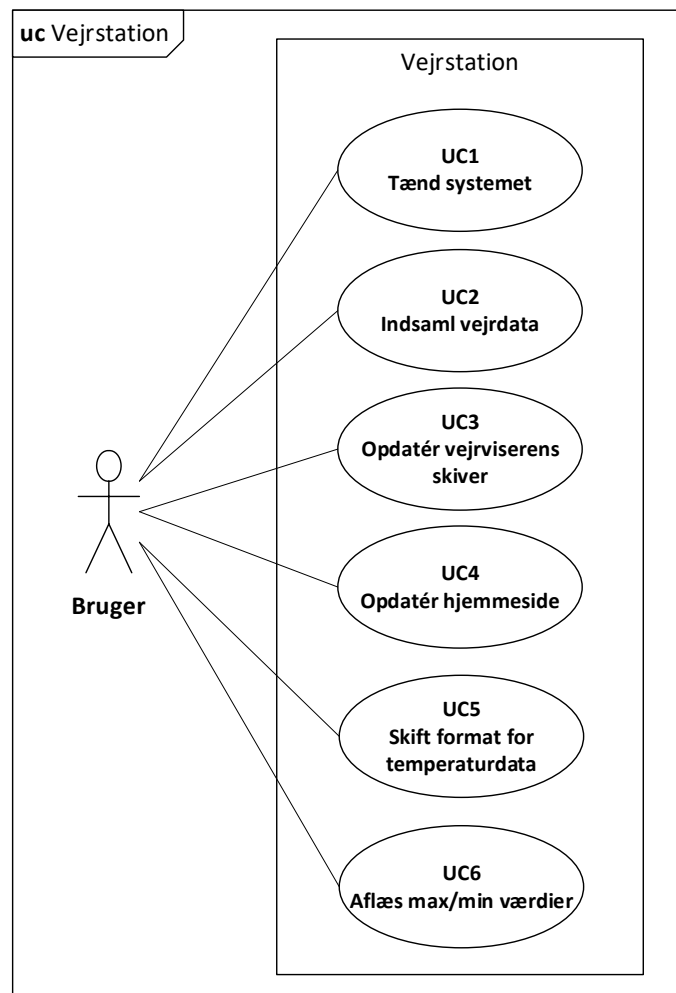
Der ses også nederst på billedet minimum og maksimum knapper, som brugeren kan interagere med for at vise specifikke data, enten den laveste målte værdi i det sidste døgn eller den højeste, på vejrskiverne.

## 5 KRAV

Med udgangspunkt i opgaveformuleringen er der opstillet en række use-cases, nogle funktionelle MoSCoW-krav og nogle ikke-funktionelle MoSCoW-krav, der til sammen udgør kravspecifikationen for projekt vejrstation. Specifikationen sørger for, at systemet kan leve op til opgaveformuleringen. I rapporten er der valgt kun at inkludere use-cases, da de er grundstenen for systemets funktionalitet. De resterende krav kan findes i bilaget<sup>4</sup>.

### 5.1 Aktør beskrivelse

Systemet har kun én aktør, som er den primære aktør, brugeren. Der er dog valgt at beskrive systemet fra en udviklers synspunkt. Grunden hertil er, at man gerne vil se de komplicerede steps, som sker inde i systemet, i disse use-cases. Brugeren initialiserer alle use-cases, som det kan ses i use-case diagrammet (Figur 2).



Figur 2: Use-case diagram for Vejrstation

<sup>4</sup> Bilag 1 – Dokumentation\1.2 – Kravspecifikation\

## 5.2 Use-case beskrivelser

Systemet består af en række use-cases, som kan ses på figuren ovenfor (Figur 2). Hver use-case beskriver en funktion i systemet og er med til at definere kravene for systemets funktionalitet. De seks use-cases kan ses med en "brief" beskrivelse nedenfor. En yderligere beskrivelse af use-cases kan ses i "fully-dressed"-format i bilaget<sup>5</sup>.

### 5.2.1 Use case 1: Tænd systemet

For at brugeren kan bruge systemet, skal brugeren som det første gennemføre UC1, hvor både vejrmåleren og vejrviseren tilknyttes strømforsyning. Herefter starter de forskellige dele af systemet automatisk op. Den eneste synlige aktivitet er, at alle vejrviserens vejrskiver kalibrerer, så de er klar til at vise den aktuelle vejrdato.

### 5.2.2 Use case 2: Indsaml vejrdato

Når UC1 er gennemført, begynder vejrmåleren et loop af indsamlingen af vejrdato. Det sker ved, at alle sensorerne kontinuerligt sender data til dataindsamleren, som laver beregninger på dataen. Hvert minut sendes den beregnede vejrdato til datasenderen, som sender vejrdatoen videre til vejrviseren via WiFi.

### 5.2.3 Use case 3: Opdatér vejrviserens skiver

Når UC1 er gennemført starter vejrviser ligeledes et uendeligt loop. Dette starter med, at vejrviseren venter på, at der modtages ny vejrdato fra vejrmåleren. Derefter omregnes den modtagne vejrdato til placeringsvariabler for vejrskiverne, som efterfølgende opdateres med disse.

### 5.2.4 Use case 4: Opdatér hjemmeside

Hver gang der modtages ny vejrdato hos vejrviseren, skal hjemmesiden ligeledes opdateres med den nye data. Herefter gemmer vejrviseren vejrdatoene i Databasen.

### 5.2.5 Use case 5: Skift format for temperaturdata

Brugeren skal have mulighed for at vælge mellem at se temperaturen på hjemmesiden som celsius og fahrenheit. Det opnås ved at have en "Skift-format-knap" på hjemmesiden. Når brugeren har hjemmesiden åben og trykker på knappen, skifter hjemmesiden visning af temperatur-formatet mellem celsius og fahrenheit.

### 5.2.6 Use case 6: Aflæs max/min værdier

Hvis brugeren ønsker at se max- og min-værdier for det seneste døgns vejrdato, skal brugeren trykke på enten max- eller min-knappen på vejrviseren. Vejrviseren detekterer dette knap-tryk, hvorefter databasen finder de højeste/laveste måleværdier for alle måletyper blandt det sidste døgns målinger. Vejrviser opdaterer herefter vejrskiverne med max-/min-værdierne. Når knappen atter slippes, opdateres vejrskiverne til igen at vise det aktuelle vejr.

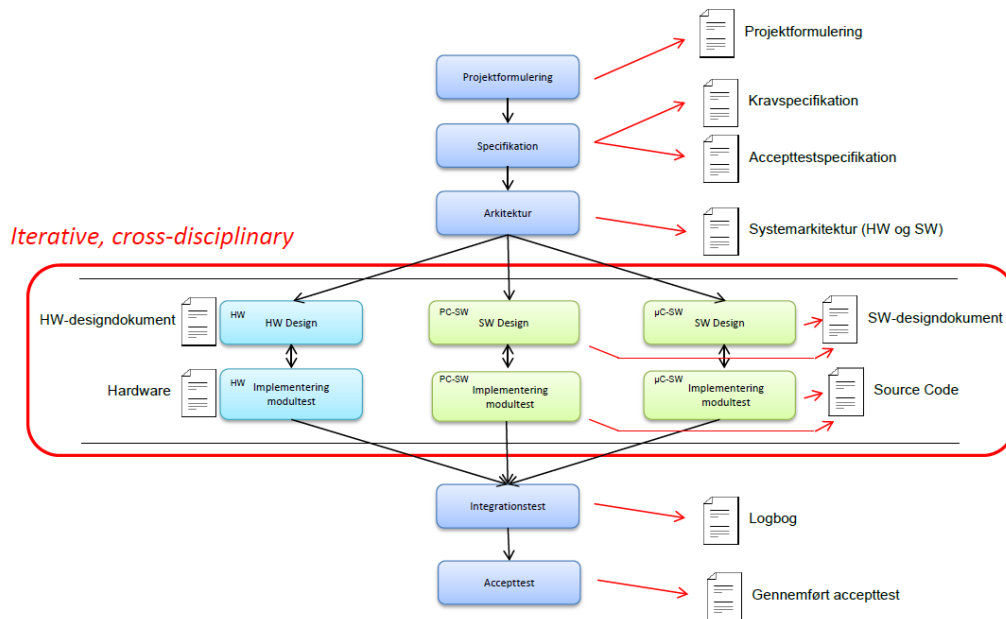
---

<sup>5</sup> Bilag \1 – Dokumentation\1.2 – Kravspecifikation\Funktionelle\_krav.pdf

## 6 PROJEKT PROCESS

### 6.1 Ase-modellen

Den overordnede arbejdsproces på dette semesterprojekt er inspireret af ASE-modellen<sup>6</sup> (se Figur 3), men med en mere iterativ tilgang. ASE-modellen forblev den overordnede retningslinje for de elementer der skulle indgå i processen, men de enkelte dele af modellen blev samlet op og modificeret gentagende igennem hele projektets forløb.



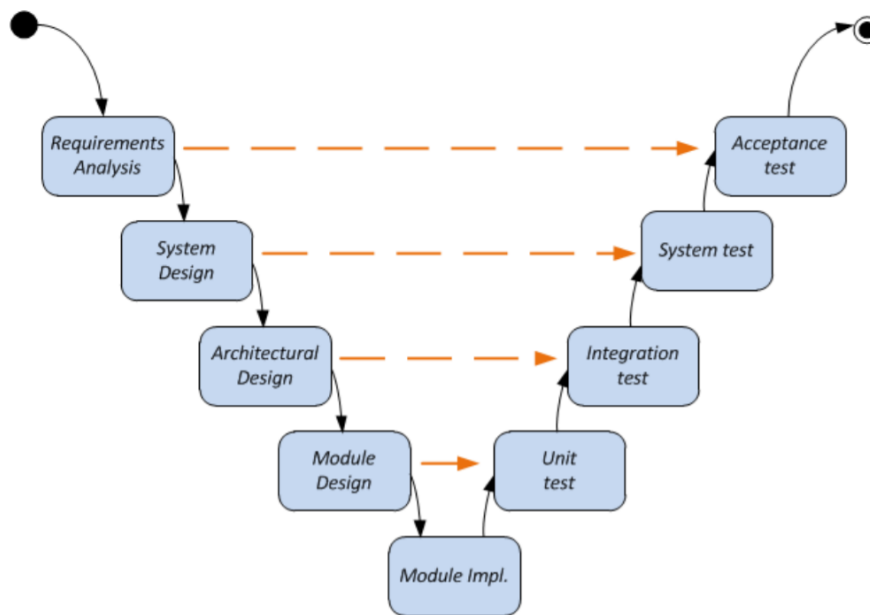
Figur 3: ASE-modellen (Bjerge, 2015)

ASE-modellen er efterhånden en velkendt model, der gør hele opstartsprocessen for et projekt meget strømlinet. Så snart gruppen fik kendskab til de helt overordnede krav for projektet, blev der taget hul på de første dele af modellen næsten per automatik. I de senere faser, flyder det dog mere sammen. Det er ikke altid intuitivt at lave helt skarpe opdelinger af design, implementering og test, da disse naturligt vil påvirke hinanden. Her blev kombinationen af ASE-modellen og den iterative proces meget naturlig. Dette beskrives yderligere i Scrum-afsnittet.

<sup>6</sup> ASE-modellen (Bjerge, 2015)

## 6.2 V-modellen

I arbejdsprocessen er der taget inspiration fra V-modellen<sup>7</sup> (se Figur 4) idet der altid har været fokus på testbarhed af de dele der bliver udviklet. Fra begyndelsen af projektet blev der bestemt krav, og på samme tid accepttestspecifikationer, som kunne afsløre om de krav der blev opsat, egentligt var testbare. Det samme gjorde sig gældende i de resterende faser af projektet, således at man, sideløbende med eksempelvis design og implementering af selve systemet, definerer tests, der kan verificere funktionaliteten af disse elementer.



Figur 4: V-modellen (Bjerger, 2015)

<sup>7</sup> V-modellen (Bjerger, 2015)

## 6.3 Scrum

Til koordinering af projektarbejdet blev der benyttet en Scrum-inspireret tilgang, så gruppen kunne påtage en iterativ arbejdsmetode. Hele projektet blev opdelt i 6 sprints, som hver havde en varighed på 1-3 uger. Ambitionen var, at dette skulle give overblik over arbejdsbyrden for de enkelte sprints og sikre, at alle krav til projektet blev indfriet til tiden. Ved et afsluttet sprint fulgte et sprint-retrospective, hvor gruppemedlemmerne fik mulighed for at kommentere på sprintets opbygning og fokus, samt de problemer de selv måtte have haft undervejs. Dermed bidrog gruppemedlemmerne til forbedring af arbejdsprocessen i de kommende sprint.

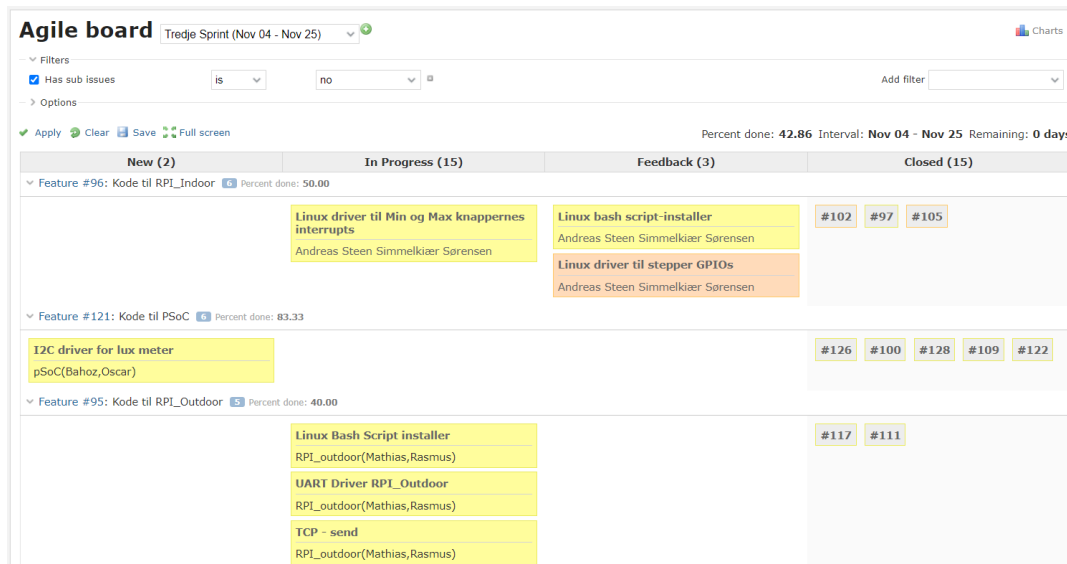
### 6.3.1 Product owner og Scrum master

Vores variant af Scrum var uden *product owner*, hvilket tillod os i fællesskab at redefinere krav i løbet af projektet. Dette var på godt og ondt. Der var stor fleksibilitet i processen, men det kostede ofte på klarheden, da ændringer i projektets krav ikke blev drøftet på lige så formel vis, som hvis man stod til ansvar for en product owner.

Grundet manglende product owner blev *scrum masterens* rolle reduceret til en form for tovholder, hvis primære opgaver bestod i at sikre, at deadlines blev holdt, at møder var strukturerede og at sprint opgaver havde en klar 'definition of done'.

### 6.3.2 Scrum board

Der blev benyttet et scrum board (se Figur 5) på Redmine hvor de enkelte gruppemedlemmer kunne påtage sig opgaver fra sprint-backloggen, som blev opdateret ved starten af hvert sprint med prioriteringer og skarpe deadlines.



Figur 5: Scrum board på Redmine

Ved udarbejdelse af disse opgaver blev det hurtigt tydeligt, hvor svært det er at definere opgaver der har en klar 'definition of done'. Opgaverne blev ofte så overordnede at de forblev i 'In Progress' fanen i flere sprints, fordi afslutningskriterierne ikke var veldefinerede, og fordi der blev arbejdet iterativt.



### 6.3.3 Den iterative proces

For hvert afsluttet sprint var intentionen, at der skulle være nået en afslutning på en mindre del af produktet, som kunne præsenteres som et 'færdigt' stykke arbejde. Dette viste sig dog at være et svært krav at efterleve, da der var enorm tidsbegrænsning på arbejdsindsatsen for det enkelte medlem, grundet pres fra sideløbende kurser. Vi pådrog os dog det dogme, at hver eneste gang der blev oprettet et nyt dokument, uanset fase, så var dette dokument en alpha. Denne definition lod os gøre mindre ud af formalia, og i stedet få indholdet på plads hurtigt, så der kunne itereres og finpudses i et senere sprint.

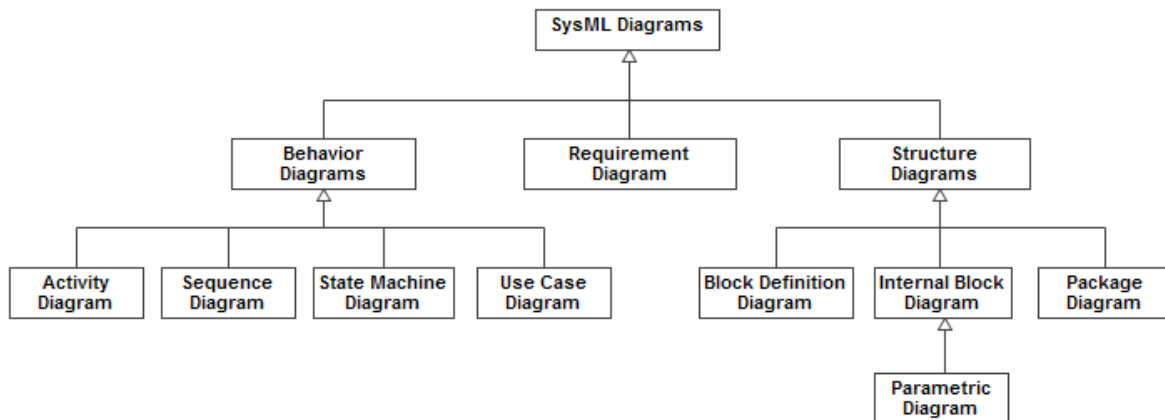
Sprint #4 var det mest ideelle sprint, da hvert medlem fik udpeget en specifik del af implementeringen de skulle få til at virke. Det blev udspecificeret, at det skulle være en løsning der virkede, fremfor en løsning der var effektiv. I dette sprint nåede næsten alle medlemmerne i mål med en fungerende implementering af deres individuelle dele, som ikke var 'pæn', men som var præsentabel og vigtigst af alt; den kunne itereres på. Det var en tilfredsstillende arbejdsproces da man kunne lægge perfektionismen på hylden og mere effektivt bevæge sig mod sit mål. Se bilag<sup>8</sup> for yderligere beskrivelse af brugen af Scrum.

---

<sup>8</sup> Bilag 2 – Organisatorisk\2.5 - Procesbeskrivelse\Procesbeskrivelse.pdf

## 6.4 SysML

SysML<sup>9</sup> blev valgt som modelling language da det supplerer veldefinerede standardmetoder til grafisk afbildning af systemer med både software- og hardware elementer. Modelsproget giver et strømlinet overblik for udvikleren og gør formidling til tredjeparter intuitivt og nemt. I projektet blev der gjort brug af Structure Diagrams og Behavior Diagrams som ses på Figur 6.



Figur 6: SysML diagram-klasser<sup>10</sup>

Modelsproget blev benyttet bredt i kravspecifikation-, arkitektur- og design-faserne i projektet til at gøre problemstillingerne tydelige for gruppen.

I kravsspecifikationen bruges SysML til at lave *use case diagrammet*, og i arkitekturen blev SysML anvendt til at definere den strukturelle opbygning af de enkelte hardware-blokke i *BDD*, og de interne signaler mellem blokkene med *IBD*. Derudover blev der i arkitekturen benyttet SysML til beskrivelse af den adfærdsmæssige del af systemet ved hjælp af *sekvensdiagrammer*.

I overgangen mellem arkitektur og design benyttedes *domænemodeller* til at give et overblik over relationerne mellem de enkelte dele af systemet, og yderligere specificere hvilke samspil software og hardware har i systemet. Disse tager udgangspunkt i use case beskrivelserne, og danner en fælles forståelse for navngivning i systemet, hvilket sikrer den røde tråd i projektet.

Ved udarbejdelse af design for software blev der lavet *applikationsmodeller* i form af sekvensdiagrammer og klassediagrammer. Disse fungerer som et overblik af både Vejrviseurs og Vejrmålers implementering. Diagrammerne bygger på sekvensdiagrammerne fra arkitekturen og sammendrager dem til to handlingsforløb for henholdsvis Vejrmålers og Vejrviseurs CPU'er.

<sup>9</sup> A Practical Guide to SysML (Friedenthal, Moore, & Steiner, 2008)

<sup>10</sup> (Inc., 2021)

## 6.5 Projektstyring

I begyndelsen af projektet definerede gruppen i fællesskab en samarbejdsaftale<sup>11</sup> der lå til grund for samarbejdet i løbet af semestret. Eftersom gruppen ikke blev sammensat baseret på personlighedsprofiler, var farvesammensætningen meget til den introverte side (blå/grøn), med en enkelt gul. Da der ikke var en oplagt rød leder, meldte Andreas og Peter sig som mødeledere med skifteordning i løbet af projektet, med Mathias som fast referent. Der blev holdt ugentlige projektgruppemøder og ugentlige scrummøder på cirka 10 minutters varighed til hurtig statusopdatering. Der blev desuden holdt møde med vejleder cirka hver anden uge, skubbet efter behov.

Undervejs i projektet blev der gjort brug af en tidsplan<sup>12</sup> som består af et Gannt inspireret diagram, til at holde overblik, og en mere detaljerig Projektplan, til grundig beskrivelse af de enkelte sprints. Eftersom der blev benyttet Scrum til projektstyringen blev risikovurderinger naturligt foretaget ved hver sprintstart. De nyligt definerede opgaver til sprint-backloggen blev sammenholdt med tidsplanen og givet en tilsvarende prioritet, i form af en farve. Herfra kunne arbejdsindsatsen for det enkelte medlem fokuseres på de vigtigste opgaver.

Scrum boardet blev benyttet hyppigt i begyndelsen af projektet og var en hjælp til at skabe overskuelige opgaver og danne overblik. Som gruppen gik ind i de sidste to sprints, blev de gældende opgavestillinger dog modificeret så ofte, at det blev en praktisk forhindring konstant at skulle opdatere scrumboardet i stedet for blot at overholde mundtlige aftaler. Brugen af Scrum blev dog generelt set som svær at tilpasse til en arbejdsuge med kun cirka 10 timer afsat til projektet. Når arbejdstiden ligger så sporadisk, og meget af tiden går med at huske hvad der blev lavet i ugen forinden, taber sprintet noget af sit formål. Pointen er, at scrum medlemmerne bør kunne dedikere sig 100% til deres specifikke opgave, så opgaven slet ikke slippes i de 2-4 uger sprintet varer. Manglen på tid til dedikation på denne måde var en mærkbar svaghed ved brugen af Scrum i dette projekt.

---

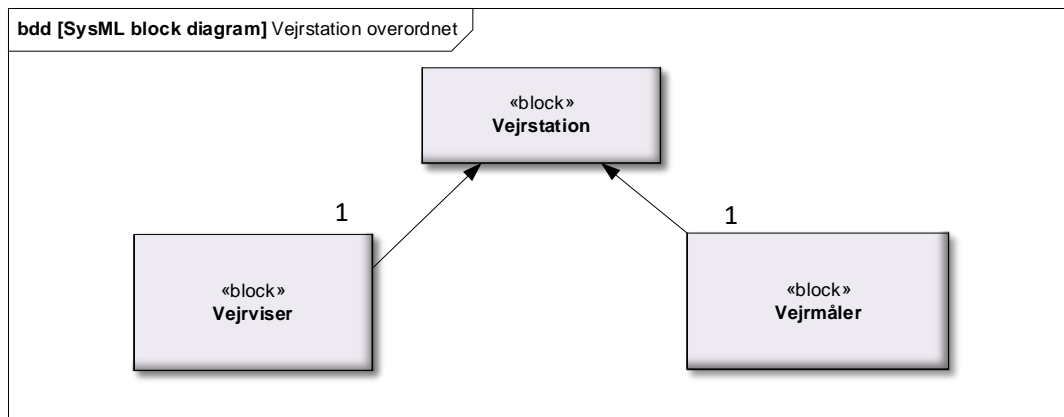
<sup>11</sup> Bilag 2 – Organisatorisk\2.6 - Samarbejdsaftale\Samarbejdsaftale.pdf

<sup>12</sup> Bilag 2 – Organisatorisk\2.1 - Tidsplan\Tidsplan.pdf

## 7 SYSTEMARKITEKTUR

I systemarkitekturen beskrives opbygningen af systemet. Denne arkitektur vil danne base for designet og implementeringen af systemet. I arkitekturen kigges der på subsystemer og moduler ud fra funktionalitet og der skældes ikke mellem software og Hardware.

Ud fra use cases, funktionelle krav og Ikke funktionelle krav er systemet opdelt i to overordnede moduler, Vejrviseren og Vejrmåleren, hvilket kan ses på nedenstående BDD.



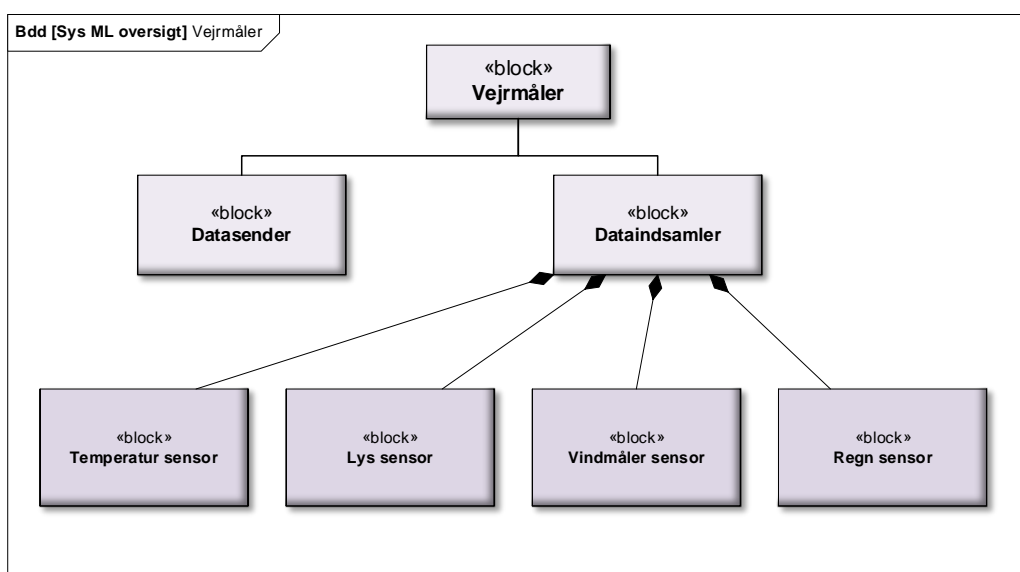
Figur 7 Overordnet block diagram over systemet hvor man kan se Subsystemerne

Vejrmåleren skal som navnet fortæller måle det lokale vejr ved hjælp af sensorer.

Vejrviseren skal vise brugeren det aktuelle lokale vejr ved hjælp af aktuatorer.

### 7.1 Blokidentifikation af Vejrmåler

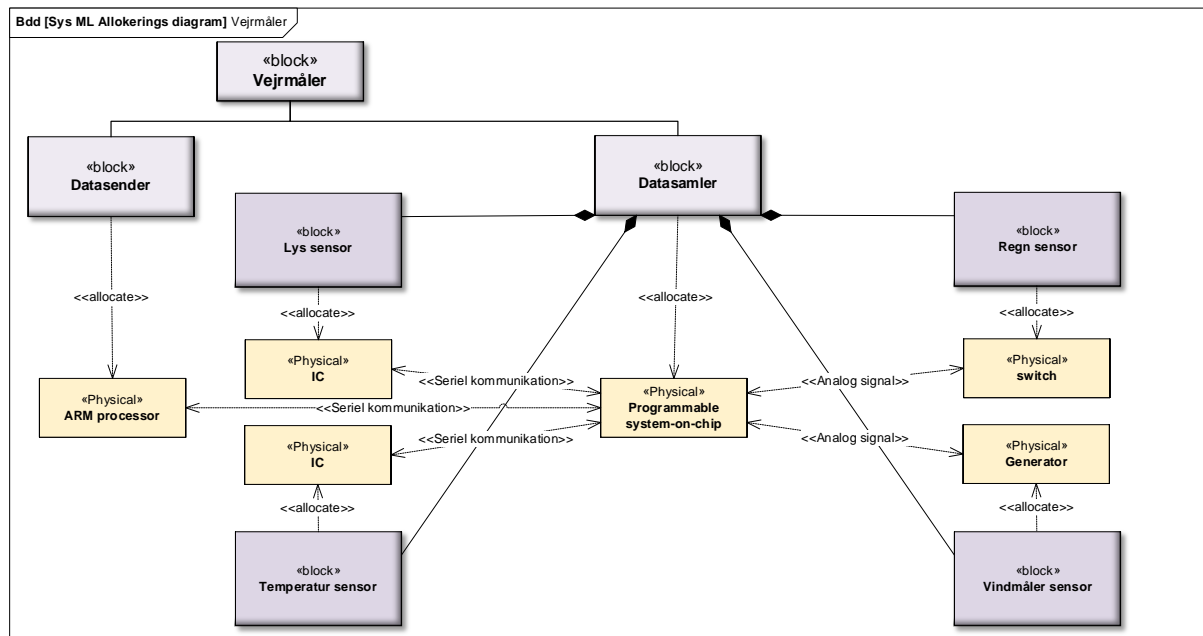
Dybere i arkitekturen findes blokdiagrammet for Vejrmåleren. Den består af to dele, en Datasender og en Dataindsamler. Målingen af vejret sker gennem dataindsamleren og det er her sensorerne sidder forbundet til systemet.



Figur 8 BDD blokke af funktioner i Vejrmåleren

### 7.1.1 Allokering af Blokke i Vejrmåler

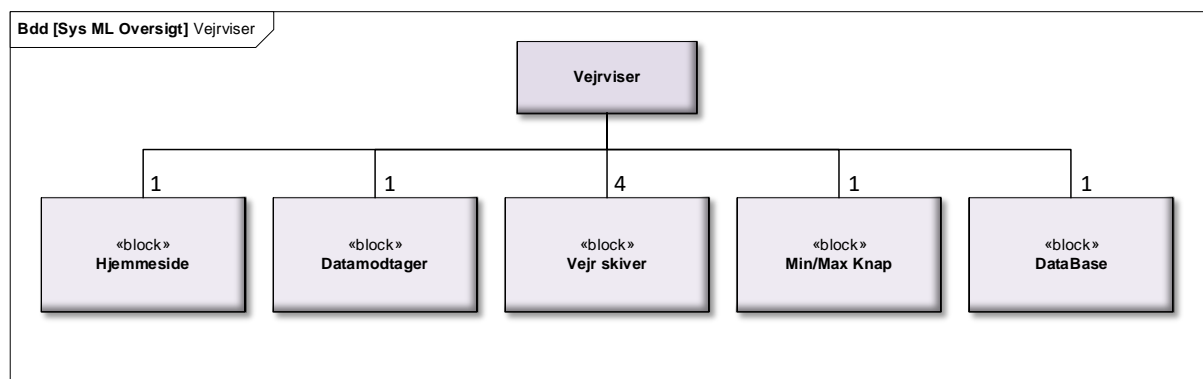
Som det kan ses på diagrammet (Figur 9) er Dataindsamleren allokeret på en "Programmable system-on-chip". Dette er med formålet, at det er nemmere at producere et godt interface til de fysiske sensorer. Datasender er valgt til at ligge på en ARM processor, da det er nemmere at sende dataene trådløst.



Figur 9 BDD allokering af blokke i Vejr måleren - udarbejdet i visio

## 7.2 Blokindikation af Vejrviser

Vejrviseren består af flere dele end vejr måleren. Den indeholder funktioner som skal vise brugeren de lokale Vejrdata. Disse funktioner ligger i blokkene hjemmeside og Vejrskiver. For at få de lokale Vejrdata hen til disse blokke har Vejrviseren en datamodtager der skaber mulighed for at modtage data fra vejr måleren og en Database som skal gemme disse data. Den sidste blok er en Min/Max knap som skal vise brugeren de sidste 24 timers ydre værdier.



Figur 10 Overblik over blokke i vejrviseren

### 7.2.1 Blokbeskrivelse for Vejrviser

Den første del som arbejder med dataene er **datamodtageren**, som skal modtage data fra Vejr måleren via en trådløs forbindelse. og gøre dem tilgængelig for de andre klasser.

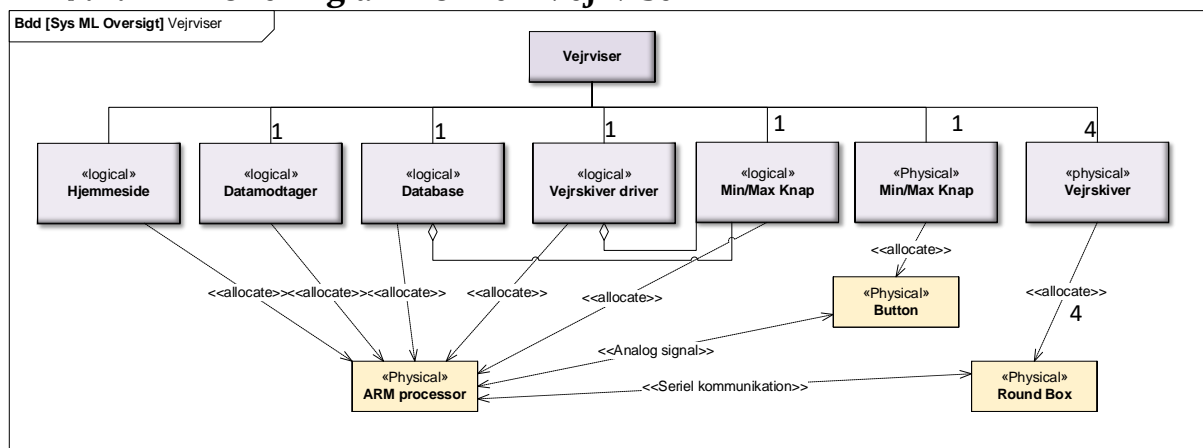
**Databasen** står til opgave at gemme de data, som serveren modtager, og den skal gemme data både dynamisk og statisk.

De gemte data skal **Min/Max knap** bruge, fordi den skal ændre det som vises på Vejrskiver til at vise de yderværdier, der er målt inden for det seneste døgn.

På **Hjemmeside** der kan tilgås ved at bruge en almindelig browser (f.eks. chrome / firefox) på hjemmesiden skal man kunne se de senest målte data.

Den sidste blok, som skal afbilde vejrdatene, er **Vejrskiverne** som har en viser der peger på de nuværende vejr forholdsværdier.

### 7.2.2 Allokering af Blokke i Vejrviser

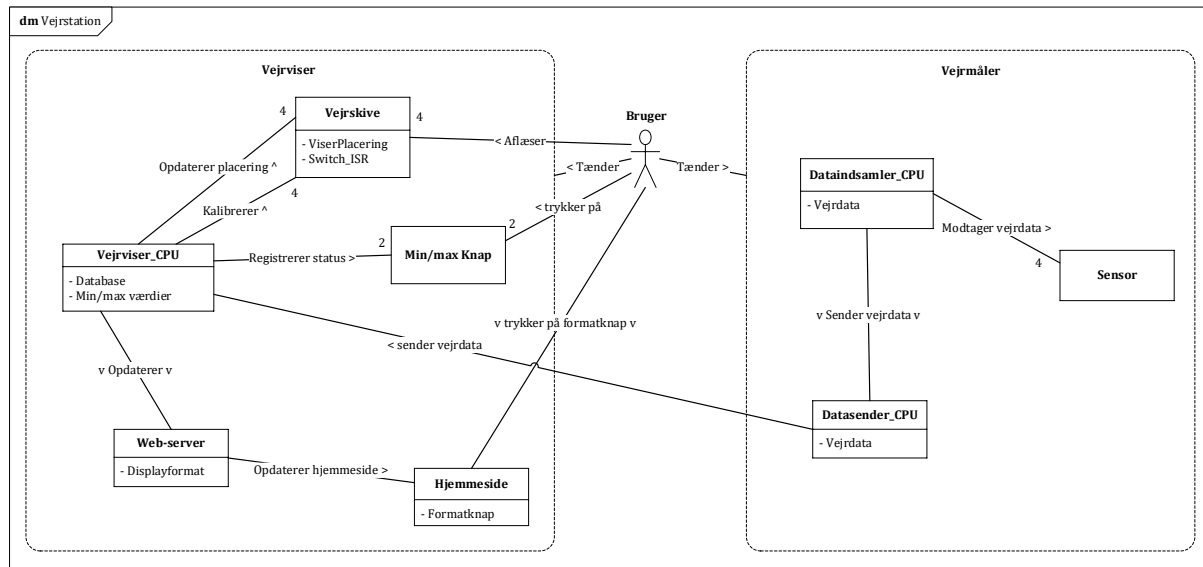


Figur 11 Allokerings diagram over vejrviser

Logikken i vejrviseren ligger alt sammen på en armprocessor. Vejrskiverne har logik(styring) på armprocessoren men har også nogle fysiske aktuatorer, som ligger inde i en "black box" kaldet Round box fordi det skal være runde urskiver. Min og Max knappen bliver ligesom vejrskiver styret af Arm Processoren men er også fysisk i form af en knap.

## 8 DESIGN OG IMPLEMENTERING

I dette afsnit kigges der på Designet af systemets moduler. Med udgangspunkt i use cases er der først udarbejdet en domæne-analyse. Det har resulteret i følgende domænemodel (Figur 12). Her ses det, hvordan systemet er opdelt i de to blokke, Vejrviser og Vejrmåler, og hvordan de interne blokke relaterer til hinanden, samt hvilke der interagerer med brugeren.

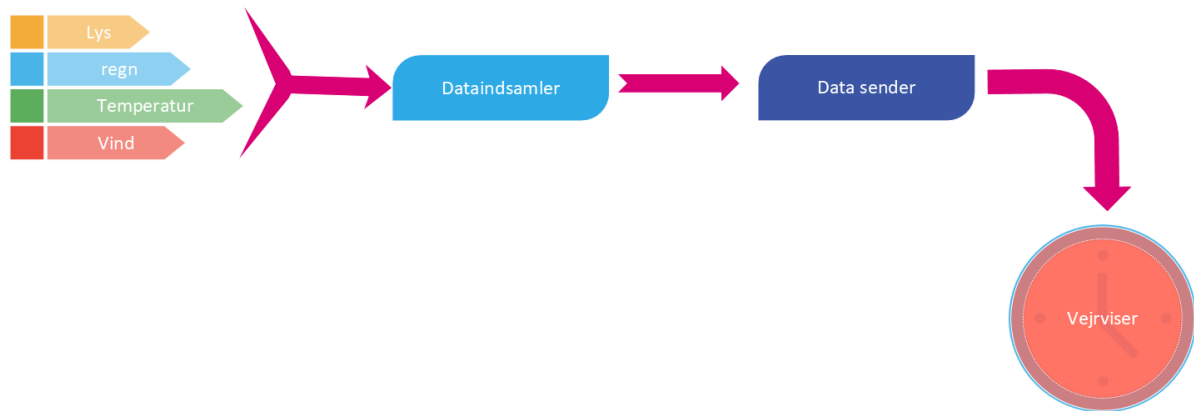


Figur 12: Domænemodel for vejrstationen

I afsnittet beskrives design og implementering for hvert modul nævnt i arkitekturen. Modulerne beskrives ud fra deres funktionalitet som en samlet pakke og kan altså både bestå af software og hardware.

## 8.1 Subsystem Vejmåler

Vejrmåleren består af to dele, en dataindsamler til at samle data, fra sensorer, og en datasender som skal flytte denne data videre til Vejrviseren.



Figur 13 Oversigt over Vejmåler

### 8.1.1 Dataindsamler

Der er valgt at bruge en PSoC25 til at samle dataene. Grunden til dette valg af PSoC er den allerede opbyggede viden og genkendelighed med denne mikrocontroller, som blev skaffet under tidligere arbejde i bl.a. kurset GFV. Heri blev der specifikt arbejdet med indsamling af data fra en sensor via I2C kommunikation. Udover dette var der også stillet et projektkrav om at anvende en PSoC i systemet.

Til udvikling af PSoC programmet til dette system, tages der udgangspunkt i kravspecifikationerne. Her er der sat nogle grundlæggende krav som systemet skulle kunne overholde, som at måle data fra fire sensorer, og derefter sende dem via en trådet forbindelse, hvor der vælges at anvende UART.

Eftersom at PSoC'en er ansvarlig for alt dataindsamling, så er den det første led i det samlede system. Derfor er det vigtigt, at der blev sendt korrekte data værdier, og dermed ingen fejlagtige spikes i programmets vejrmåling. Dette løses ved at sende en gennemsnitsværdi for lys og regn data, sådan at en enkel spike ikke vil give en måling der ikke giver mening. Der skal ikke tages gennemsnit af data fra vindmåler og regnmåler, da de skal akkumulere data.

Systemet skal også være i stand til at måle vejrdato i 60 sekunder, og derefter sende dataene videre til RPI\_Outdoor ved brug af UART-kommunikation.

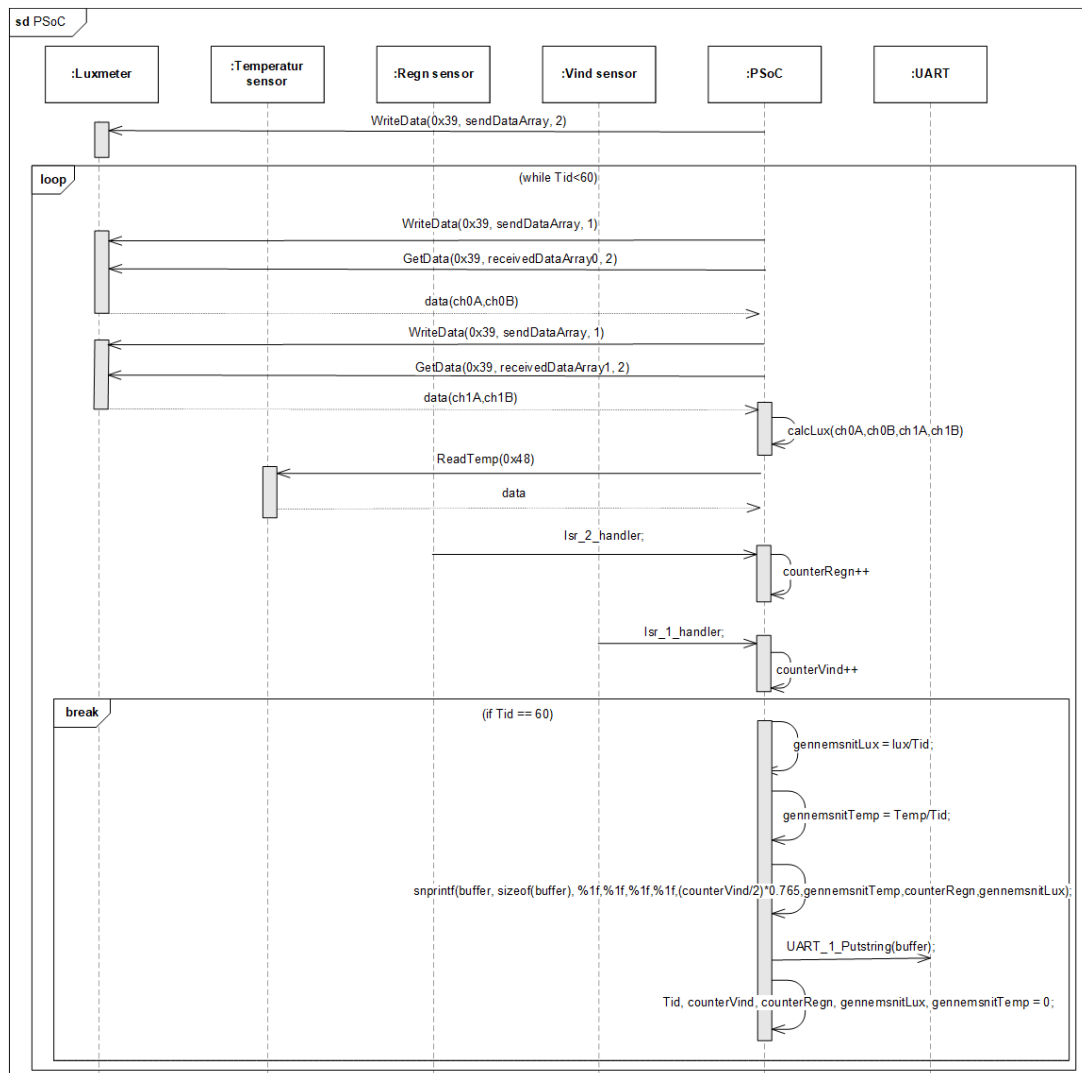


## Sekvensdiagram for PSoC

For at vise hvordan programmet skal samle dataene er der blevet oprettet et sekvensdiagram for PSoC, som der kan ses på Figur 14. Her ses I2C kommunikation mellem PSoC og Luxmeter, og PSoC og Temperatur sensor, samt signaler, som udløser systemets interrupts, der kommer til PSoC fra vind- og regn sensor. Videre ses der også i slutningen af diagrammet at dataene samles i en buffer og sendes ud via. UART.

For yderligere uddybelse af kommunikationen mellem PSoC og dens sensorer, henvises der til bilag<sup>13</sup> implementeringsdokumentet for PSoC.

For ekstra information vedr. diagram henvises til bilag<sup>14</sup>.



Figur 14 PSoC sekvensdiagram

<sup>13</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\1.4.1 – dataindsamler\implementation og design Psoc.pdf

<sup>14</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\Vejrmåler\_design.pdf - kap. 2.5

#### 8.1.1.1 Lys

For at måle lysstyrken er der valgt at bruge et luxmeter. Hertil er der af krav at den skal kunne måle lys, og gøre det mindst en gang i sekundet.

Ud fra disse krav blev der valgt en TSL2561. Denne blev valgt, da der kan læses data fra den op til hvert 13,7 ms<sup>15</sup>, så der kan nemt blive sendt data hvert sekund til PSoC. Ved anvendelsen af denne sensor kan der opfyldes krav om at system skal kunne måle lys, og at systemet skal kunne måle fra en lyssensor mindst en gang i sekundet.

For at få en TSL2561 til at virke skal der anvendes I2C kommunikation for at sende og modtage data. Dette betyder, at der i PSoC's topdesign skal implementeres et I2C modul. Disse moduler blev sat op med passende pins i PSoC pin design. Derefter kunne sensoren sættes op med SCL og SDA på PSoC.

Selvom en TSL2561 har mange indstillinger, som kan brugerdefineres, så er det oplagt at anvende standardindstillingerne, som er på 402ms integrationstid og 1x gain. Dette er da 402ms giver den mest præcise måling og lægger stadig indenfor kravet for at sende data mindst hvert sekund.

For at kunne sende og modtage data fra en TSL2561 anvendes WriteData() og GetData() funktionerne i I2C.h

For videre forklaring af indstillingerne, illustration og fuld forklaring af opstilling, samt en udvidet forklaring af funktionerne anvendt i forbindelse med TSL2561 refereres til bilag<sup>16</sup>.

#### 8.1.1.2 Regn

Til måling af nedbør er der nogle opstillede krav. Disse er; at den skal kunne måle regnmængden mindst en gang i sekundet, og kunne måle regn med en præcision på +/- 10%.

Derfor er der valgt at bruge en Small rain gauge med en tipping bucket, som giver muligheden for at måle nedbør. Fremover i rapporten vil regn sensor blive anvendt fremfor small rain gauge med tipping bucket.

Denne løsning anvendes, da den nemt og præcist kan måle mængden af nedbør. Programmet skal læse et interrupt hver gang tipping bucket "tipper", som den gør når den indeholder ca. 5ml vand. Yderligere kan denne løsning anvendes, da den opfylder de opstillede krav til regn sensoren. Når regn sensoren tipper så modtages det et interrupt indenfor 300ms. Derved kan kravet om at være i stand til at læse data hvert sekund godkendes. Videre har denne regn sensor en afvigelse på +/- 5%<sup>17</sup>, som ligger indenfor præcisionskravet stillet i de ikke funktionelle krav.

For at få regn sensoren til at virke oprettes en variabel counterRegn, som læser interrupts på en rising-edge. Denne variabel inkrementeres hver gang, at den får et interrupt. Når 60 sekunder er gået, samles denne værdi med de andre værdier i systemets output buffer,

---

<sup>15</sup> (TSL2561 Datablad, 2021)

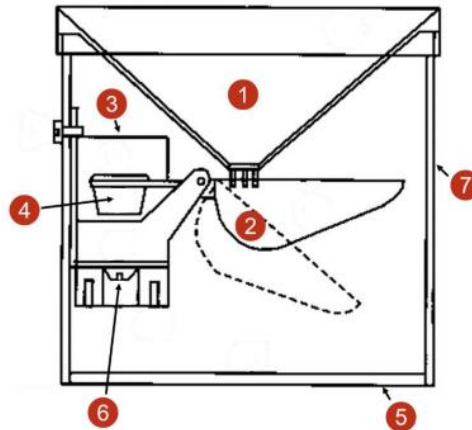
<sup>16</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\1.4.1 – Dataindsamler\implementation og design Psoc.pdf

<sup>17</sup> (Small Rain Gauge datablad, 2021)

og sendes med UART-kommunikation. Bagefter sættes værdien til 0 og der påbegyndes en ny læsning.

Regn sensoren fungerer således at den har to pins. På den ene sendes der en VCC, og på den anden pin er output signalet. Dette output signal kommer kun igennem når regn sensor tipper. Så denne spike fra 0-5V er det vi læser med vores interrupt.

Opstilling og modultest kan findes i bilag<sup>18</sup>



Figur 15 Illustration af small rain gauge<sup>19</sup>

På Figur 15, ses der en illustration, lavet af firmaet Pronamics, af en small rain gauge med en tipping bucket. Her ses der selve tipping bucket'en, som står for at opsamle nedbør, og når der er 5ml så tipper den og lader signalet komme igennem, som der læses på PSoC og udløser dens interrupt.

For fuld forklaring af punkter i Figur 15, samt uddybelse af regn sensor henvises der til bilag<sup>20</sup>

### 8.1.1.3 Temperatur

Systemet skal kunne opfylde kravet 5.1.4<sup>21</sup>, som er at det skal kunne måle temperaturen. Dette opnås ved at bruge en LM75, som er en digital temperatursensor. Denne sensor er brugt i projektet, da den er blevet anvendt tidligere, og der er derfor erfaring med arbejdet og I2C kommunikationen som anvendes her.

For at styre temperatursensoren anvendes funktionen, som kan ses på Tabel 1. Denne funktion vil være i I2C.h, hvor en udvidet forklaring af I2C protokollen kan findes i bilag<sup>22</sup>.

<code>double ReadTemp(uint8 address)</code>
---

Tabel 1: Funktion for temperatursensor

<sup>18</sup> Bilag \1 - Dokumentation\1.4 - Vejrmåler\1.4.1 - Dataindsamler\implementation og design Psoc.pdf

<sup>19</sup> (Small Rain Gauge datablad, 2021)

<sup>20</sup> Bilag \1 - Dokumentation\1.4 - Vejrmåler\1.4.1 - Dataindsamler\implementation og design Psoc.pdf

<sup>21</sup> Bilag \1 - Dokumentation\1.2 - Kravspecifikation\Funktionelle\_krav.pdf

<sup>22</sup> Bilag \1 - Dokumentation\1.4 - Vejrmåler\1.4.1 - Dataindsamler\implementation og design Psoc.pdf

Denne funktion anvendes så temperaturen kan læses på sensoren. Der vil blive sendt en besked til temperatursensoren om at starte temperaturlæsningen, og derefter vil data sendes i det tidsinterval det sættes til. I dette tilfælde vil systemet kunne aflæse fra temperatursensoren mindst en gang i sekundet. Test for dette kan ses i bilag<sup>23</sup> Og dette viser at det ikke funktionelle krav til målinger for sensoren, vil være opfyldt. Derudover tjekker funktionen også om den givne temperatur er negativ eller positiv. I koden vil der blive behandlet to variable med 8-bit som til sammen vil udgøre temperaturen. En mere detaljeret beskrivelse af testen kan findes under bilag<sup>24</sup> hvor det også ses, at der vha. funktionen fås den målte temperatur fra sensoren.

#### 8.1.1.4 Vind

Systemet skal kunne måle vindstyrken og der anvendes et NRG 40 anemometer, som er en vindmåler. Den har et måleinterval på 1-96 m/s<sup>25</sup>, og dette dækker alle vindstyrker i Danmark<sup>26</sup>. Der vil altså kunne opnås målinger af vindstyrken og derved vil det funktionelle krav 5.1.1 kunne indfries.

Vindmåleren producerer et udgangssignal hvor frekvensen er proportionalt med vindhastigheden. Dette udgangssignal vil være et sinusformet signal. Programmet er designet således, at der laves et interrupt, hver gang vindmåleren roterer en halv omgang, hvor en fuld rotation svarer til 0,765 m/s<sup>27</sup>. Derved vil det sige, at to interrupts svarer til 0,765 m/s. På Figur 16 ses hvordan denne vindmåler vil se ud.



Figur 16: Illustration af selve vindmåleren<sup>28</sup>

For at kunne detektere signalet fra vindmåleren, skal den kobles til en zero crossing detector. Den vil nemlig stå for at detektere sinussignalet, som vindmåleren vil generere, og dette vil så kunne give et højt eller lavt output. Det er netop dette output der læses som et interrupt, og derefter omregnes dette til m/s.

<sup>23</sup> Bilag \1 – Dokumentation\1.4 – Vejrmåler\1.4.1 – Dataindsamler\implementation og design Psoc.pdf

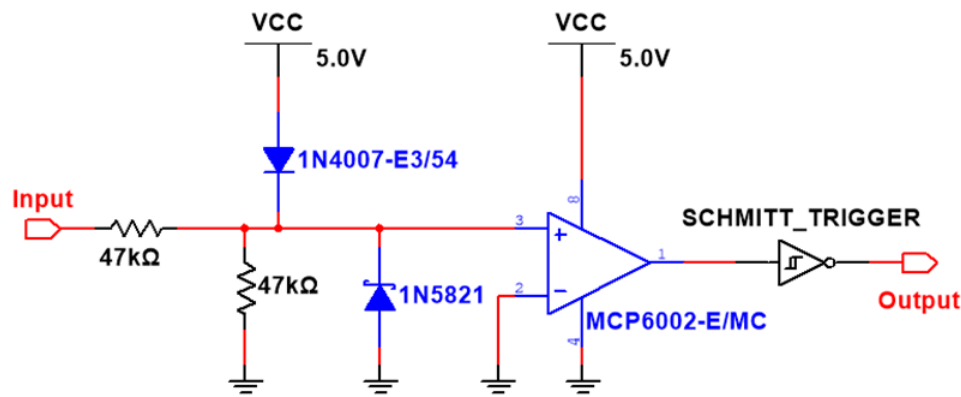
<sup>24</sup> Bilag \1 – Dokumentation\1.4 – Vejrmåler\1.4.1 – Dataindsamler\implementation og design Psoc.pdf

<sup>25</sup> Bilag \1 – Dokumentation\1.7 – Datablade\NRG40.pdf

<sup>26</sup> (DMI, 2021)

<sup>27</sup> Bilag \1 – Dokumentation\1.7 – Datablade\NRG40.pdf

<sup>28</sup> (Kintech Engineering, 2021)



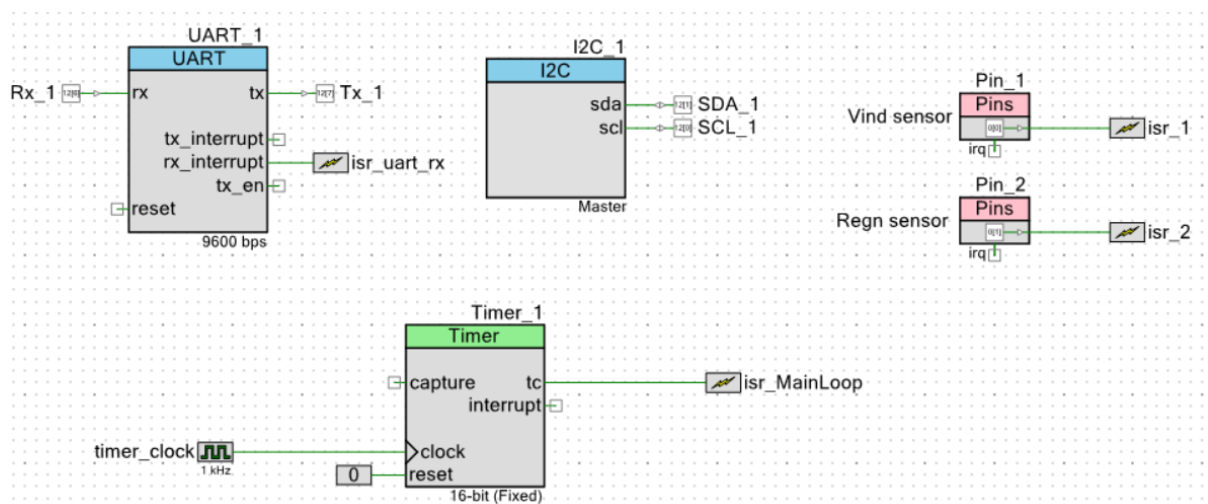
Figur 17: Zero cross detector kredsløb

På Figur 17 ses zero cross kredsløbet. Det er designet så signalet fra vindmåleren kommer ind i zero cross kredsløbets input. Output signalet, er signalet som registreres hos PSoC'en. Test og mere detaljeret beskrivelse for vindmåleren og zero cross kan findes i bilag<sup>29</sup>.

#### 8.1.1.5 Main

Efter der er blevet målt data fra de fire sensorer i et minut, hvor der samles data fra sensorer en gang i sekundet, samles alt denne data ved brug af funktionen `snprintf()`. Denne funktion gør det muligt at samle dataene i en string, som derefter opbevares i en buffer. Når denne data er blevet samlet i bufferen, kan PSoCs indbyggede UART kode anvendes. Her bruges funktionen `UART_PutString()`, til at sende fra bufferen vha. UART.

For at inkorporere alle brugte sensorer i et samlet program, så måtte de individuelle topdesigns for hver sensor program samles, sådan at den indeholdt de krævede komponenter i PSoC Creator, som er et I2C komponent, et UART-komponent, to pins med interrupts og et Timer komponent der bruges til at tælle til 60 sekunder hvorefter der sendes data, som ses på Figur 18.



Figur 18 PSoC Main Top design

<sup>29</sup> Bilag \1 – Dokumentation\1.4 – Vejrmåler\1.4.1 – Dataindsamler\implementation og design Psoc.pdf

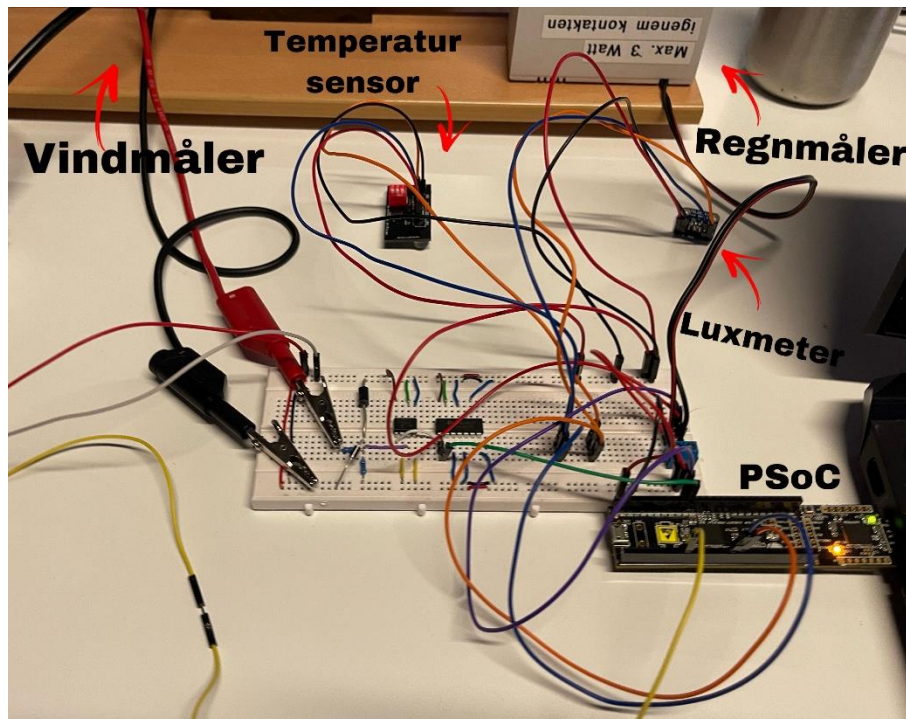
Hernæst skal forbindelser til top designets komponenter også sættes til en port, dette gøres under Pin fanen i PSoC Creator. Her er der f.eks. at Pin\_1, som står for vind sensor, er sat til port P0 [0], som er pin 48.

Topdesign, samt pin og port tildelingerne kan findes med fuld forklaring i bilag<sup>30</sup>.

#### 8.1.1.6 Samlet test af datamåling

For at vise hver sensors individuelle funktionalitet var der opstillet modultest for hver sensor, som kan findes i bilag<sup>31</sup>, udover dette blev en samlet integration med alle sensorer og main program også testet.

Der skulle der opstilles en integrationstest, hvor der testes om alle fire sensorer virkede sammen, og om programmet var i stand til at sende data mindst hvert sekund. Derfor bliver der opstillet en test, hvor data sendes hvert halve sekund.



Figur 19: Opstilling for samlet test for vejrmåler

Strømforsyning		Forbindes til VCC
Ground		Forbindes til GND
SDA		Forbindes til SDA
SCL		Forbindes til SCL
P0[0] fra PSoC		Forbindes til output signal fra vindmåler+zero cross
P0[1] fra PSoC		Forbindes til regnmåler

<sup>30</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\1.4.1 – Dataindsamler\ implementation og design Psoc.pdf

<sup>31</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\1.4.1 – Dataindsamler\implementation og design Psoc.pdf



Figur 20: Forbindelser for opstillingen

På Figur 19 ses der den samlede opstilling med alle sensorer på en PSoC. Til denne opsætning er ledningerne "colour-coded", og hver farves betydning kan aflæses i Figur 20.

For at teste om at disse sensorer virkede i samspil, blev der anvendt RealTerm til at se de sendte data fra sensorer og tid. I denne test blev hver sensor påvirket, og der kan derfor ses ændringer i deres måle data. Derfor må vi konkludere at de valgte sensorer virker og opfylder kravene stillet til dem. For fuld test henvises til bilag<sup>32</sup>, og for video af test henvises til bilag<sup>33</sup>.

Tid	Wind	Temp	Regn	Lys
0.500000 sekunder,	0.147500	27.500000	2.000000	388.000000
1.000000 sekunder,	0.765000	27.500000	2.000000	388.000000
1.500000 sekunder,	3.825000	27.500000	2.000000	359.000000
2.000000 sekunder,	1.147500	27.000000	2.000000	279.000000
2.500000 sekunder,	1.530000	28.000000	2.000000	221.000000
3.000000 sekunder,	1.147500	28.000000	3.000000	181.000000
3.500000 sekunder,	0.765000	28.000000	3.000000	172.000000
4.000000 sekunder,	0.765000	28.000000	3.000000	281.000000
4.500000 sekunder,	0.382500	28.500000	2.000000	421.000000
5.000000 sekunder,	0.765000	28.000000	3.000000	41.000000
5.500000 sekunder,	0.382500	28.000000	3.000000	40.000000
6.000000 sekunder,	0.382500	28.000000	3.000000	41.000000
6.500000 sekunder,	0.382500	28.000000	3.000000	41.000000
7.000000 sekunder,	0.382500	28.000000	3.000000	41.000000
7.500000 sekunder,	0.382500	28.000000	2.000000	41.000000
8.000000 sekunder,	0.382500	28.000000	2.000000	42.000000

Figur 21 Samlet sensor testresultater i RealTerm

<sup>32</sup> Bilag \1 – Dokumentation\1.8 – Integrationstest\Integrationstest.pdf

<sup>33</sup> Bilag \1 – Dokumentation\1.9 – Accepttest\Videoer til accepttest\Video 8 - systemet sender data fra psoc til Vejrviser

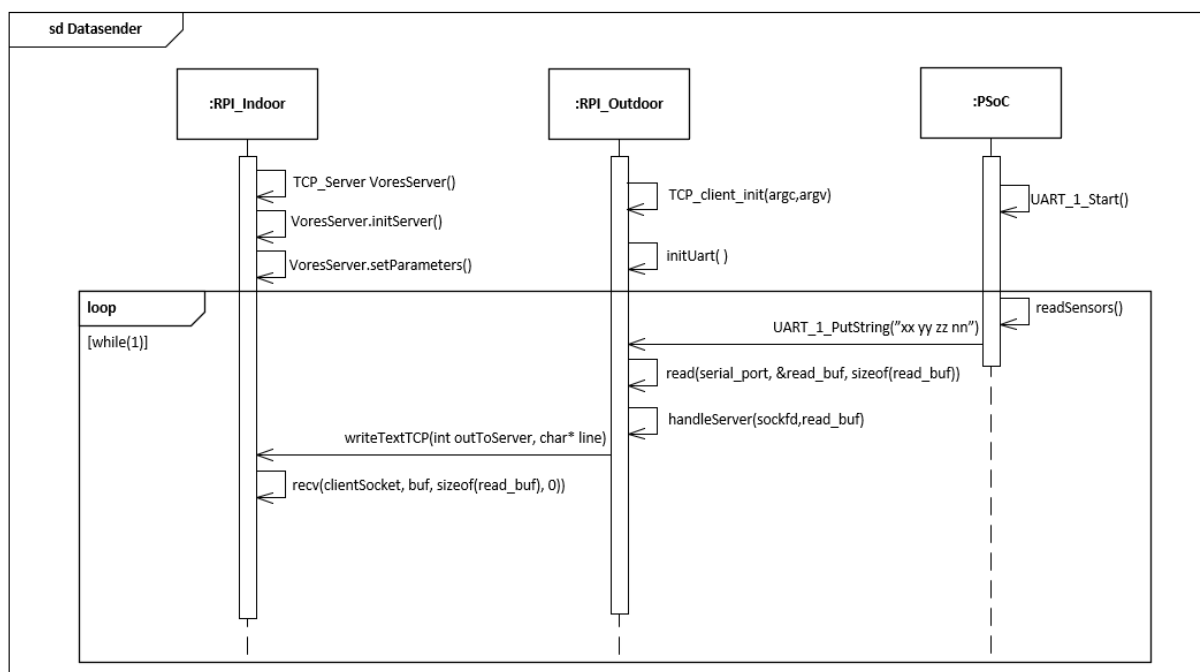
## 8.1.2 Datasender Modul

Datasender skal modtage den målte og behandlede data fra Dataindsamler (PSoC m. sensorer). Der skal samtidig oprettes en TCP-forbindelse imellem Datasenderen som Client og Vejrviser som Server. Dette gøres for at opfylde kravet: ” Use case 2: Indsamle vejrdato ” hvor data kontinuert skal kunne modtages fra Dataindsamler og sendes til Vejrviser over WiFi.

Ved boot af Datasender skal der automatisk oprettes en TCP-forbindelse til Vejrviser over WiFi med et konfigurerede WLAN. Denne forbindelse holdes åben og den kontinuerte læsning af vejrdato fra UART skal sendes herigennem videre til Vejrviser. Boot applikationen laves for at opfylde kravet: ” Use case 1: Tænd systemet ” som kræver en automatisk opstart af systemet.

Datasender har på denne måde både en grænseflade til Dataindsamler, som er wired og forbundet ved UART, men den har også en grænseflade til Vejrviser, som er wireless forbundet over WLAN med en TCP-forbindelse. Disse kommunikationsgrænseflader gør det muligt at anvende de sensorer, som er forbundet til Dataindsamler, til at interagere med aktuatorerne, som er forbundet til Vejrviser, der består af steppermotorerne i systemet.

I Figur 22 vises et sekvensdiagram for Datasender, hvor RPI\_Indoor, som er en del af Vejrviser, er medtaget for at illustrere kommunikation imellem klasserne.



Figur 22 - Datasender

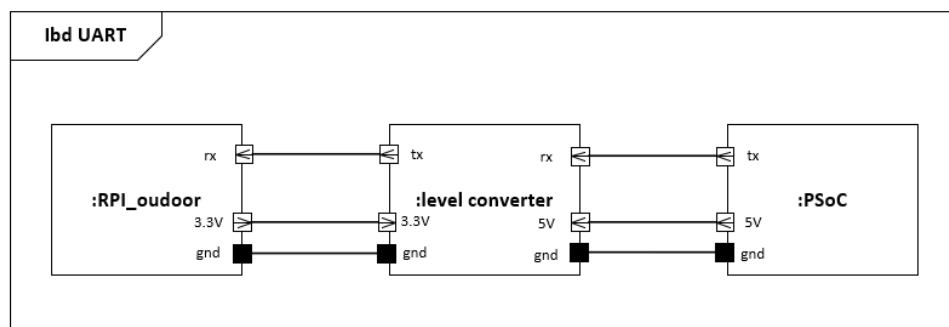


### 8.1.2.1 Modtager – UART

UART-modulet har til formål at læse de værdier, som repræsenterer de målte og efterfølgende behandlede sensordata fra Dataindsamler og på denne måde opfylde kravet: "Use case 2: Indsaml vejrdato".

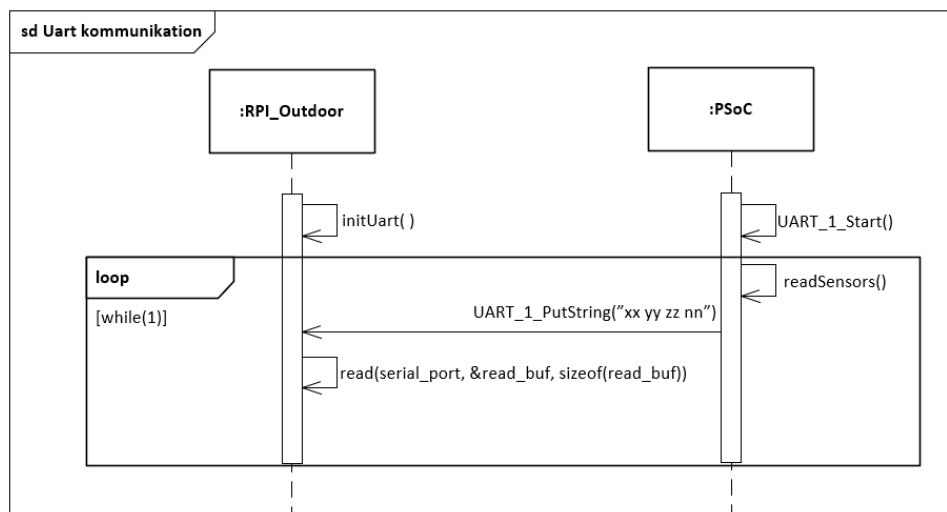
For at oprette en UART forbindelse mellem PSoC og RPI\_Outdoor benyttes der en Level Converter til at nedjustere fra 5V på PSoC TX pin til 3.3V.

Et hardware overblik til UART forbindelsen mellem Dataindsamler og Datasender vises nedenfor i Figur 23 for at skabe en bedre hardwaremæssig forståelse.



Figur 23 - ibd UART

Nedenfor i Figur 24 vises et sekvensdiagram hvor UART-delen af PSoC klassen indgår for at give en bedre forståelse for kommunikationen i og mellem disse klasser.



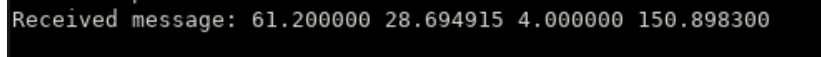
Figur 24 - UART-kommunikation

Når Datasender løbende modtager de målte værdier, skal disse efterfølgende sendes til Vejrviser over en oprettet TCP-forbindelse.

Dette vil implementeres ved at en TCP-forbindelse mellem Vejrviser og Datasender oprettes over WiFi, når RPI\_Outdoor booter. De målte og efterfølgende sendte værdier læses fra UART modulet på Datasender, og værdierne skrives efterfølgende som en string,

hvor værdierne er adskilt af et mellemrum (space) fra TCP-client (Datasender) til TCP-server (Vejrviser).

Et eksempel på resultatet af en modtaget string fra PSoC vises nedenfor i Figur 25. Hver kolonne er svarende til en værdi for hhv. vind, temperatur, regnmåler og lysintensitet.



```
Received message: 61.200000 28.694915 4.000000 150.898300
```

Figur 25 - UART resultat fra PSoC

For yderligere dokumentation af UART-modulet henvis til dokumentationen jf. bilag<sup>34</sup>

#### 8.1.2.2 Sender – TCP

For at løse, punkt 4 af use case 2; “*Datasender sender vejrdato trådløst til Vejrviseren*”, skal datasenderen sende, den data der er kommet ind med UART, videre til Vejrviseren. Det bliver besluttet at der er en server på Vejrviseren, hvorfor der skal oprettes en client på datasenderen. Denne client skal oprette forbindelse til serveren. Det gøres med TCP således at der er en sikker forbindelse.

Ved at bruge TCP kan den streng, som kommer ind fra UART-klassen sendes direkte videre til Serveren på en sikker måde.

Når der arbejdes med en webforbindelse, er der to ting som er væsentlige. Det er IP-adressen på Serveren og Portnummeret. IP-adressen defineres af den router, som serveren er koblet op til, hvilket betyder at det ikke er noget, der kontrolleres af udvikleren. Portnummeret er bestemt i serveren og skal kendes af clienten for at skabe forbindelse.

For at overkomme udfordringen i at kende IP-adressen på forhånd indsættes et argument fra terminalen ;

```
int TCP_client_init(int argc, char* argv[])
```

Figur 26 TCP-client konstruktor

Dette betyder, at når programmet startes fra terminalen eller fra et script skal IP-adressen skrives efter “programstart”.

Resten af clientes funktioner og beskrivelser kan læses i bilag<sup>35</sup>.

For at teste, at dette modul løser kravet om at sende data til Vejrviseren, laves et test program, som med denne TCP client skaber forbindelse til den server der ligger i Vejrviseren (fortælles mere om i afsnittet 398.2.3 Datamodtager) og derefter sender en hard coded streng som kan aflæses på Serveren

<sup>34</sup> Bilag \1 - Dokumentation\1.4 – Vejr måler\1.4.1-Datasender\UART.pdf

<sup>35</sup> Bilag \1 - Dokumentation\1.4 – Vejr måler\Vejr måler\_Design.pdf, side 8-11.

```

stud@stud-virtual-machine: ~/vejrstationen/RPI_Outdoor
stud@stud-virtual-machine:~/vejrstationen/RPI_Outdoor$ ./bin/x86-64/RPIOCLIENT localhost 54001

stud@stud-virtual-machine: ~/ServerTest/bin/x86-64
not connected
not connected
not connected
not connected
not connected
not connected
not connected
not connected
not connected
localhost connected on port 52078
run server midt
Hardcoded Section
run server slut
counter: 1
Vind er: 0
regn er: 0
temperaturen er: 0
lyset er 0

run server midt
Hardcoded Section
run server slut
counter: 2
Vind er: 0
regn er: 0
temperaturen er: 0
lyset er 0

```

Figur 27 resultat på test af client tidlig modultest

På Figur 27 ses det at der modtages strengen "HardCoded section" som bliver sendt af klienten. Hele testen er dokumenteret tydeligere i Bilag<sup>36</sup>. Med antagelse af, at der også kan sendes over lokalt wifi, når der kan sendes over internt net på den virtuelle maskine i linux, vurderes det, at testen viser, at kravet fra use case 2 bliver indfriet. For at sikre at kravet er løst sådan at det kan sendes over wifi henvises til intergrationstesten<sup>37</sup>

### 8.1.2.3 Integration af UART og TCP

I Figur 28 ses resultatet efter, at der er oprettet en TCP-forbindelse mellem Vejrviser og Datasender. UART-læsninger på Datasender af vejrdato, som er målt og behandlet på Dataindsamler, sendes over WLAN, hvor Vejrviser fungerer som TCP-server og Datasender fungerer som TCP-client.

```

buffer sending
handled server
while loop
Received message: 57.757500 26.686441 0.000000 343.152557

buffer sending
handled server
while loop
Received message: 39.397500 30.525424 123.000000 247.288132

buffer sending
handled server

```

Figur 28 - UART TCP

Nedenfor i Figur 29 vises en status for filen my\_uart\_app.service, som ved boot af RPI\_Outdoor automatisk starter applikationen uartClient med den tildelte IP-adresse og det ønskede portnummer.

<sup>36</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\1.4.2 - Datasender\TCP\_Client\_outdoor.pdf

<sup>37</sup> Bilag \1 - Dokumentation \1.8 – Integrationstest\Intergrationstest.pdf

```
root@raspberrypi0-wifi:~# systemctl status my_uart_app.service
● my_uart_app.service - QT Application
   Loaded: loaded (/etc/systemd/system/my_uart_app.service; enabled; vendor p
   Active: active (running) since Mon 2021-12-13 09:20:27 EST; 8min ago
     Process: 240 ExecStart=/bin/sh -c source /etc/profile ; /home/root/uartclie
    Main PID: 248 (uartClient)
       Tasks: 1 (limit: 265)
      CGroup: /system.slice/my_uart_app.service
              └─248 /home/root/uartClient 192.168.223.170 54000 -qws

Dec 13 09:20:26 raspberrypi0-wifi systemd[1]: Starting QT Application...
Dec 13 09:20:27 raspberrypi0-wifi systemd[1]: Started QT Application.
```

Figur 29 - my\_uart\_app.service status

Opfyldelse af kravet "Use case 1: Tænd systemet" som kræver en automatisk opstart af systemet, verificeres herved og bekræftes yderligere i accepttesten af systemet.

Applikationen uartClient er en integration af UART-modulet og TCP-Client, som beskrives yderligere i følgende bilag.<sup>38</sup>

Yderligere dokumentation og beskrivelser omkring bootloading og my\_uart\_app.service kan findes i følgende bilag.<sup>39</sup>

Som der er vist i Figur 29 og nedenfor i Figur 30, oprettes der forbindelse til TCP-serveren ved at connecte til en hardcoded IP-adresse og portnummer.

```
root@raspberrypi0-wifi:~# ./uartClient 192.168.223.170 54000
Success in connectTCP init
i main after UART open
while loop
```

Figur 30 - TCP connection succes

IP-adressen 192.168.223.170 er den adresse, som RPI\_Indoor er tildelt på det konfigurerede WLAN hvorfor der verificeres at forbindelsen og kommunikation sker over WiFi. Derved vurderes det at kravet: "Use case 2: Indsaml vejrdato" hvor data kontinuert skal kunne modtages fra Dataindsamler og sendes til Vejrviser over WiFi er opfyldt.

Yderligere dokumentation til opsætning og konfigurerings af WiFi og WLAN kan findes i dokumentationen jf. bilag<sup>40</sup>.

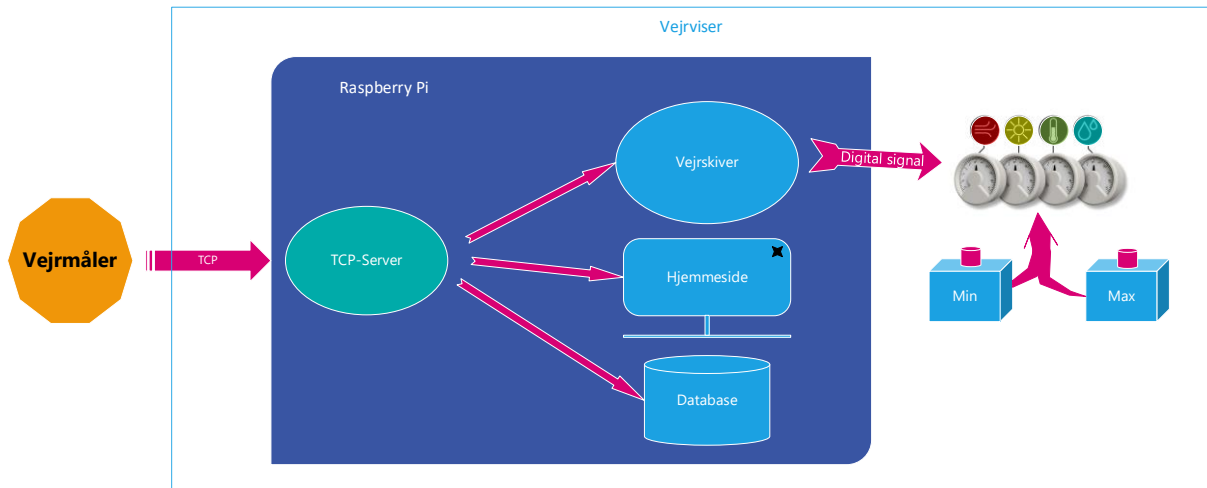
<sup>38</sup> Dokumentation\1.4 – Vejrmåler\1.4.1-Datasender\UART.pdf, side 15-16

<sup>39</sup> Dokumentation\1.4 – Vejrmåler\Bootloader.pdf, side 6-7

<sup>40</sup> Bilag \1 - Dokumentation\1.4 – Vejrmåler\Wifi.pdf

## 8.2 Subsystem Vejrviser

Vejrviseren står for at afbilde vejr situationen til brugeren. Delsystemet administreres af en CPU, som er en Raspberry Pi Zero W. Der er valgt en RPI som controller, fordi den opfylder kravene om WiFi og arm-processor, og fordi den er komfortabel at bruge efter, anvendelse i undervisningen. For at kunne skelne denne RPI fra den underdørs RPI, kaldes denne for RPI\_Indoor. Nedenfor er det illustreret (Figur 31), hvordan Vejrviseren hænger sammen.



Figur 31: Illustration af vejrviserens opbygning

### Valg af kodesprog

Kodesproget benyttet på RPI\_Indoor er C++, da det ønskes at benytte Objekt Orienteret Programmering, så delsystemet kan opdeles yderligere og sikre en høj grad af indkapsling. Herved forsøges det at skabe lav kobling mellem klasserne, så de i udviklingsprocessen kan skabes nærmest uafhængigt og også er skalérbare.

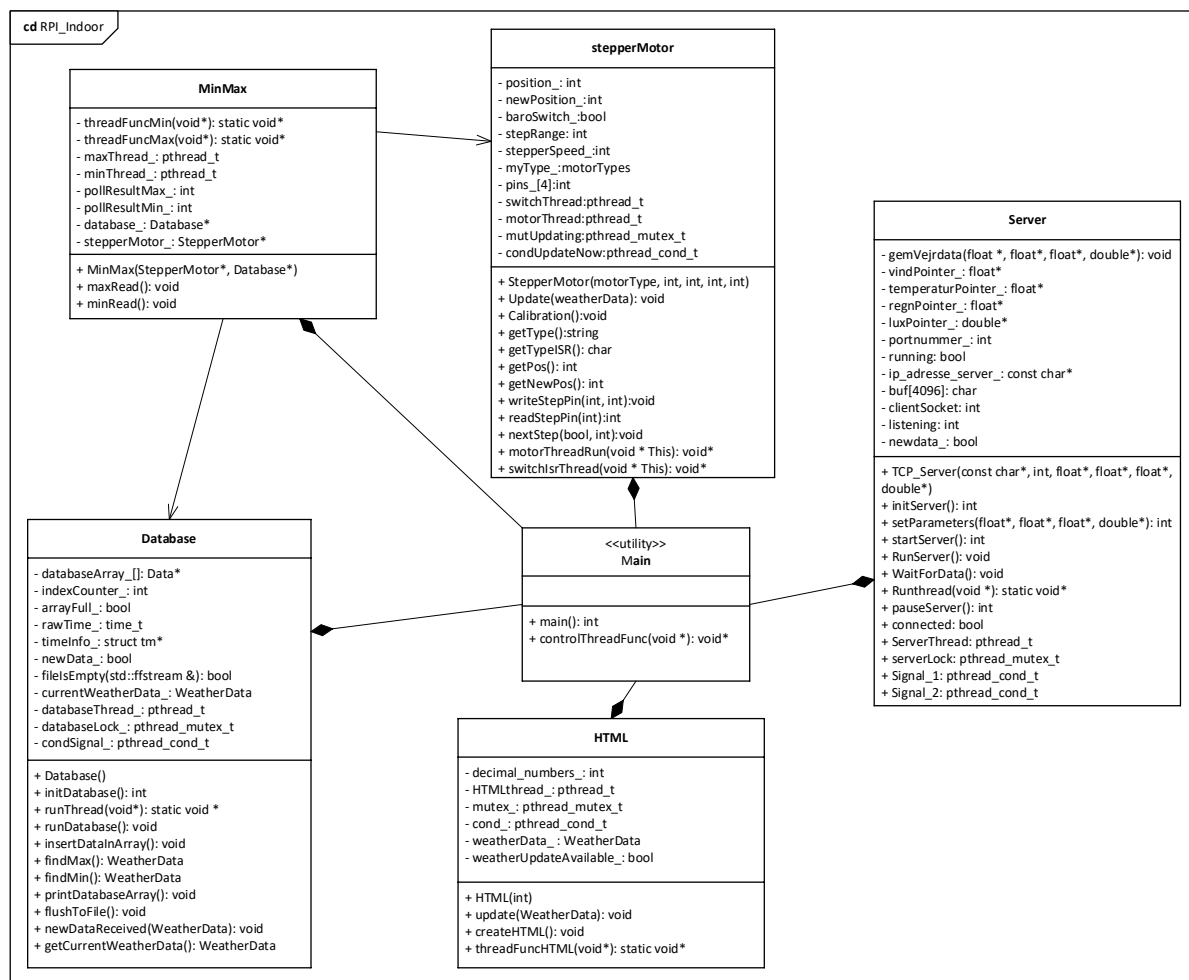
### Brug af tråde

Der benyttes tråde til hvert modul i delsystemet, da der er meget concurrency, når der bliver modtaget data. Trådene er strengt taget kun nødvendige for Server og stepperMotor, da de ikke må afbrydes i henholdsvis modtagelse af ny vejrdato og opdatering af vejrsriver. At hvert modul har sin egen tråd, giver dog en fordel i eksekveringstiden, da Main blot kalder en opdaterings-funktion til hver af modulerne, og så kører trådene selv resten af afviklingen. Dermed optimerer vi systemets opdateringshastighed, hvilket er godt, såfremt det ønskes at udvikle systemet og sende flere målinger i minuttet.

## 8.2.1 Klassediagram

RPI\_Indoor består af fem moduler og et Main-program. Den fulde beskrivelse af Vejrviserens opbygning kan læses i design-dokumentet for vejrviseren<sup>41</sup>. På diagrammet nedenfor (Figur 32) kan det ses, hvordan de fem moduler og Main er forbundet. Derudover vises der, hvilke funktioner og attributter de enkelte dele består af. Disse bliver dog først uddybet i de enkelte modulers afsnit.

Der er en lav kobling mellem de forskellige moduler. Koblingen sker primært gennem Main, der er forbundet til alle fem moduler. Det er nødvendigt, da Main konstruerer objekter af de forskellige klasser. Derudover er det Main, der fungerer som en distribution, når der modtages nye vejrmaalinger fra Server. Her kalder Main nogle opdateringsfunktioner i Database, HTML og stepperMotor, som signalerer, at der er modtaget ny vejrdato. Foruden koblingerne fra Main er de eneste koblinger fra MinMax-modulet, som er forbundet til både Database og stepperMotor. Grunden hertil er, at MinMax skal indhente min- og max-værdier fra Databasen, og efterfølgende den nyeste måling. Disse værdier bruger MinMax til at opdatere stepperMotor's visere.

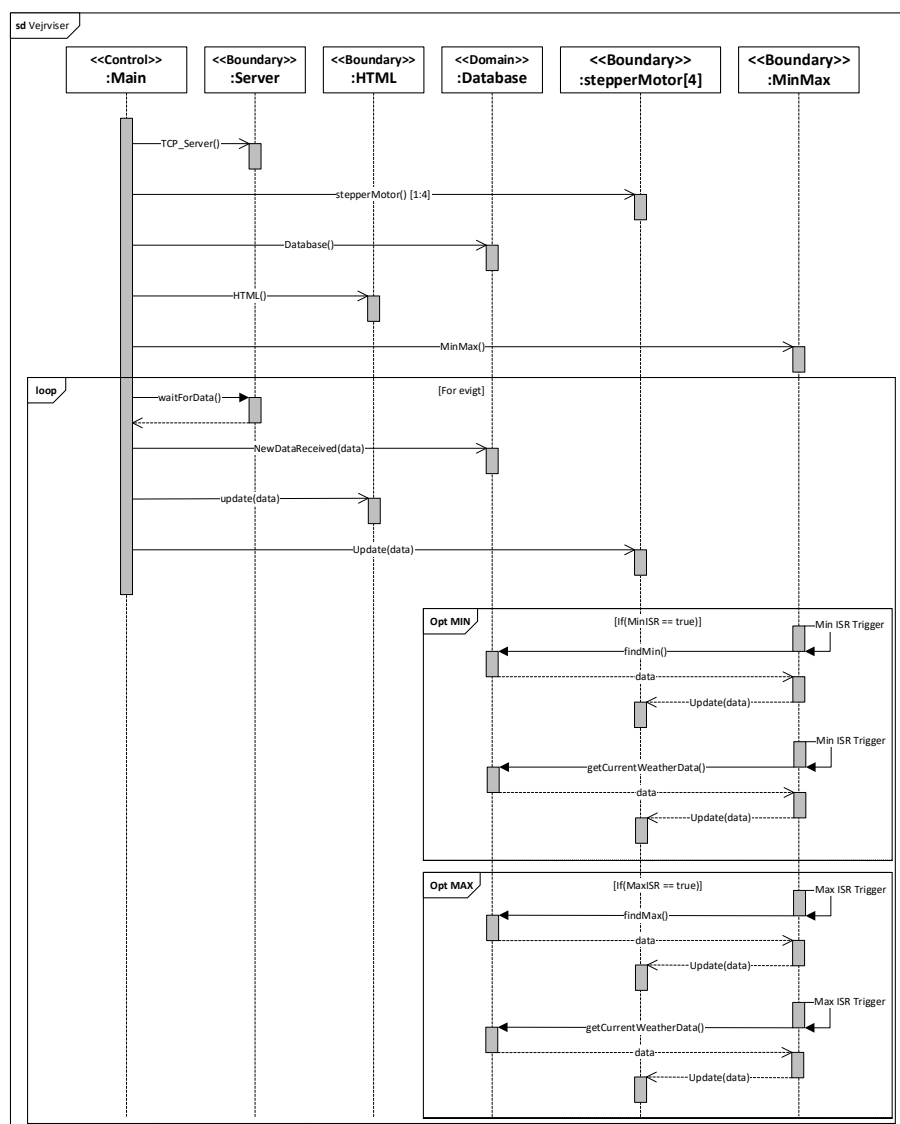


Figur 32: Klassediagram for vejrviser, RPI\_Indoor

<sup>41</sup> Bilag \1 - Dokumentation\1.5 – Vejrviser\Vejrviser\_Design.pdf

### 8.2.2 Sekvensdiagram

For at vise handlingsforløbet for vejrviseren er der lavet et sekvensdiagram for RPI\_Indoor (Figur 33). Sekvensdiagrammet samler følgende use-cases fra kravspecifikationen<sup>42</sup>: **UC1** - Tænd systemet, **UC3** - Opdater vejrviseren vejrskiver, **UC4** - Opdater hjemmeside og **UC5** - Aflæs max-/min-værdier. Alle moduler initialiseres fra Main. Herefter startes et uendeligt loop, som kalder metoden `waitForData()` i Server. Denne metode returnerer først, når der er modtaget ny vejrdato, hvorefter HTML, Database og stepperMotor kan opdateres med den nye data. Derudover viser diagrammet nederst, hvordan der tages hånd om, at der trykkes på Min- eller Max-knappen. Diagrammet viser ikke, hvilke metoder, der kaldes indbyrdes i modulerne, når eksempelvis `newDataReceived(data)` kaldes fra Main. Modulernes individuelle dokumentation kan ses i bilag<sup>43</sup>.



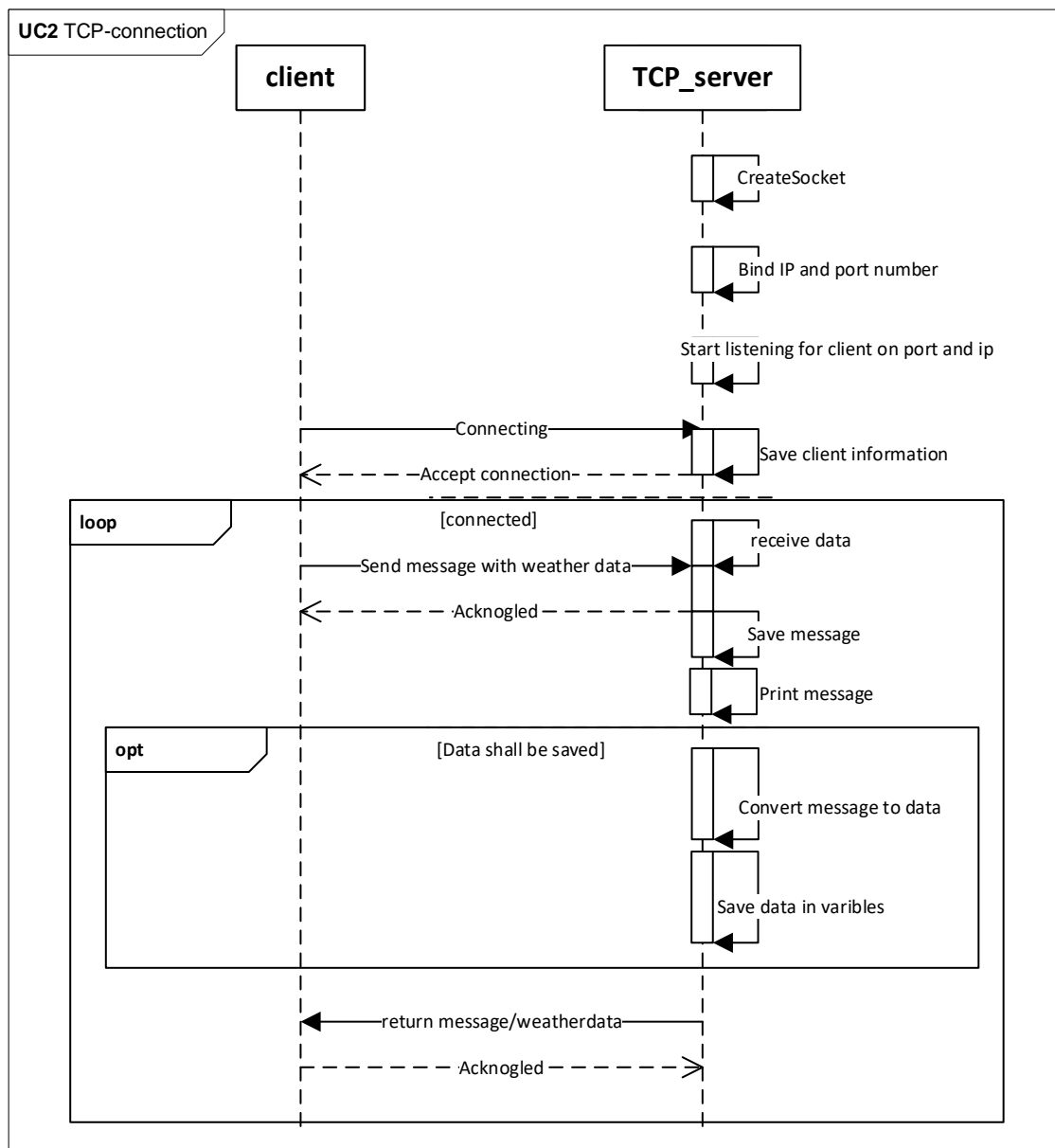
Figur 33: Sekvensdiagram for vejrviser, RPI\_Indoor

<sup>42</sup> Bilag \1 - Dokumentation\1.2 – Kravspecifikation\Funktionelle\_krav.pdf

<sup>43</sup> Bilag \1 - Dokumentation\1.5 – Vejrviser

### 8.2.3 Datamodtager

Vejrviseren skal ifølge usecase 3-6 modtage vejrdato. Så for at modtage data på vejrviseeren skal der være en server som kan modtage værdier. For at sikre at forbindelsen holder, bruges TCP. De 2 store grunde til at bruge TCP er *ordnet modtagelse*; dette sikre at modtageren får data pakker i sammen rækkefølge som de bliver afsendt, og *stabilitet*; dette sikre at hvis en datapakke går tabt så bliver den sendt igen. Den ordnede modtagelse er ikke grunden til at der er valgt TCP i dette projekt. Da der kun er et led mellem sender (outdoor client) og modtager (TCP-server) er der ikke nogen risiko for, at routen skulle være anderledes på internet-laget. Når TCP-beslutningen er på plads, kan man begynde at se på et sekvensdiagram (Figur 34) over dens handlinger.



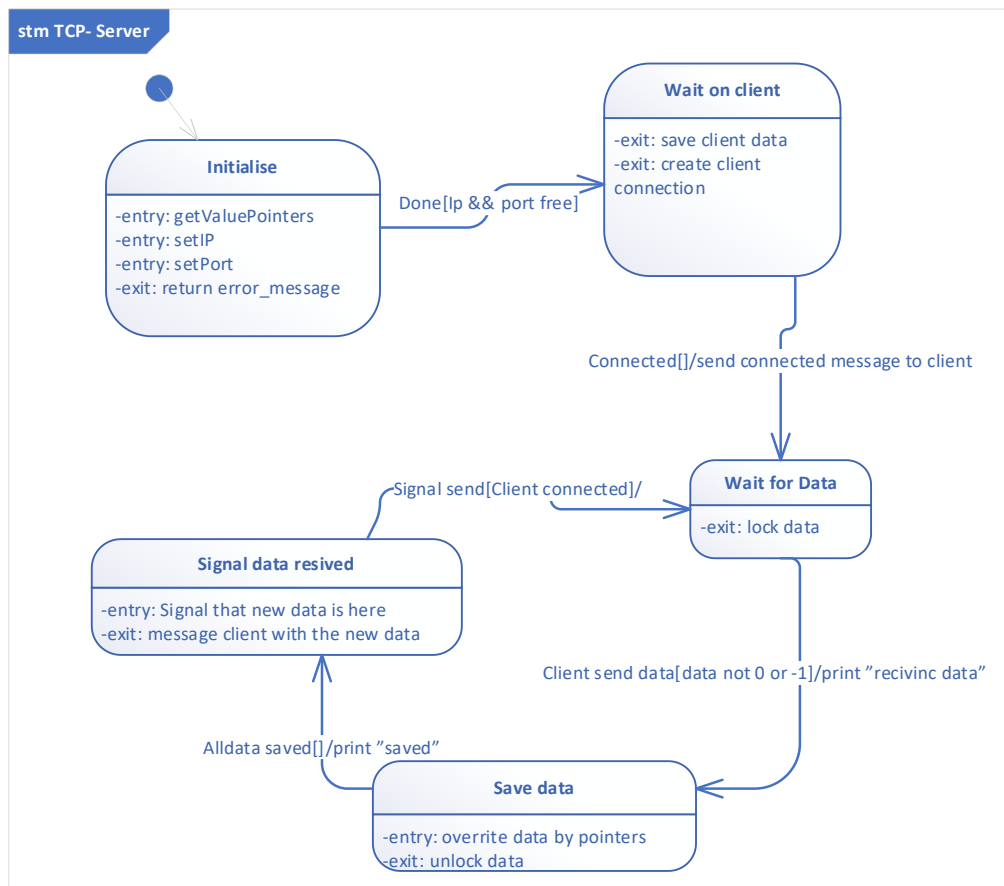
Figur 34: Sekvens diagram over TCP-serveren overordnet



Til at starte med opsættes netværksforbindelsen, dvs. Socket, IP og portnummer. Så ventes der på Client connection. Så skal der bare ventes på data.

Disse data skal så gemmes et sted og for at serveren kan køre automatisk overskriver den de samme 4 variabler igen og igen. Serveren kender til variablerne ved at få en pointer med til de fire variabler i konstruktoren. På den måde bliver der ikke oprettet nye variabler og gemt data i klassen TCP-server. I stedet skal der en anden klasse til hvis disse data skal gemmes.

For at give overblik på, hvad modulet skal gøre, er der lavet et state machine diagram. Det giver et godt overblik i når der arbejdes med gentagende processor. Det kan ses på Figur 35 .



Figur 35 State machine diagram over TCP-Server til at give bedre forståelse af statierne

For at serveren kan låse brugen af disse data mens den skriver til dem, bruges en `pthread_lock`, som er en public attribut i serveren. Signalet er også en public attribut. Dette gør det muligt at bruge serverens lås og signal, og på den måde sikre, at der ikke kommer problemer med shared data.

En anden løsning er også implementeret i serveren, nemlig en simpel if sætning. Dette gør at serveren kun gemmer data hvis en privat variabel er true. Ved at have to public funktioner som kan ændre denne variabel, kan man manuelt låse serveren ude så den ikke kan overskrive før den igen får lov manuelt.

I bilag<sup>44</sup> for TCP\_server implementering kan man se en mere uddybet forklaring af koden.

For at teste at serveren virker er der bygget et lille testprogram som kan ses i bilag<sup>44</sup> på side 12 som er testet ved hjælp af telnet.

Figur 36 resultat på test af TCP-server.

I testprogrammet gemmer serveren dataene via pointere til variabler og disse variabler udskrives. Ved denne test af TCP-serveren er det vist at dette modul kan modtage værdier via en TCP forbindelse. Hvilket gør det muligt at modtage vejrdato fra vejrmåleren og derved leve op til kravene i usecase 3-6. koden findes i bilag<sup>45</sup>

## 8.2.4 Database

I kravspecifikationen UC4 beskrives det, at vejrmålingerne skal lagres i en database, når det modtages hos vejrviseren. For at løse dette krav er der lavet et database-modul (Det fulde design og implementering kan ses i bilag<sup>46</sup>). Dette modul sørger for arkivere den modtagne vejrdato. Hver gang serveren modtager en ny vejrmåling, sendes denne videre til Databasen. Databasen tilføjer en dato til målingen, hvorefter den lægges i et Array. Dette Array består af 1440 pladser - plads til én måling hvert minut i et døgn. Der er valgt et statisk Array fremfor en dynamisk container, da størrelsen på den interne Database hele tiden er den samme. Hver gang der kommer en ny måling, udskiftes den ældste måling i Arrayet.

Databasens opgave er hovedsageligt at forsyne MinMax-modulet med min- og max-værdier for det seneste døgn. Når Databasens metoder; findMin() eller findMax() kaldes af MinMax, gennemløber Databasen hele Arrayet for at finde min-/max-værdierne for alle måle-typer, hvorefter disse værdier returneres til MinMax.

Hver gang Databasens Array er fyldt op på ny - sidste plads i Arrayet er nået, flushes hele Arrayet ud i en csv-fil. Dvs. at én gang i døgnet bliver alle vejrdato-objekter i Arrayet kopieret ud i en ekstern csv-fil. På den måde er historikken for vejrmålingerne gemt - også selvom systemet slukkes. Visionen var, at den eksterne fil skulle bruges til at lave grafer over tidligere dages vejrforhold, men det kom ikke med i denne omgang. Af den årsag virker den eksterne fil måske lidt overflødig.

<sup>44</sup> Bilag \1 - Dokumentation \1.4 – Vejrviser \ServerIndoor.pdf

<sup>45</sup> Bilag \1 - Dokumentation \1.6 – Kode \1.6.1 – RPI\_indoor\1.6.1.3 – Modultest\TCP\_server

<sup>46</sup> Bilag \1 - Dokumentation \1.4 – Vejrviser \Database.pdf

Databasen bruger en tråd for at sikre hurtig afvikling. Databasen er opbygget til, at Main kan kalde Databasefunktionen, `newDataReceived()`, hvor en ny vejrmåling gives med som parameter. Denne metode lægger den nye vejrmåling ind i Databasens private `WeatherData`-objekt, hvorefter den sender et signal til Database-tråden om, at der er modtaget ny data. Herefter klarer Database-tråden resten af arbejdet med at lægge vejrmålingen i Databasen. På den måde skal Main-tråden kun igangsætte processen, hvorefter den kan gå videre i Main.

Databasen er testet ved at udvikle et main-program (se bilag<sup>47</sup>), der laver et Database-objekt med en array-størrelse på 5. Herefter simuleres det fra main, så der sendes 7 nye målinger til Databasen. Det resulterede i, at der blev oprettet en csv-fil med de første 5 målinger. Efter afsendelsen af vejrmålingerne blev indholdet af `databaseArray_` udskrevet, hvorefter `findMin()` og `findMax()` blev kaldt og returnerværdien udskrevet. Her kunne det ses, at værdierne blev fundet blandt de sidste 5 målinger. Hermed virkede Databasen efter hensigten og kan leve op til UC4. Testen kan ses på Figur 37.

```
root@raspberrypi0-wifi:~# ./prog
Sender 7 Weatherdata til Databasen:
Maaling 1 - vind: 1 - lux: 2 - temp: -9 - regn: 3
Maaling 2 - vind: 2 - lux: 3 - temp: -8 - regn: 4
Maaling 3 - vind: 3 - lux: 4 - temp: -7 - regn: 5
Maaling 4 - vind: 4 - lux: 5 - temp: -6 - regn: 6
Maaling 5 - vind: 5 - lux: 6 - temp: -5 - regn: 7
Flushing to file database.csv ...
Maaling 6 - vind: 6 - lux: 7 - temp: -4 - regn: 8
Maaling 7 - vind: 7 - lux: 8 - temp: -3 - regn: 9

Indhold i databaseArray_:
Maaling 1 - vind: 6 - lux: 7 - temp: -4 - rain: 8 - dato: 3.12.2021
Maaling 2 - vind: 7 - lux: 8 - temp: -3 - rain: 9 - dato: 3.12.2021
Maaling 3 - vind: 3 - lux: 4 - temp: -7 - rain: 5 - dato: 3.12.2021
Maaling 4 - vind: 4 - lux: 5 - temp: -6 - rain: 6 - dato: 3.12.2021
Maaling 5 - vind: 5 - lux: 6 - temp: -5 - rain: 7 - dato: 3.12.2021

Max- og min-værdier i databaseArray_:
Max vind: 7 - Max lux: 8 - Max temp: -3 - Max regn: 9
Min vind: 3 - Min lux: 4 - Min temp: -7 - Min regn: 5
^C
root@raspberrypi0-wifi:~# cat database.csv
dato, lux, temp, rain, vindstyrke
3.12.2021, 1, 2, -9, 3.
3.12.2021, 2, 3, -8, 4.
3.12.2021, 3, 4, -7, 5.
3.12.2021, 4, 5, -6, 6.
3.12.2021, 5, 6, -5, 7.
root@raspberrypi0-wifi:~#
```

Figur 37: Modultest af Database

<sup>47</sup> Bilag \1 - Dokumentation \1.6 – Kode \1.6.1 – RPI\_indoor\1.6.1.3 – Modultest\Database

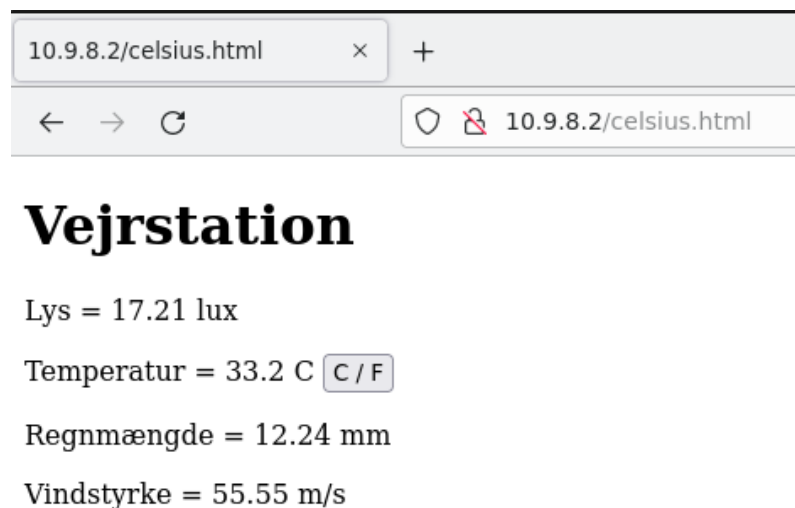
### 8.2.5 Hjemmeside

Hjemmesidens funktionaliteter håndteres af HTML modulet, der står for at efterkomme alle krav relateret til hjemmesiden bl.a. Use case 6, samt det funktionelle krav 5.1.6. Se bilag for kravspecifikationer<sup>48</sup>.

Modulet er designet som et rent software modul der sikrer brugerens adgang til de nuværende vejrdata over en WiFi forbindelse via en webbrowser. Fordi der er valgt en RPI som CPU, findes der i forvejen en webserver som kan benyttes til at hoste hjemmesiden.

Idéen er at modulet signaleres fra main, så snart Server modulet har opdateret en vejrdatavariabel i main. Den variabel skal håndteres i HTML modulet så data skrives til to filer i to forskellige visningsformater på RPI\_Indoor's webserver. Herfra skal vejrdata kunne tilgås via webbrowseren. Webbrowseren bør opdateres automatisk hvert tiende sekund og brugeren skal desuden kunne ændre visningsformat for temperaturen via en knap, der skifter mellem celsius og fahrenheit visning.

Modulet er implementeret i C++, og skriver data opstillet som et html dokument direkte til to forskellige filer på RPI'en til /www/pages som er derfra webserveren bliver hosted. Implementeringen benytter tråde så der kan skrives til filerne uafhængigt af andre kørende processer. På Figur 38 ses hjemmesiden for celsius visning under modultest.



Figur 38: HTML celsius visning

Implementering og modultest af HTML modulet kan ses yderligere beskrevet i bilag<sup>49</sup>.

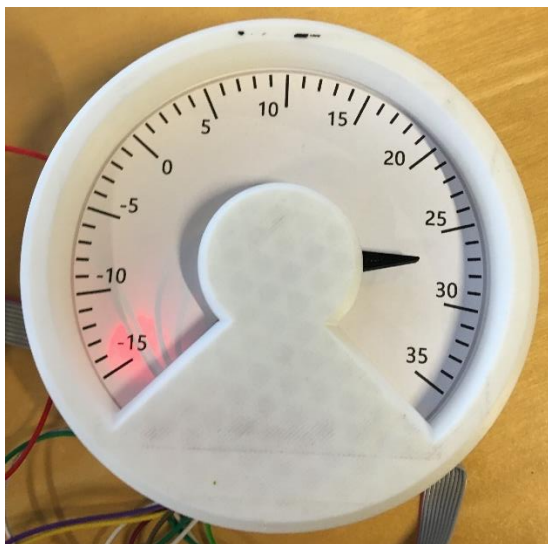
<sup>48</sup> Bilag 1 – Dokumentation\1.2 – Kravspecifikation\

<sup>49</sup> Bilag 1 – Dokumentation\1.5 – Vejrviser\HTML.pdf

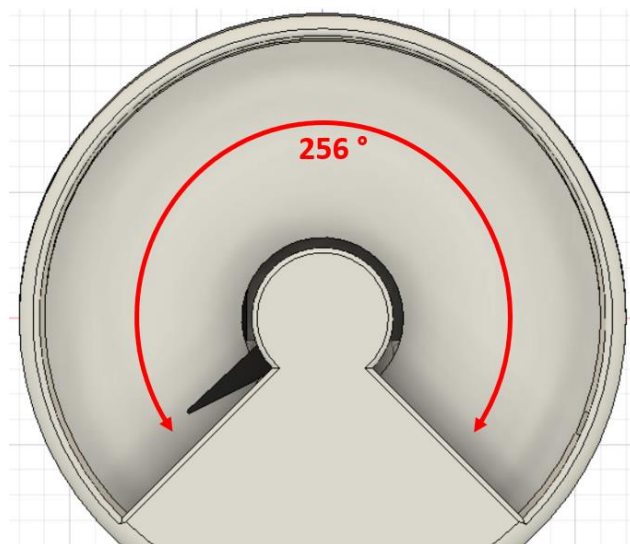
### 8.2.6 Vejrskiver

Vejrskiverne varetages af StepperMotor modulet. Det har til formål at løse kravet, fra kravspecifikationerne, at systemet skal afbilde vindstyrken, regnmængden, lys niveauet og temperaturen, på fysiske måle-skiver. Tidligt i designprocessen blev det besluttet at vejrdatoene skulle kunne afbildes, ikke med display, men med en viser på en skive, for at imitere et barometer el. ur m. visere. Løsningen til dette designvalg, blev at bruge en stepper motor, da den med høj præcision kan finde frem til samme position, gentagende gange. En 5V unipolar, gearede, stepper motor blev valgt, da den er nem at arbejde med, især i kombination med et Darlington Array.

I hjertet af modulet er softwareklassen StepperMotor, som har til formål at tage dataene, modtaget af serveren, og bruge disse data til at drive de fire stepper motorer, som repræsenterer hhv. vindstyrke, regnmængde, lysniveau og temperatur, til specifikke positioner på deres vejr skiver, så vejr dataene kan aflæses fra de fysiske skiver, se Figur 39. Derudover skal stepperMotor klassen også stå for at kalibrere stepper motorernes positioner, som Vejrviseren tændes, for at leve op til kravet i use case 1. Modulet består af et tæt sammenspil mellem software og hardware.



Figur 39: Afbildning af en temperaturmåling på en vejrskive



Figur 40: Afbildningsintervallet på vejrskiven

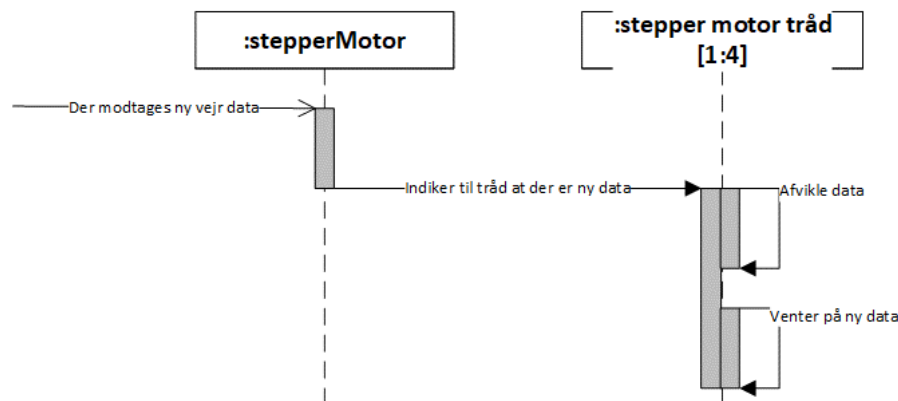
Den modtagne data skal konverteres til værdier, der stemmer overens med de steps, som afdækker hele afbildningsintervallet.

I løbet af undersøgelsen af stepper motorerne, er det blevet fundet frem til at den skal bevæge sig 2048 trin for at bevæge sig én fuld rotation. Det er blevet bestemt at designet af en vejrskive skal kunne vise vejr data værdier hen over 256 grader af skivens overflade, som giver 1460 trin for stepper motoren. Denne værdi er derfor det komplette interval, som stepper motorerne skal bevæge sig inden for, se Figur 40.

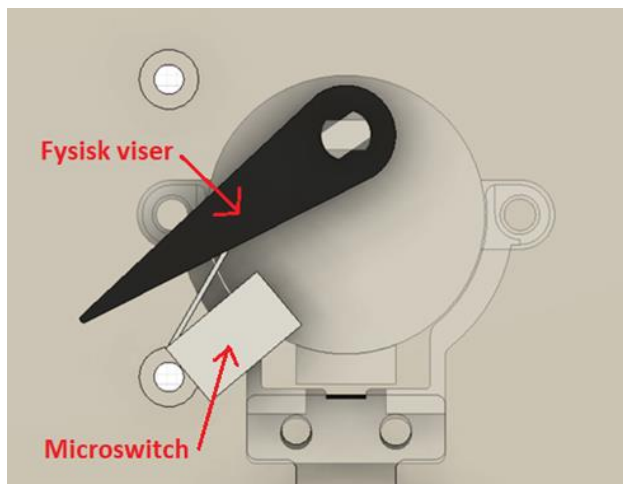
StepperMotor klassen består i essens af en suite af funktioner, som kan udføre handlinger som manipulerer bevægelsen af stepper motorerne. Det er bl.a. manipulation af GPIO-

pins og deres værdier gennem filedescriptor funktioner, samt styring af hvilket step motoren er på, og sætte dem i bevægelse. Se bilag<sup>50</sup> for fuld beskrivelse af klassen.

En af udfordringerne med stepperMotor klassen har bl.a. været at få samtlige vejr skiver til at afvikle bevægelser mod nye dataværdier, samtidig. Denne udfordring er blevet løst ved at bruge tråde, en tråd per objekt, el. rettere, en per stepper motor. Når objektet bliver givet en ny vejr data værdi, vil stepperMotor klassen vække stepper motor tråden, så den kan tage den nye værdi, og begynde at bevæge motorens viser mod den nye værdi, se sekvensdiagrammet i Figur 41 som viser denne interaktion.



Figur 41: Samtidighed blandt vejrskiverne



Figur 42: Interaktion mellem vejrskive viser og microswitch

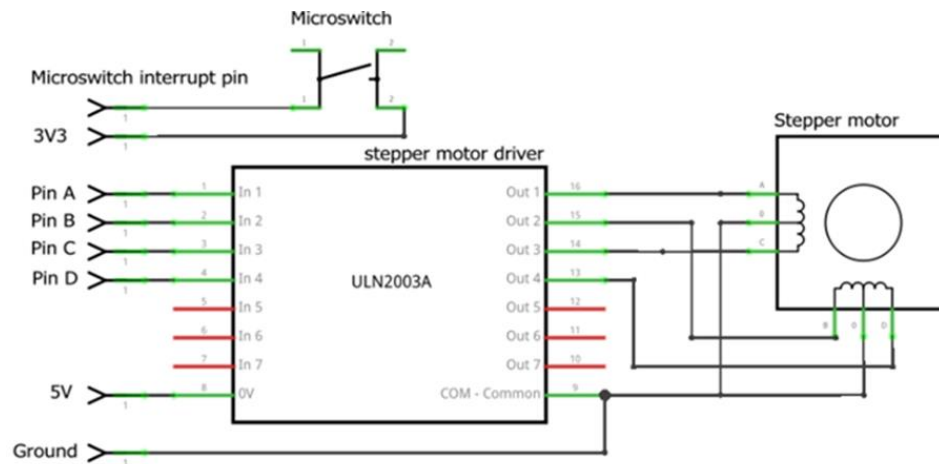
Et andet krav er at, før at stepper motorerne kan afbilde vejr data, skal stepper motorerne vide hvor den fysiske er i forhold til vejr værdiskiverne. Løsningen har været at lave en kalibrerings funktion, som får stepper motoren til at bevæge sig mod uret, og fortsætte med dette, indtil den fysiske viser rammer en microswitch placeret i vejen for den fysiske viser monteret på motoren, se Figur 42.

Når switchen trykkes af denne viser, vil det aktivere et interrupt, som aflæses af stepperMotor objektet, og som så stopper motoren med at bevæge sig, og nulstiller dens position. StepperMotor objektet er nu klar til at afbilde data korrekt.

<sup>50</sup> Bilag \1 – Dokumentation\1.5 – Vejrviser\StepperMotor.pdf



Multipliciteten i at der forefindes fire vejrskiver, gør at en stor del af hardwaren skal produceres fire gange. Det er dog gjort mere overskueligt ved at benytte et ULN2003 Darlington array, se diagram af kredsløbet for vejrskiverne i Figur 43.



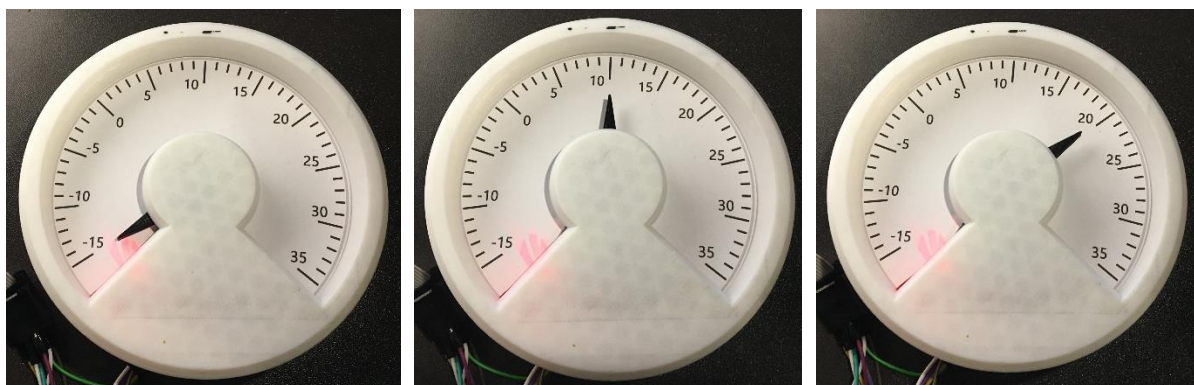
Figur 43: Vejrskive kredsløbsdiagram

Efter processen med at implementerer stepperMotor modulet, har det været testet, om det faktisk efterlever de krav som blev udpenslet i starten af projektet. Der er blevet lavet en del tests, f.eks. sikre at alle GPIO-pins bliver initieret korrekt, og at alle interrupt drivers virker efter hensigten, men den mest spændende test har været den endelige test, hvor samtlige funktioner kom i spil.

Testen går ud på at der køres et testprogram, kun med temperaturvejrskiven, og som udfører 3 handlinger:

1. Kalibrere positionen af vejrskiven.
2. Vejrskiven bliver opdateret med ny data, 10 grader.
3. Vejrskiven bliver opdateret med ny data, 20 grader.

Resultatet kan ses i følgende fotos, se Figur 44, og et screenshot fra terminalen under kalibreringsprocessen kan ses i Figur 45.



Figur 44 StepperMotor modultest

```
temp Scanning for beginning: -438
temp Scanning for beginning: -439
temp Scanning for beginning: -440
temp Scanning for beginning: -441
! ISR for temp fired!
Stopping ISR for temp...
temp Scanning for beginning: -442
Resetting position: -442 -> 0
```

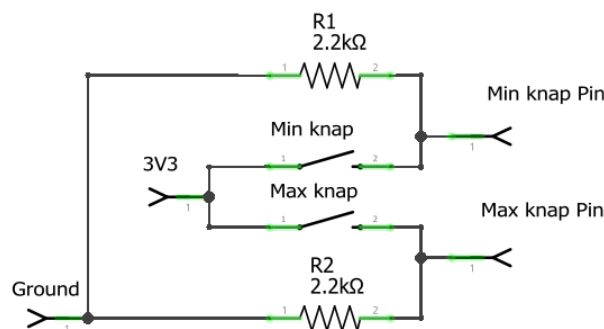
Figur 45 StepperMotor under kalibrering

Efter denne modultest kan det så konkluderes at stepperMotor modulet virker efter hensigten, og lever op til de opstillede krav.

### 8.2.7 MinMax

MinMax er et kombineret software-/hardware-modul der skal efterkomme det funktionelle krav opstillet i use case 6: Aflæs Max/Min værdier. Det er valgt at modulet skal bestå af to fysiske knapper, der skal generere interrupts og et softwaremodul der skal sikre visning af maksimum-/minimumværdier for vejrdata målt indenfor det seneste døgn i systemet. Disse værdier vises på vejrskiverne, når interrupts fra de fysiske knapper bliver triggeret.

Hardwaren bag Min og Max knapperne, blev implementeret som 2 momentary knapper, og to pull-down modstande. Disse blev forbundet til hhv. RPI\_Indoor og veroboardet som begge er monteret inde i RPI\_Indoor etuiet. Kredsløbet for knapperne ses på Figur 46.



Figur 46: Kredsløb for Min og Max knapperne.

Til implementering af interrupt funktionalitet for knapperne blev to drivers skrevet til GPIO'erne 16 og 20, som trigger på både rising- og falling edge, så der kan sikres at min/max værdierne vises ved tryk på knappen og vender tilbage ved slip af knappen. Drivers kan findes i bilag<sup>51</sup>.

Softwaremodulet er designet så der bruges tråde, for at modulet ikke behøver at afbryde igangværende processer for at håndtere interrupts fra knapperne. Ved hvert interrupt skal

<sup>51</sup> Bilag \1 – Dokumentation\1.6 – Kode\1.6.1 – RPI\_Indoor\1.6.1.1 - Drivers



der simuleres et state skifte, hvor default state er visning af nuværende data og et sekundært state er visning af min/max.

I implementeringen af MinMax softwaremoduliet gøres der stor brug af read() systemkaldet. I driveren findes et kald af wait\_event\_interruptible() i read-implementeringen, så hver gang read() kaldes på devicet vil systemet vente med at fortsætte indtil der sker et interrupt. Essensen er derfor, at read() systemkaldet bruges blokerende så der kun sker state-skifte ved interrupt triggers. Se implementering i bilag<sup>52</sup>

Resultatet fra en modultest til eftervisning af den ønskede funktionalitet ses på Figur 47.

```
root@raspberrypi0-wifi:~/PRJ3/MinMaxmodultestv2# ./prog
Min and Max threads created
MAXBUTTON ACTIVATED: Updating stepper motor position to MAX
Møling 1 - vind: 13.1 - lux: 22.5 - temp: 57 - regn: 5.5
MAXBUTTON RELEASED: Updating stepper motor position to current weather
Møling 1 - vind: 8.3 - lux: 16.2 - temp: 42 - regn: 3.8
MINBUTTON ACTIVATED: Updating stepper motor position to MIN
Møling 1 - vind: 4.3 - lux: 8.2 - temp: 30 - regn: 2.5
MINBUTTON RELEASED: Updating stepper motor position to current weather
Møling 1 - vind: 8.3 - lux: 16.2 - temp: 42 - regn: 3.8
```

Figur 47: Resultat af modultest af MinMax modul

Det fremgår, at de høje målinger vises ved tryk på maxknappen, og lave målinger ved tryk min-knappen, hvilket opfylder kravene for UC6. Yderligere beskrivelser af modultesten af MinMax findes i bilag<sup>53</sup>.

### 8.2.8 Integrationstests af klasserne

Det vigtigste ifht. sammenspillet mellem klasserne, har været integrationstests. Det er her hvor det er blevet bevist at systemet, el. bare en del af systemet fungerer i koncert.

Accepttesten for systemet er blevet udført på baggrund af et system, som i forvejen er gennemtestet gennem disse integrationstests. Se Integrationstests bilaget<sup>54</sup> for mere information.

<sup>52</sup> Bilag \1 – Dokumentation\1.5 – Vejrviser\MinMax.pdf

<sup>53</sup> Bilag \1 – Dokumentation\1.5 – Vejrviser\MinMax.pdf

<sup>54</sup> Bilag \1 – Dokumentation\1.8 – Integrationstests\Integrationstests.pdf

## 9 RESULTATER OG DISKUSSION

Det kan observeres fra accepttesten, at det er lykkedes at implementere alle de funktionelle krav. Derimod har det ikke været muligt at godkende alle de ikke-funktionelle krav. Der er taget højde for prioriteringen af kravene, så fokus var på de vigtigste krav, der har haft en betydning for at vejrstationen er funktionsdygtig. Brugeren kan tænde systemet, og derefter vil vejrstationens vejrmåler indsamle vejrdato og sende det til vejrviseren, der vil opdatere sine vejrskiver. Videoer der demonstrerer de forskellige tests, er vedhæftet i bilag<sup>55</sup>. De ikke funktionelle krav som ikke er blevet godkendt, er overordnet set de krav, hvor det ikke har været muligt at eftervise og teste på dem.

Accepttesten består af godkendte, delvist godkendte og ikke godkendte resultater. I nedenstående tabel ses hvorvidt de forskellige scenarier/krav er godkendte eller ej. For en mere detaljeret beskrivelse af følgende refereres til bilag<sup>56</sup>, hvor selve testen, det forventede og resultaterne står noteret.

Use case nummer	Vurdering (OK/FAIL)
UC1: Tænd systemet	OK
UC2: Indsaml vejrdato	OK
UC3: Opdatér vejrviserens skiver	OK
UC4: Opdatér hjemmeside	OK
UC5: Skift format for temperaturdata	OK
UC6: Aflæs max/min værdier	OK

Tabel 2: Oversigt over resultat af accepttest for Use cases

Som det kan ses af Tabel 2, er der kun udført accepttest for de MoSCoW prioriterede **skal** og **bør** ikke-funktionelle krav. **Kunne** og **Vil ikke** krav er ikke eftertestet, men kan findes beskrevet i bilag<sup>57</sup>.

Funktionelle krav nr.	Vurdering (OK/FAIL)
1. Skal krav	OK
2. Bør krav	OK

Tabel 2: Oversigt over resultat af accepttest for funktionelle krav

Ikke funktionelle krav nr.	Vurdering (OK/FAIL)
1. Generelt	FAIL
2. Målinger	Delvist OK
3. Visninger	OK

Tabel 3: Oversigt over resultat af accepttest for ikke funktionelle krav

Systemet er derfor overordnet set funktionsdygtigt, og der er taget højde for, hvorvidt kravene, der er blevet udarbejdet, er blevet opfyldt eller ej. De generelle ikke-funktionelle

<sup>55</sup> Bilag 1 – Dokumentation\1.9 – Accepttest\Videoer til accepttest

<sup>56</sup> Bilag 1 – Dokumentation\1.9 – Accepttest\Accepttest.pdf

<sup>57</sup> Bilag 1 – Dokumentation\1.2 – Kravspecifikation\Funktionelle\_krav.pdf

krav er vurderet til at være "FAIL", da det ikke har været muligt at eftervise om sensorerne kan virke under bestemte vejrforhold. Ikke-funktionelle krav, som har med målinger at gøre, er blevet vurderet til at være "Delvist OK". Det vurderes ud fra, at kravene om hvor hurtigt der kan måles fra sensorerne er godkendt, men at kravene for målenøjagtigheden ikke kunne blive testet, og derfor er vurderet som ikke godkendt. Dette kan tilføjes som videre arbejde, så der kan verificeres at systemet er nøjagtigt.

## 10 UDVIKLINGSVÆRKTØJER

---

### **RealTerm**

For at kunne aflæse data over UART, blev der anvendt terminalen RealTerm. Dette blev gjort da RealTerm giver muligheden for at emulere interaktion med en fysisk terminal. RealTerm gør det dermed muligt at læse den data, som bliver sendt via UART'en.

### **PSoC Creator 4.4**

PSoC creator blev anvendt til programmering af PSoC microcontrolleren. Dette program blev valgt da det er specielt udviklet til at lave programmer til PSoC.

### **NI MultiSim**

For at kunne lave kredsløbsdesign for projektets hardware-dele, blev der anvendt programmet NI Multisim. Dette gøres da NI MultiSim giver brugeren mulighed for nemt at teste systemer for fejl og mangler, før de sættes op på eks. fumblebræt.

### **Analog Discovery**

Dette er et stykke hardware, som kan bruges, sammen med programmet: Waveforms, til at analysere analog og digitale signaler, samt som funktionsgenerator.

### **Waveforms**

Dette er programmet som interagerer med Analog Discovery. Waveforms er blevet brugt en del ifht. test af hardware komponenter, og især ved test af PSoC programmet, og stepperMotor funktionerne.

### **Microsoft Visio**

Visio blev benyttet til udarbejdelse af diagrammer til arkitekturen og design. Hertil blev benyttet SysML stencils til ensartet layout af diagrammerne.

### **Visual Studio**

Til at implementere vores kode har vi brugt Visual Studios.

### **Redmine**

Redmine er et project management system, vi har valgt at bruge det da det både kommer med et Agile board, men også et git repository, som blev brugt til kode-versionsstyring.

### **VMware**

Vmware er en virtual workstation der blev brugt til at udvikle på en Linux platform uden at skifte styresystem.

### **Fusion 360**

Dette program er blevet benyttet til at 3D modellere vejrskivernes, og RPI\_Indoor etui'erne.

### **Ultimaker: Cura**

Dette program er blevet benyttet til at tage 3D modellerne fra Fusion 360, og konvertere dem til G-code, så de kunne 3D printes på Hhv. Prusa og Ender 3D printere.

## 11 NYE ERFARINGER

---

Ved arbejdet med dette projekt er der opnået et række erfaringer både for specifikke faglige elementer, men også læring i forhold til processen og gennemførelsen af dette projekt. Der er arbejdet med forskellige tekniske og faglige problemstillinger til løsning og realisering af projekt hvor både indlejret Linux og PSoC er benyttet til at anvende sensorer og aktuatorer i systemet.

Ligeledes er kommunikationsgrænsefladerne for CPU'erne en vigtig del af dette projekt, hvor der både kommunikeres serielt ved brug af UART og trådløst over WLAN ved brug af en TCP-forbindelse mellem to indlejret Linux systemer (Raspberry Pi).

En erfaring ved arbejdet med indlejret Linux systemer (Raspberry Pi) i dette projekt har været at undersøge og finde løsninger til at implementere bootloading af applikationer og opsætning og benyttelse af WiFi/WLAN til TCP-forbindelsen. Der er arbejdet iterativt for at nedbringe risikoen ved implementeringen af dette samt test heraf i takt med nødvendige moduler blev udarbejdet.

Arbejdet med databasen har også givet en erfaring. Selvom undervisningen har lært os om en masse om datastrukturer, så må vi konkludere, at det ikke kan bruges i alle sammenhænge. Nogle gange er det simpelthen smartest med et simpelt array.

Benyttelsen af Scrum til at facilitere den iterative proces var en stor udfordring som har givet yderligere erfaring med kommunikationen i en projektgruppe. Til tider gavnede Scrum udviklingen og andre gange sænkede det tempoet.

Der blev benyttet git til kodedeling og koordinering af programmeringsarbejde. Dette er en stor opgradering til kommunikationsmulighederne på de forrige semestre, da deling af source kode har været utroligt besværlig.

## 12 FREMTIDIGT ARBEJDE

---

Trods der er blevet fremstillet et fungerende system gennem processen af dette semesterprojekt, så er der bestemt plads til forbedringer. Vi har som hold vokset meget, og lært endnu mere, og som projektet når sin ende, er der nu kommet refleksioner over hvad som kunne forbedre systemet, som evt. fremtidigt arbejde med systemet. Her er en liste over hvilke dele, som der kunne forbedres eller genovervejes.

### Software

- Messagesystem: Koden som kører på RPI\_Indoor, bruger et evigt loop, som data modtages og leveres til de andre klasser, som HTML, database, og stepperMotor klasserne. Problemet med dette system er at 2.delt:
  - Hvis opdateringen af indkommende data øges, så kan der blive et “race condition” problem, så dataene bliver skrevet til og læst, samtidig.
  - Det giver også fordelene at der så kunne være et kø-system, så hvis en af klasserne er bagud med at behandle dataene, så er efterfølgende data bare sat i kø, i stedet for at være gået tabt.
- Web-server udvidelse: Funktionaliteten på web-serveren er nuværende yderst begrænset. Der er dog et vælg af muligheder ifht. hvad den kunne bruges til Datafremvisning el. Styring af indstillinger for systemet som f.eks.:
  - Tidligere registreret vejrdato, via. databaseklassen, kunne f.eks. vises som graf, og der kunne være opslag af data fra specifikke perioder el. Dage.
  - Der kunne vises advarsler omkring vejret, hvis værdierne overskrider statens anbefalinger omkring vejr man ikke burde bevæge sig ud i.
  - Der kunne benyttes port forwarding, og evt. med et registreret domæne, så web-serveren kan tilgås selv når man er ude af ens hjem.
- Database-klassen: For at systemet skal være skalerbart, så skal der foretages ændringer til database-klassen, f.eks.:
  - Som det er nu, er alt data gemt som CSV-format i en tekstfil, det kunne ændres til at bruge JSON format i stedet, for at gøre det mere skalerbart.
  - Databasefilen vil kun vokse med tiden, og med evt. filstørrelse begrænsninger, så burde man skrive nye filer, og lade database leve i flere filer. F.eks. en ny fil hver uge.

### Hardware

- Stepper Motors:
  - Selv når stepper motorerne står stille, på en værdi på skiven, bruger de aktivt strøm, bare for at holde pladsen. De kunne sandsynligvis stadigvæk godt holde deres plads, selv om at der blev lukket for strømmen, og dermed spares 4x100mA/h, når de ikke er i bevægelse.
  - Systemet opdaterer pt. Vejrdato en gang i minuttet, dette kunne teknisk set øges helt ned til 1 sekund, problemet er at de udvalgte stepper motorer er gearret ned x48, så der er ret langsomme. Disse motorer kunne byttes ud med motorer som er hurtigere, så ville systemet kunne efterleve en hurtigere rate af data afbildning.
- Min og Max knapperne: Der opleves en del prel, når disse knapper benyttes, og dette gjorde koden mere udfordrende, og at funktionerne kan delvist fejle i deres formål. Ved f.eks. at tilføje en kondensator, kunne der skabes et delay på knappen, og derved kunne det undgås at få prel på kontakten.

## 13 KONKLUSION

---

Et produkt er blevet designet og udviklet således at brugeren kan finde ud af, hvordan vejret ser ud på vedkommendes egen adresse og derved vil det være langt lettere mht. tøjvalg og det er også en mere lokal måling af vejret. Der kan blot ses på vejrskiverne, og brugeren vil så have en idé om, hvordan vejret ser ud. Det er præcis denne simplicitet for brugeren, der sættes stor fokus på med dette produkt.

På baggrund af resultaterne, kan det generelt siges, at der er opnået en fungerende vejrstation som henholdsvis kan måle data, sende data videre og får vist dette både fysisk i form af vejrskiver og over en lokal hjemmeside. Så systemet har kunne interagere med omverdenen via sensorer i form af vejrmåleren som vha. PSoC har kunne måle selve dataene. Derudover har systemet funktioner til at ændre temperaturenheder, der bliver vist på hjemmesiden, og fysiske knapper, der får vejrskiverne til at vise min/max værdier. Den lokale hjemmeside og vejrskiverne er hermed brugerinterfacet.

Der har dog været krav, som systemet ikke har levet op til. Disse krav har ikke haft en betydning overordnet set, da de krav der er sat for at projektet skal være funktionsdygtigt, alle er blevet opfyldt efter accepttesten, så vejrstationen lever op til, skal kravene for projektet. Der er anvendt faglige elementer fra semesterets fag, og der blev anvendt ASE modellen i forhold til processen med projektet. Linux platformen er blevet anvendt til udarbejdelse af opstart scripts for både RPI\_Outdoor, som er en del af vejrmåleren og for RPI\_Indoor, som er en del af vejrviseren.

Projektet kan derfor i forhold til den udarbejdede problemformulering ses som værende tilfredsstillende, da de grundlæggende elementer, der har en betydning for vejrstationens virkning, er fungerende. Altså, alt i alt har det været et succesfuldt projekt, hvor en prototype til en vejrstation er blevet designet og udviklet.

Under hele denne proces har der også generelt været godt samarbejde mellem gruppemedlemmerne, hvor planlægning og kommunikation har været en essentiel del.

## 14 LITTERATURLISTE

---

Bjerger, K. (2015). *Vejledning for gennemførelse af projekt 2*. Aarhus Universitet.

Deutscher Wetterdienst. (16. 12 2021). Hentet fra Deutscher Wetterdienst:  
[https://www.dwd.de/DE/wetter/schon\\_gewusst/qualitaetvorhersage/qualitaetvorhersage\\_node.html](https://www.dwd.de/DE/wetter/schon_gewusst/qualitaetvorhersage/qualitaetvorhersage_node.html)

DMI. (2021). *Storme i Danmark*. Hentet fra DMI: <https://www.dmi.dk/vejr-og-atmosfare/temaforside-vind/storme-i-danmark/>

Friedenthal, S., Moore, A., & Steiner, R. (2008). *A Practical Guide to SysML*. Morgan Kaufmann.

Inc., N. M. (16. 12 2021). *Diagrams+ownership*. Hentet fra No Magic:  
<https://docs.nomagic.com/display/SYSMLP190/Diagrams+ownership>

Kintech Engineering. (2021). Hentet fra <https://www.kintech-engineering.com/catalogue/cup-anemometers/nrg-40/>

MIKKELSEN, P. H. (2. september 2020). Oplæg til Semesterprojektet 3. Hentet fra brightspace: <https://brightspace.au.dk/>

*Small Rain Gauge datablad*. (2021). Pronamic.

*TSL2561 Datablad*. (2021). Taos.



## 15 BILAGSLISTE

---

### 1 – Dokumentation

- 1.1 - Projektformulering
- 1.2 - Kravspecifikation
- 1.3 - System arkitektur
- 1.4 - Vejrsmålere
  - 1.4.1 - Dataindsamler
  - 1.4.2 - Datasender
- 1.5 - Vejrvise
- 1.6 - Kode
  - 1.6.1 - RPI\_Indoor
    - 1.6.1.1 - Drivers
    - 1.6.1.2 - Main Program
    - 1.6.1.3 - Modultest
    - 1.6.1.4 - Services og scripts
  - 1.6.2 - Rpi\_Outdoor
    - 1.6.2.1 - Main program
    - 1.6.2.2 - Modultest
    - 1.6.2.3 - Services og scripts
  - 1.6.3 - PSoC
    - 1.6.3.1 - Main
- 1.7 - Datablade
- 1.8 - Integrationstest
- 1.9 - Accepttest

### 2 – Organisatorisk

- 2.1 - Tidsplan
- 2.2 - Scrum
  - 2.2.1 - Product backlog
  - 2.2.2 - Sprint opgaver
- 2.3 - Mødedokumenter
  - 2.3.1 - Projektmøder
  - 2.3.2 - Vejledermøder
  - 2.3.3 - Review
- 2.4 - Logbog
- 2.5 - Procesbeskrivelse
- 2.6 - Samarbejdsaftale