AARHUS
UNIVERSITET
INGENIØRHØJSKOLEN

E4-PRJ4

Projektrapport

Aston Antony - 201505839

Andreas Blaabjerg - 201510924

Marie Unmack Bærentzen - 201608667

Casper Fevre Hansen - 201501949

Minik Nathanielsen Olsen - 201600341

Morten Sahlertz - 201410062

Saif Salih - 201609346

Andreas Lippert Tange - 201605454

Gruppe # 6

Vejleder: Lars G. Johansen

Antal tegn: 70,689

29. maj 2018

Indholdsfortegnelse

1 Versionshistorik

Dato	Initialer	Version	Afsnit	Ændringer
21.05	ALLE	0.1	Dokument oprettet	-
22.05	ALLE	0.2	Analyse, Arkitektur	Afsnit om Analyse og Arkitektur tilføjet.
23.05	ALLE	0.3	Analyse, Indledning, Forord, Krav	Afsnit om Indleding, Forord og Krav tilføjet. Analyse rettet.
24.05	ALLE	0.4	Afgrænsning, Metode og Proces, Design og implementering	Afsnit om Afgrænsning, Metode og Proces, Design og implementering tilføjet.
25.05	ALLE	0.5	Test, Resultater, Diskussion, Konklusion, Fremtidigt arbejde	Afsnit om Test, Resultater, Diskussion, Konklusion og Fremtidigt arbejde tilføjet.
26.05	ALLE	1.0	Arkitektur	Arkitektur for Battery Management System rettet. Alle afsnit tilføjet.
27.05	ALLE	1.1	Design og implementering, Test	Rettet design af spændingsregulator, Parkeringssensor. Test af strømmåler tilføjet.
28.05	ALLE	1.2	Design og implementering	Rettet modultest af GUI.

2 Ansvarsområder

I følgende tabel er det beskrevet, hvilke dele af projektet de enkelte gruppemedlemmer har arbejdet med.

Emne	**Primær**	**Sekundær**
Smartlygter SW og HW	Morten	
Parkeringssensor SW og HW	Andreas B, Casper	Saif
GUI	Andreas T	Saif
Strømmåler SW og HW	Saif	Andreas T, Morten
Spændingsmåler HW	Marie	
Spændingsmåler SW	Saif	
Forsyning	Minik	

3 Resume/Abstract

Dette projekt omhandler optimeringer på en elbil. Den specifikke elbil er en Mitsubishi iMiEV fra 2012. Der bliver identificeret tre problemer, der vil blive forsøgt løst i dette projekt. Disse er information omkring rækkevidde, hjælp til parkering og styring af forlys. Et system der måler spændingen af batteriet, strømtrækket fra batteriet, samt hastigheden af elbilen vil informere brugeren omkring den tilbageværende rækkevidde. En parkeringssensor vil med lyd informere om afstanden mellem elbilen og en forhindring. Et reguleringssystem vil regulere bilens forlygter, og sørge for at lygterne konstant har den korrekte hældning. Disse tre nævnte funktionaliteter udgør et system hver for sig. Det betyder, at dette projekt består af tre separate delsystemer.

This project is about optimizations of an electric vehicle. The specific vehicle is a Mitsubishi iMiEV from 2012. Three problems are identified, for which a solution will be attempted in this project. These are information about the range of the electric vehicle, help with parking and control of head lights. A system measuring the voltage of the battery, the current drawn from the battery, and the velocity of the vehicle will inform the user of the available range. A parking sensor will inform the user of the distance between the vehicle and an obstacle with sound. A control system will regulate the lights of the vehicle, and make sure the lights constantly have the correct angle. These three functionalities make three separate systems. Therefore this project is composed of three separate subsystems.

4 Forord

Dette dokument indeholder projektrapporten udarbejdet af gruppe 6 for kurset E4PRJ4 gennemført foråret 2018 ved Ingeniørhøjskolen Aarhus. Gruppen består af otte studerende på elektronikretningen. Gruppens vejleder er lektor Lars G. Johansen. Rapporten er afleveret ved semestrets afslutning 30. maj 2018. Rapporten udgør en del af bedømmelsesgrundlaget ved den tilhørende mundtlige eksamen d. 27. juni 2018.
Projektgruppen vil gerne takke ingeniørdocent ved Ingeniørhøjskolen Björn Andresen for hans råd og vejledning ifbm. design af et system til måling af batterikapaciteten af en elbil. Endvidere takker gruppen elektriker ansat ved Ingeniørhøjskolen Rasmus Rolighed Christensen for hans hjælp med test og montering af strøm- og spændingssensor på elbilen.

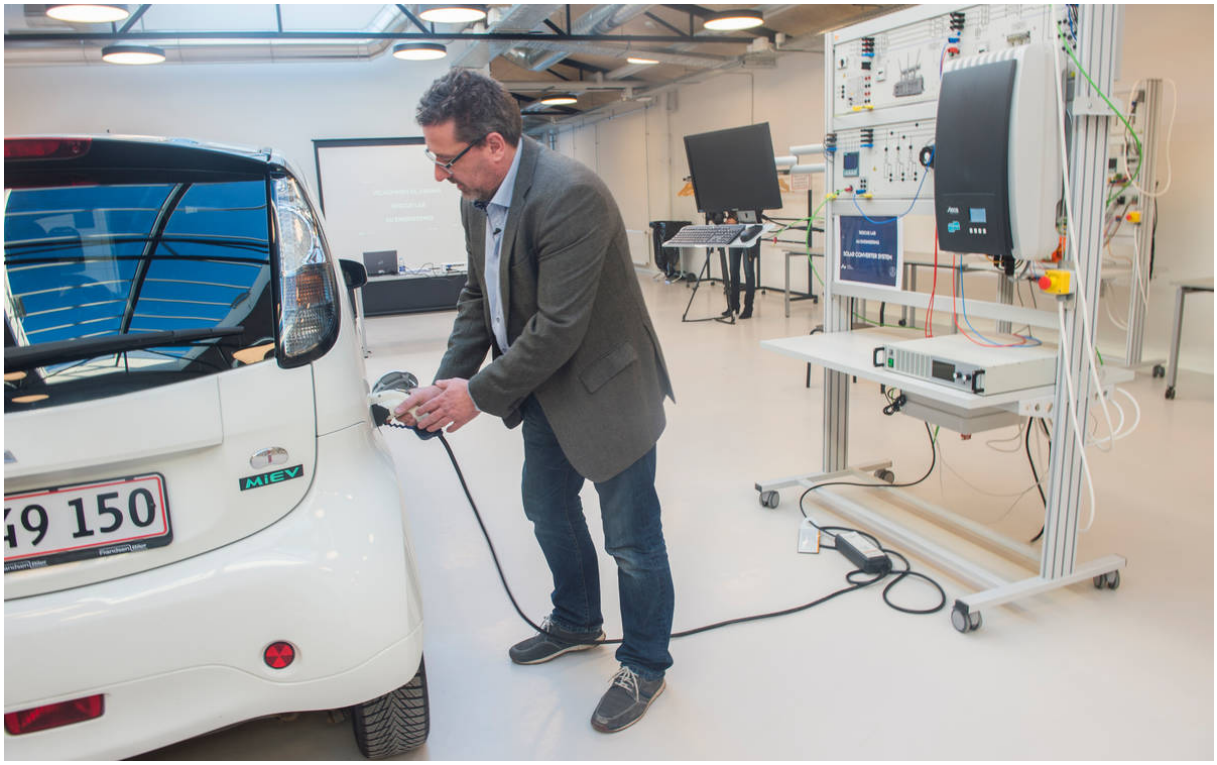
4.1 Læsevejledning

Projektrapporten er opdelt i en række afsnit, der hver især beskriver forskellige aspekter af udviklingsforløbet. De grundlæggende overvejelser omkring projektets formål og relevans præsenteres indledningsvis. Dernæst beskrives kravene til det produkt, der ønskes udviklet. Projektets afgrænsning og fokus defineres efterfølgende. Rapporten indeholder desuden en beskrivelse af de udviklingsmæssige værktøjer og metoder, som projektgruppen har taget i anvendelse. Analyse af problemdomænet, som er foretaget af gruppen tidligt i udviklingsforløbet, er også beskrevet. Tekniske beskrivelser af arkitektur, design og test af det udviklede system følger. Rapporten

afsluttes med en præsentation og diskussion af de opnåede resultater. Eventuelle mangler og ufærdige dele af projektet beskrives i et afsluttende afsnit om fremtidigt arbejde.

5 Indledning

Ingeniørhøjskolen Aarhus er indehaver af en elbil af modellen Mitsubishi iMiEV fra 2012 se figur 1. Denne elbil har det problem, at den giver en meget dårlig vurdering af antal resterende km ud fra batteriets restkapacitet. Udgangspunktet for projektet er at forsøge at udvikle et system, der kan give en bedre vurdering af restkapaciteten og dermed antal resterende km på netop denne elbil. Denne problematik danner udgangspunkt for projektet. Til projektet er endvidere tilføjet to delsystemer, som hver især repræsenterer unikke løsninger på problemer for elbiler. Disse delsystemer er hhv. et system til automatisk justering af forlys samt en parkeringssensor.



Figur 1: Opladning af Ingeniørhøjskolens elbil Mitsubishi iMiEV [22].

5.1 Problemidentifikation

Elbiler er i dag stadigvæk ikke en særlig attraktiv mulighed for de fleste bilkøbere, og mange af disse anskaffer sig derfor stadig benzinbiler. Den globale markedsandel for elbiler var i 2016 kun lidt mere end 1% [7]. En af grundene til dette er måske, at elbiler mangler gavnlige og interessante funktioner, der kan gøre dem mere attraktive for almene bilkøbere.
Der er en række muligheder for forbedringer af Mitsubishi iMiEV. Dette omfatter en mere nøjagtig måling af batteriets restkapacitet, smart parkeringshjælp og automatisk justering af forlys. Endvidere er mange elbiler ikke udstyret med en grafisk brugergrænseflade, der informerer føreren omkring bilens resterende rækkevidde i km. En sådan grænseflade kan også give forslag

til, hvordan føreren kan øge rækkevidden ved optimal kørsel.
Forslag til forbedringer af de elektriske systemer i en elbil er ikke kun begrænset til de ovennævnte eksempler idet en lang række andre kan foreslås:

- Solceller der monteres på elbilen.
- Forbedringer af batteriteknologi så elbilens rækkevidde og opladetid øges.
- Elektronisk reguleret affjedring.

5.1.1 Projektets fokus

I dette projekt er kun et lille udpluk af disse problemer blevet udvalgt til analyse, design og implementering. De fire problemer der er blevet udvalgt er følgende:

1. Batterikapacitet og estimering af rest-rækkevidden – Måling af elbilens restkapacitet.

   Mange elbiler har blot en simpel måling af den aktuelle restkapacitet i batterierne. Dette kan være et stort problem for føreren af bilen, hvis systemet f.eks. overestimerer antal resterende km. Løsningen er et mere avanceret system, der ud fra måling af restkapaciteten og aktuel kørsel giver en estimering af rest-rækkevidden.

2. Information og forslag til bilens fører – Optimering af kørslen.

   Mange elbiler er ikke udstyret med en brugergrænseflade, der muliggør at forskellige typer af information, såsom information om energiøkonomi og kørselsmønster kan vises. En sådan brugergrænseflade vil forbedre kørselsoplevelsen væsentligt.

3. Forlys der justeres automatisk, så de ikke blænder — Hældning og længde på nærlys.

   Mange bilister har oplevet at blive blændet af en modkørende bil, som kører over en bakke. Derfor ønskes der udviklet et system, som automatisk justerer bilens forlygter, således at de ikke lyser længere end det anbefalede i specifikationerne. Denne styring skal desuden styre hældningen af lyset, da dette også er en del af specifikationerne i en almindelig bil.

4. Parkeringssensor – Gør det lettere at parkere.

   Når det er tid til at parkere, er det ikke altid lige let, da der er en række blinde vinkler for bilisten. Løsningen på dette er en parkeringssensor, som kan gøre bilisten opmærksom på genstande bilen er ved at påkøre. Ved at have parkeringssensor på bilen, vil man kunne undgå skader, der opstår ved parkering på egen hånd.

5.2 Problemløsning

Projektet har fokus på udvikling af tre delsystemer under hovedtemaet elektriske systemer i en elbil. De tre delsystemer benævnes under ét for Electric Vehicle Manager (EVManager):

1. **Battery Management System med grafisk brugergrænseflade (GUI)**

   En enhed der måler den resterende kapacitet i elbilens batteri. Kapacitetsmåleren tænkes udviklet vha. spændings-, strøm- og hastighedsmåleudstyr. Måling af hastighed er nødvendig, da man gerne vil sammenholde data for effektforbruget og køretøjets hastighed og rækkevidde.

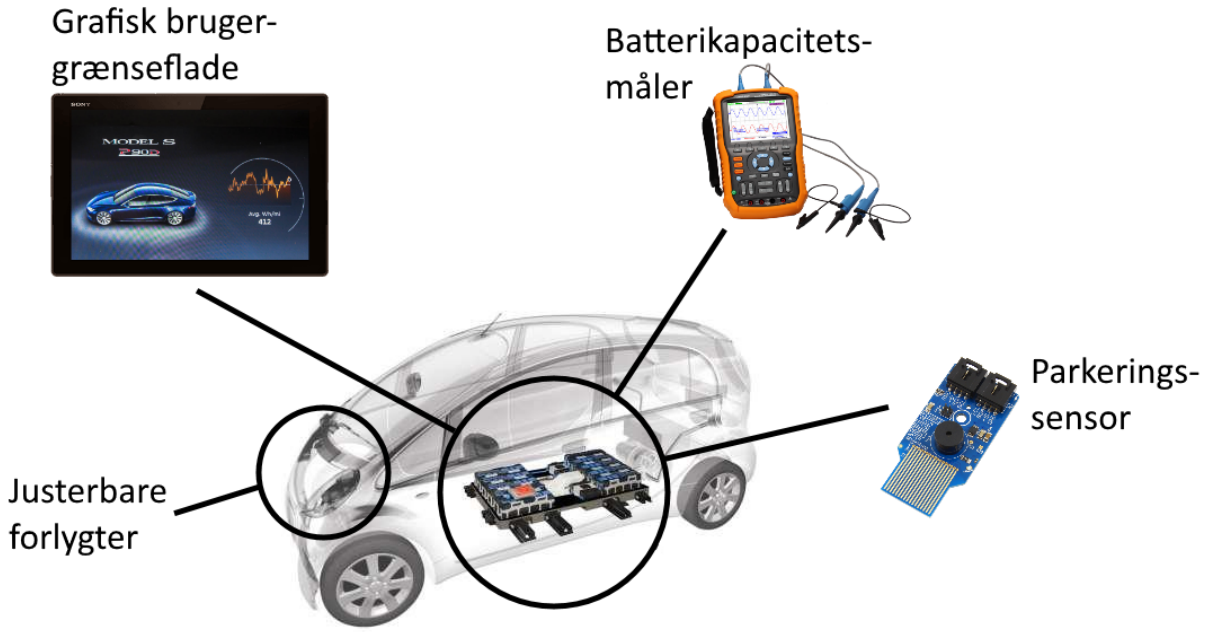
På baggrund af målinger af batteriets restkapacitet samt kørselsmønster, skal en grafisk brugergrænseflade informere føreren om energiøkonomi, batteriets restkapacitet, elbilens rækkevidde i km samt feedback til brugeren.

2. **Smartlygter**

   Forlygter hvis vinkel korrigeres automatisk, så de ikke blænder modkørende. Forlygterne korrigerer, således at de ikke lyser længere end det specificerede. Denne del af projektet tænkes implementeret som en separat prototype, der ikke forsøges implementeret på selve elbilen.

3. **Parkeringssensor**

   Parkeringssensor som måler bilens afstand til objekter ved bilens bagende. Hvis bilen kommer for nær en forhindring, vil føreren blive informeret om dette. Denne del af projektet tænkes også implementeret som en separat prototype, der ikke forsøges implementeret på selve elbilen.



Figur 2: Skitse af de tre delsystemer, der tænkes implementeret på elbilen [18].

6 Krav

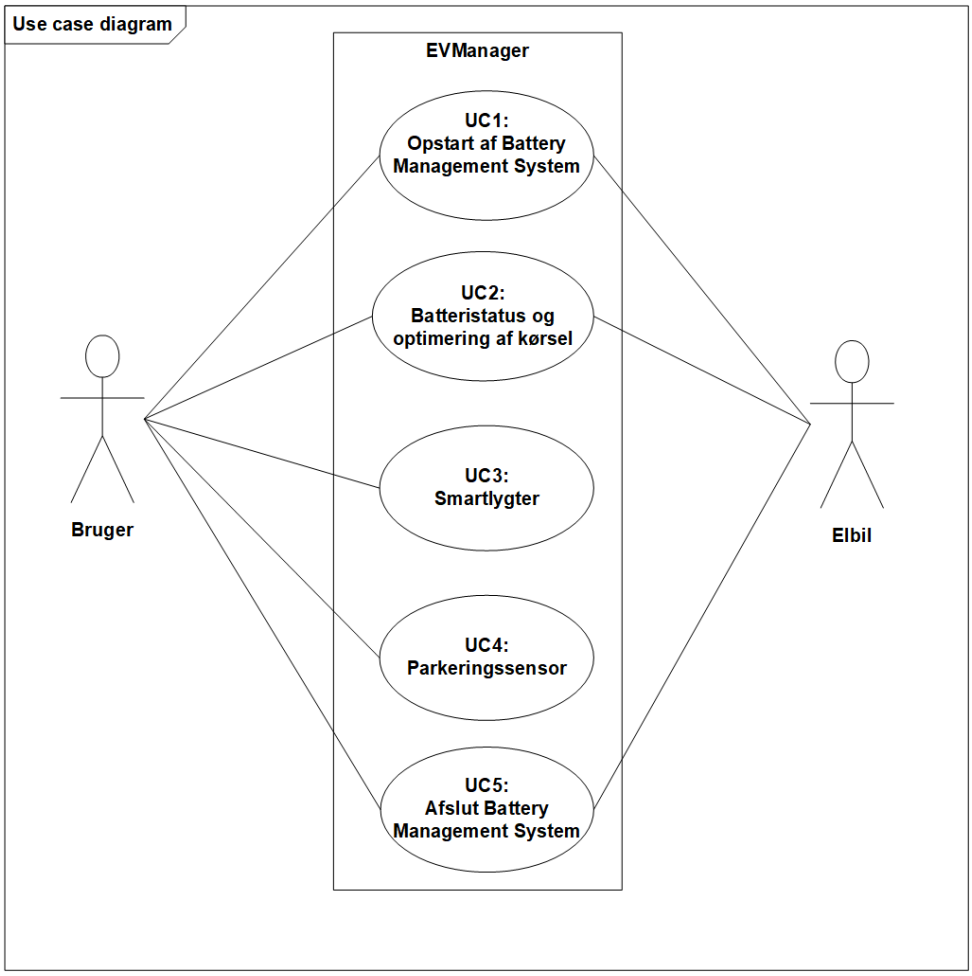
Kravene til EVManager er opdelt i funktionelle og ikke-funktionelle krav. De funktionelle krav er defineret vha. use cases. De ikke-funktionelle krav er opstillet som målbare krav til systemet efter en række forskellige parametre.

6.1 Aktørbeskrivelse

For EVManager er der defineret to aktører. Systemets primære aktør er Bruger. Bruger er den person, som kan anvende de forskellige funktioner i EVManager. Bruger kan i almindelighed opfattes som føreren af elbilen, eller som betjener af EVManagers forskellige delsystemer, i form af Smartlygter eller Parkeringssensor. Elbil er systemets sekundære aktør. Elbil er Mitsubishi iMiEV, hvorpå dele af EVManager er implementeret. Elbil er at opfatte som en sekundær aktør, da den har både software og hardware grænseflader til EVManager. F.eks. aflæser EVManagers Battery Management System spænding og strøm fra Elbilens batteri.

6.2 Use case beskrivelse

I det følgende beskrives de forskellige use cases kortfattet. Oversigt over de forskellige use cases og aktørernes relation til dem kan ses på figur 3.



Figur 3: Use case diagram.

6.2.1 Use case 1: Opstart af Battery Management System

Brugeren befinder sig i dette scenarie i Elbilens førersæde. Brugeren ønsker at starte Battery Management System. Dette sker ved at Brugeren indsætter bilnøglen, og sætter den i "on" position. Derved startes Battery Management System, og Brugeren kan anvende den grafiske brugergrænseflade.

6.2.2 Use case 2: Batteristatus og optimering af kørsel

Brugeren befinder sig i dette scenarie i Elbilens førersæde. Scenariet forudsætter at use case 1 er gennemført. Brugeren ønsker at anvende Battery Management Systemets grafiske brugergrænseflade (GUI) til at opnå information om energiøkonomi, batteriets resterende kapacitet, elbilens rækkevidde samt få feedback om kørslen. Ved kørsel opdateres information på GUI'en kontinuerligt, således at Brugeren ajourføres om Elbilens tilstand.

6.2.3 Use case 3: Smartlygter

Brugeren ønsker at starte Smartlygter. Dette sker ved hjælp af en knap. Herefter vil sensoren i systemet foretage målinger og lygterne vil vinkles op eller ned alt efter de målte data fra sensoren.

6.2.4 Use case 4: Parkeringssensor

Brugeren ønsker at anvende parkeringssensor til at opnå viden om afstanden til objekter. Brugeren tænder for parkeringssensor og kan nu bruge lyden, som modulet udsender til at vurdere afstanden til objekter. Ved afstande under 1,5 m vil parkeringssensoren ikke udsende en lyd. Når afstanden bliver mindre, vil frekvensen af lyden gradvis blive øget.

6.2.5 Use case 5: Afslut Battery Management System

Brugeren befinder sig i dette scenarie i Elbilens førersæde. Scenariet forudsætter at use case 1 er gennemført. Brugeren ønsker at afslutte Battery Management System. Dette sker ved at Brugeren drejer Elbilens nøgle fra "on" til "off" position. Derved afsluttes Battery Management System.

6.3 Ikke-funktionelle krav

For EVManager er der defineret syv ikke-funktionelle krav. De er som følger:

- Battery Management System skal kunne måle strømme op til 300 A DC.
- Battery Management System skal kunne måle spændinger op til 400 V DC.
- Smartlygter skal kunne justere lysvinklen med en opløsning på 2 ° med en usikkerhed på under 10%.
- Smartlygter skal kunne måle afstande fra 1 m til 40 m.
- Smartlygter skal være i stand til at sende data om målt afstand mindst 1 gang i sekundet.
- Parkeringssensor skal kunne måle afstande fra 2 cm til 400 cm.
- Parkeringssensor skal kunne måle afstand med en afvigelse på mindre end 10%.

7 Afgrænsning

En MoSCoW analyse [20] er blevet anvendt til at fastlægge prioriteringen af kravene til EVManager. De vigtigste krav kategoriseres som "Must", hvorimod funktionalitet som under ingen omstændigheder vil blive implementeret kategoriseres som "Won't". I kategorierne herimellem ("Should" og "Could") findes krav med lavere prioritet.
I kravene til systemet er der specificeret fem use cases og ud fra MoSCoW analysen, er det fastlagt, at use case 2, 3 og 4 har højeste prioritet. Use case 1 og 5 har lavest prioritet. Af de use cases med højest prioritet, har use case 2 den allerhøjeste prioritet, da den omfatter langt de fleste "Must" krav fra MoSCoW analysen. Denne use case omfatter også det mest omfattende delsystem i EVManager i form af Battery Management System.
Systemets ikke-funktionelle krav er alle vægtet lige højt, da de hver især vedrører centrale aspekter ved de tre delsystemer. De ikke-funktionelle krav opfattes som essentielle for at sikre, at de udviklede systemer overhovedet er anvendelige.

8 Metode og proces

8.1 SysML og UML

Til udfærdigelse af arkitekturen for EVManager er SysML [1] og UML [2] blevet anvendt. Disse metoder anvendes til at udarbejde diagrammer, der beskriver de forskellige dele af arkitekturen. I SysML standarden findes diagrammer til forsimpling af produktet til mindre bestanddele med lav kobling og høj samhørighed. Diagrammerne kaldes henholdsvis block definition diagram og internal block diagram (bdd og ibd). Bdd anvendes til definition af indholdet af en hardwareblok. Ibd anvendes til at vise forbindelserne mellem underblokke, som tidligere er vist i det tilhørende bdd. Her defineres type og antal af hver forbindelse mellem underblokkene, og dermed skabes overblik over grænsefladerne.
De UML diagrammer der anvendes i denne rapport er udelukkende til beskrivelse af software. Dette gælder sekvens- og klassediagrammer. Sekvensdiagrammer bruges til at vise den sekventielle interaktion mellem objekter, der f.eks. kan være instanser af en klasse. Klassediagrammer anvendes til at give overblik over relevante klasser og deres relationer.

8.2 Udviklingsmodel

Under arbejdet med dette projekt er ASE's model til udvikling af hardware og software blevet anvendt. ASE modellen er en semi-iterativ model hvor projektformulering, kravspecifikation, accepttestspecifikation og systemarkitektur laves først i den nævnte rækkefølge og uden overlap. Dernæst påbegyndes en iterativ fase, hvor der på kryds og tværs laves design og implementering-stest for de forskellige hardware- og softwaredele. I denne periode af udviklingen arbejdes der iterativt med de forskellige dele af systemet. Til slut påbegyndes endnu en lineær fase i projektet, hvor der gennemføres integrationstest af de forskellige hardware- og softwaredele samt acceptest ud fra accepttestspecifikationen.

8.3 Scrum

Scrum er anvendt i projektet via onlineplatformen Redmine [29]. I løbet af projektet er der lavet sprints på 7 dage fra mandag til mandag. På Redmine blev der opstillet tasks, hvor "progress" og "burndown" for hver task og sprint fremgik. Et medlem af projektgruppen blev udvalgt til, som "Scrum master", at sørge for organisering og fordeling af arbejdsopgaver. Desuden blev de roller som findes i Scrum også uddelt til forskellige gruppemedlemmer. Rollerne i Scrum blev ikke fulgt fuldkomment, men blev snarere opslugt af nogle andre roller, som gruppen havde defineret selv i begyndelsen af forløbet.
Igennem forløbet blev der også afholdt møder hver uge. Hver mandag morgen blev der holdt Scrum-møde, hvor der blev samlet op på det foregående sprint. Hver onsdag og torsdag blev der holdt arbejdsmøder, hvor de nedsatte arbejdsgrupper kunne samarbejde om de opgaver, de hver især var blevet tildelt. Der blev også afholdt vejledermøder inden et Scrum- eller arbejdsmøde. I nogle tilfælde blev der også holdt møder med andre undervisere på Ingeniørhøjskolen Aarhus. Dette skyldtes arbejdet med høje spændinger og strømme, der krævede involvering af bl.a. en elektriker.

9 Analyse

9.1 Måling af elbilens batterikapacitet

Udfordringen ved at måle restkapaciteten er, at dette skal gøres real-time. Normalt kaldes restkapaciteten "state-of-charge" (SoC) [32]. SoC skal forstås som batteriets nuværende kapacitet udtrykt som en procentdel af den totale kapacitet angivet i Ah eller Wh. Der er flere metoder til at estimere SoC, men størrelsen kan ikke måles direkte.

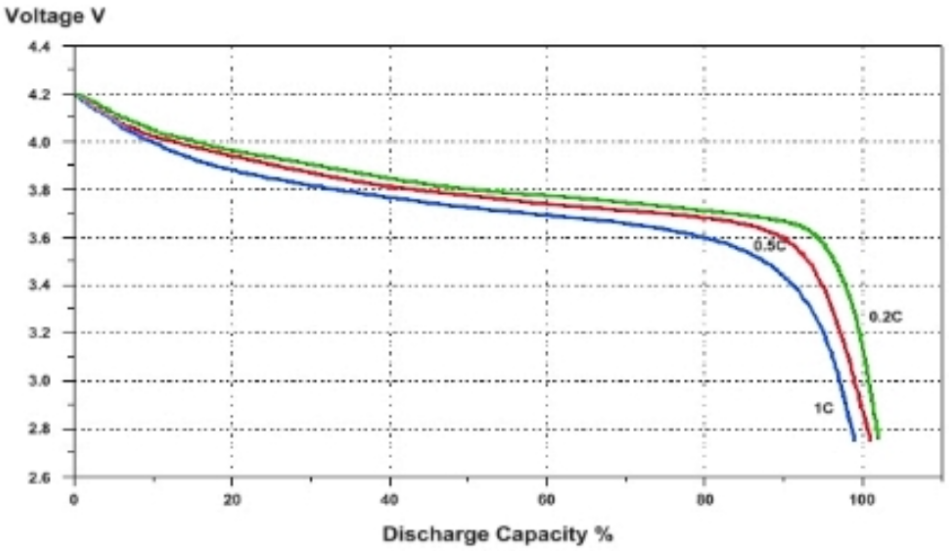
9.1.1 Funktionalitet

Et system der kan måle restkapaciteten af elbilens batteri og på den baggrund komme med en vurdering af bl.a. elbilens rækkevidde ønskes udviklet. Funktionaliteten kan deles op i to overordnede problemstillinger:

- Mål aktuel resterende kapacitet af batteriet.
- Vurder antal resterende km ud fra resterende kapacitet.

9.1.2 Estimering af SoC ud fra batterispændingen

Alle batterier har en karakteristik afladningskurve se figur 4 for et eksempel.



Figur 4: Typisk afladningskurve for Lithium-ion (li-ion) batteri.[35]

Afladningskurven afbilder batterispændingen som funktion af SoC. Hvis den karakteristiske afladningskurve for batteriet er kendt, kan SoC dermed estimeres ud fra batterispændingen. Det er dog temmelig unøjagtigt, da afladningskurven for f.eks. li-ion batterier er flad over et stort stykke, hvorefter spændingen falder pludseligt. Den elbil som systemet er tiltænkt anvender netop li-ion batterier [12].

9.1.3 Estimering af SoC ved strøm-integration

En anden metode til estimering af SoC er at måle strømmen I, der trækkes fra batteriet og integrere over tid:

$$\text{Forbrugt kapacitet} = \int_0^t I\text{dt} \; [\text{A} \cdot \text{s}]$$

En sådan metode forudsætter, at man kender det sted hvor SoC er 100% (f.eks. en given batterispænding). Dermed kan SoC beregnes som:

$$\text{SoC} = \frac{\text{Total kapacitet} - \text{Forbrugt kapacitet}}{\text{Total kapacitet}}$$

Denne metode har en række ulemper. F.eks. vil fejl blive akkumuleret (på grund af integralet). Derudover er der tale om en open-loop målemetode, så der er risiko for drift. Man kan forbedre metoden ved også at måle spændingen, og hvis afladningskurven er kendt, kan man bruge den til at korrigere for fejl pga. drift. Endvidere er det nødvendigt at anvende en numerisk algoritme til at estimere integralet af strømmen - dette vil også give ophav til fejl. Det er nærliggende at anvende en metode som f.eks. Newton-Cotes [21] idet strøm og spænding kan samples med et fast interval. Men en trapez-integrationsmetode vil også fungere fint [34].

9.1.4 Algoritme til beregning af rækkevidde

Elbilens rækkevidde vil afhænge af en lang række faktorer som f.eks. [17]

- Hastighed
- Kørselsforhold
- Aerodynamik
- Vægt
- Bakker
- Temperatur
- Kørestil
- osv...

Der vil som udgangspunkt først blive udtænkt en simpel algoritme til at vurdere antallet af resterende km. Til udregning af elbilens totale rækkevidde kan følgende metode anvendes:

1. Gennemsnitshastigheden i et vist tidsinterval beregnes.
2. Gennemsnitseffekten i samme tidsinterval beregnes.
3. Den resterende kapacitet divideres med gennemsnitseffekten, og derved bestemmes den resterende tid.
4. Den resterende tid ganges med gennemsnitshastigheden, og derved bestemmes den resterende rækkevidde.

Den gennemsnitlige hastighed og effekt i et vist interval kan beregnes effektivt med et eksponentielt midlingsfilter af formen

$$y[n] = \alpha x[n] + (1 - \alpha) \cdot y[n - 1]$$

Størrelsen af α der svarer til et almindeligt midlingsfilter med n antal samples kan beregnes med formlen [19]

$$\alpha = \frac{2}{n + 1}$$

Samples der f.eks. 1 gang i sekundet og man ønsker gennemsnittet af de seneste 60 samples fås

$$\begin{aligned} \alpha &= \frac{2}{60 + 1} \\ &= 0,033 \end{aligned}$$

Dette er en simplificeret metode, som ikke tager højde for de tidligere nævnte faktorer, der påvirker rækkevidden. Samtidig er der i algoritmen et problem med den resterende kapacitet. Denne værdi er svær at håndtere, når der slukkes og tændes for bilen, da det ikke kan vides om kapaciteten er faldet eller steget den tid hvor bilen var slukket - dvs. hvis batteriet er blevet op- eller afladt uden systemet har været tilsluttet. Løsningen på dette problem er at estimere kapaciteten ud fra batterispændingen hver gang systemet tændes:

1. Hvis batterispændingen er det samme som før systemet blev afbrudt, kan det antages at batteriet hverken er blevet opladt eller afladt.

2. Hvis batterispændingen er højere eller lavere kan den resterende kapacitet estimeres ud fra den karakteristiske spændingskurve som på figur 4.

Den totale kapacitet har også dette problem, og det kan i denne algoritme kun antages, at den har en konstant værdi, som ikke ændrer sig over tid eller afhænger af f.eks. temperaturen.

9.1.5 Valg af strømmåler

Til måling af den strøm der trækkes fra elbilens batteri, er der valgt en hall effekt sensor af modellen LEM DHAB S/133 [14]. Denne sensor kan iflg. databladet måle strømme op til 750 A. Valget er faldet på netop denne hall effekt sensor af flere årsager. Iflg. specifikationerne for Mitsubishi iMiEV [12] er elmotorens maksimale output 49 kW og den maksimale batterispænding 330 V. Ud fra dette kan en vurdering af det maksimale strømtræk, når bilen kører findes som

$$\frac{49 \text{ kW}}{330 \text{ V}} = 148 \text{ A}$$

Det maksimale strømtræk er altså omkring 150 A. Dette gælder dog ikke ved antændelsen, hvor strømtrækket givetvis er meget højere. Den valgte sensor har altså et måleområde, der er passende. Endvidere er denne sensor indpakket i en plastikkasse, der giver en vis beskyttelse imod fugt og smuds. Dette er vigtigt, da sensoren skal monteres under bilen, hvor den vil blive udsat for det danske klima.

9.2 Parkeringssensor implementeret med sonar

En sonar er en enhed, der kan udsende og modtage ultralydsbølger. Denne kan bruges til at bestemme en afstand. Idéen her er at hjælpe brugeren med at forhindre skader på elbilen fra uventede objekter under parkering. Sonaren arbejder uden for det lydspektrum, som det menneskelige øre opfanger, hvilket generelt er fra 20 Hz til 20 kHz. Vi arbejder derfor i et højere frekvensspektrum, som kaldes ultralyd også kendt som højfrekvent lyd.

9.2.1 Valg af Sonar

Til valg af sonar har vi enkelte muligheder i forhold til hvilke der indehaves af IHA Embedded Stock. Disse er lettilgængelige for os. Valget landede på at kigge nærmere på HC-SR04 [36]. Beslutningen bærer primært præg af sonarens funktionslængde. Ved en parkeringssensor vil der blive arbejdet med relativt korte ultrasoniske signaler, så derfor er denne et godt valg, da den er specificeret til at fungere inden for 2 cm - 400 cm. Der er yderligere en del materiale i forhold til lige netop HC-SR04 og brugen af Raspberry Pi, hvilket uden tvivl vil være en fordel, når systemet skal implementeres.
Sonaren HC-SR04 har den fordel at den ikke har nedsænket funktionalitet når den påvirkes af direkte sollys, i modsætning til Sharps Rangefinder Modul [30]. Dog har sonaren den ulempe, at den kan have svært ved at opfange ikke-akustiske objekter, som f.eks. stof og andre støjdæmpende materialer.

9.3 Graphical User Interface (GUI)

Målet med at have en GUI er at kunne orientere føreren af bilen om en række forskellige forhold herunder resterende batterikapacitet og bilens rækkevidde. GUI'en tænkes realiseret vha. en touchskærm og en embedded platform. Dette sikrer, at systemet kan monteres et sted på elbilens instrumentbræt.

9.3.1 Valg af embedded platform

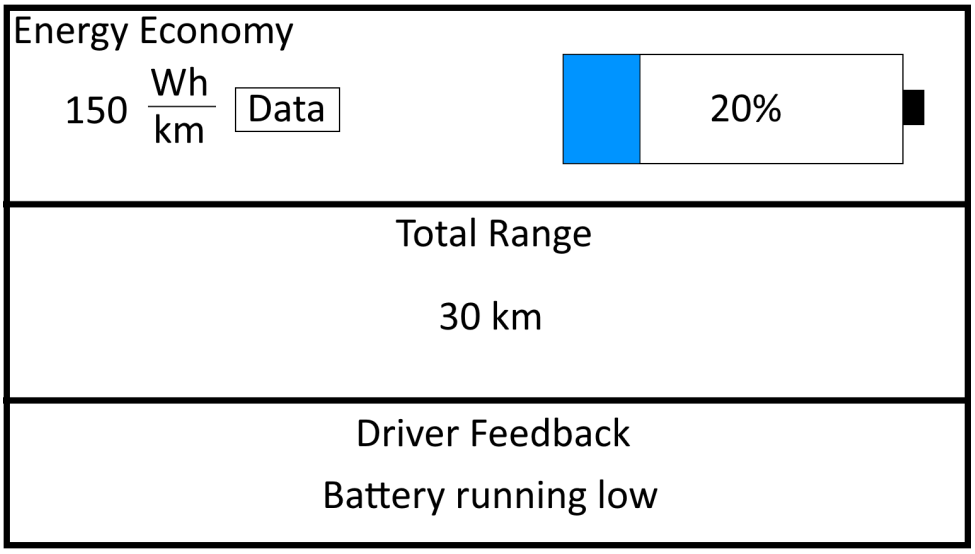
Valg af embedded platform til implementering af GUI afhænger af en række faktorer. Den embeddede platform er nødt til at have mulighed for at kommunikere med de forskellige hardwareenheder, som den skal modtage data om batteriets strøm og spænding fra. Endvidere skal den have nok processorkraft til at kunne køre en grafisk brugergrænseflade uden problemer. Mulighed for at anvende en touchskærm, der fungerer som plug-and-play ville også være en fordel. Med udgangspunkt i ovenstående overvejelser er valget faldet på Raspberry Pi 2 [13]. Som styresystem anvendes den officielle Linux distribution fra Raspberry Pi "Raspbian OS" [28].

9.3.2 Valg af skærm

Skærmen er den enhed hvorpå GUI'en vises. Da brugeren også skal være i stand til at interagere med GUI'en, er ønsket fra projektgruppens side, at den skal have touch-funktionalitet indbygget. Endvidere skal skærmen være af en passende størrelse, således at den kan monteres i umiddelbar nærhed af førersædet i elbilen. Under hensyntagen til ovenstående kriterier er valget faldet på det officielle 7" Raspberry Pi Touchscreen Display [27]. Dette display har den rigtige størrelse og er desuden plug-and-play med Raspbian.

9.3.3 Valg af toolkit til programmering af GUI

Efter undersøgelse af de forskellige muligheder er valget faldet på Qt [25]. Dette skyldes bl.a., at Qt giver mulighed for at implementere GUI'en i højniveau-sproget Python vha. bindings, der wrapper C++ koden fra Qt. Valget af Python bindings til Qt er faldet på PyQt [24]. Endvidere findes programmet Qt Designer, der tillader grafisk design af GUI ved drag-and-drop af designelementer. Dette gør at udviklingstiden for GUI'en forhåbentlig kan skæres ned. En skitse af GUI'en kan ses på figur 5.



Figur 5: Skitse af GUI'en for Battery Management System.

9.4 Smartlygter

9.4.1 Måleenhed

Det er vigtigt at have noget, der kan måle afstand på bilens nærlys, da lyset efter specifikationerne skal lyse mindst 30 m [33]. For at de automatiserede forlygter kan fungere, skal der findes en sensor, som kan måle minimum 30 m frem med en fornuftig nøjagtighed. Dette kan gøres med en laser. Problemet med at finde en sådan laser er, at der er mange lasere, der kan, men kun har et display som output. Da vi skal sørge for at forlygterne skal hælde automatisk nedad, når man når toppen af en bakke, er det nødvendigt at programmere en motor til at vende lygterne nedad, men til dette er der behov for et serielt interface fra måleredskabet. Dette kan være svært at finde, og kan muligvis give problemer, hvis en lasermåler skal udvikles fra bunden, eller en laser skal modificeres til at give en værdi, der kan bruges som serielt interface. Hertil er der derfor valgt en Laser Distance Sensor 701A-40 [15], som kan måle den nødvendige længde, samt give os målingsdata tilbage, ved hjælp af UART kommunikation.

Problematikker med den valgte laser, er at den bliver let påvirket af omgivende lys, og måletiden er ikke hurtig i forhold til real-time. Dette kan skabe et delay i reguleringen. En hel optimal laser til systemet kunne have været VDM100-50 [10]. Valget af 701A-40 er blevet taget på grund af økonomiske årsager.

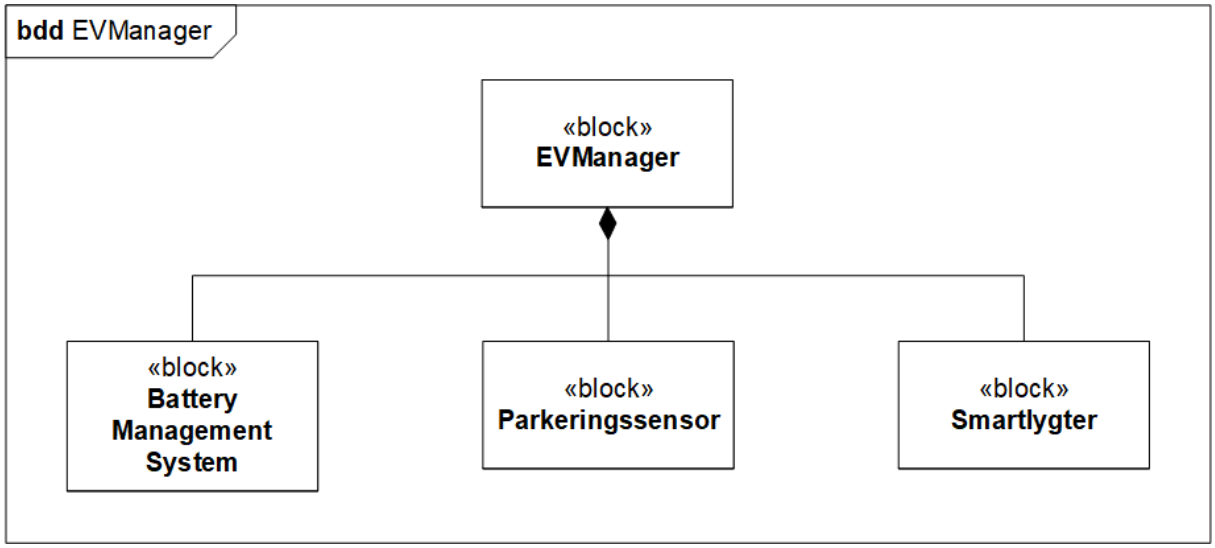
9.4.2 Motorstyring

Ud over målehenheden skal der installeres motorstyring for nærlyset. Dette kan gøres med en DC motor, en servo motor, eller en step-motor. Den sidstnævnte kan styre vinkeldrejningen af nærlyset, hvor at der ved de to andre motorer, også skulle være et gyroskop installeret før dette var muligt. Ved en step-motor, er der en vinkel per step, som kan bruges til at sørge for at vinklen på lyset er optimal. Lyset bliver sat til at starte med at være vandret, og derefter kan

step-motoren styre vinklen. step-motoren, skal derfor være præcis, og bliver valgt ud fra dens step vinkel og step præcision.

10 Arkitektur

Beskrivelsen af arkitekturen for systemet er opdelt i et afsnit for hvert delsystem. Dvs. både hardware og software for hvert delsystem beskrives samlet. Systemet EVManager består overordnet set af de tre separate delsystemer Battery Management System, Smartlygter og Parkeringssensor. Overordnet bdd for EVManager kan ses på figur 6.

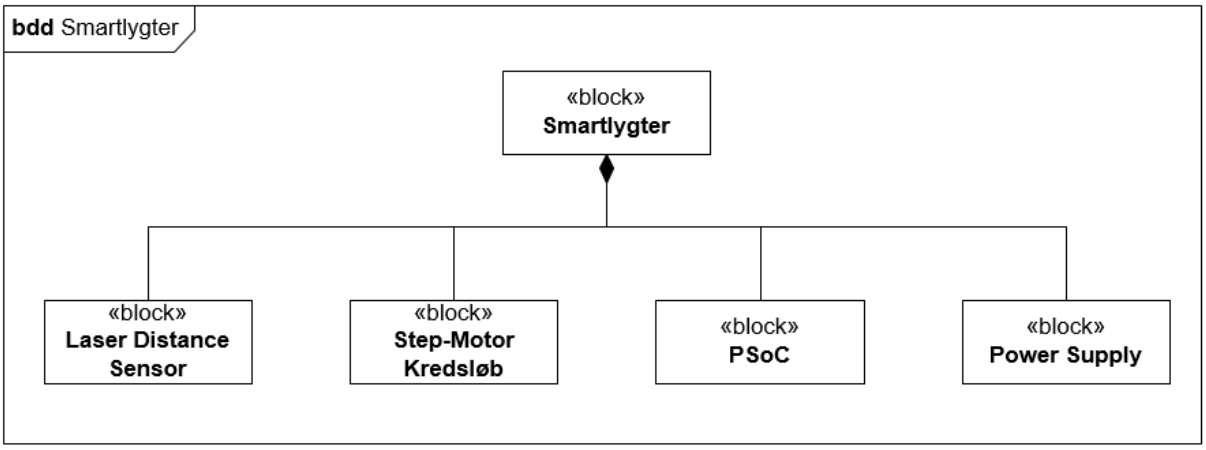


Figur 6: Bdd for EVManager der består af 3 uafhængige delsystemer: Battery Management System, Parkeringssensor og Smartlygter.

10.1 Smartlygter

10.1.1 Hardwareblokke

Smartlygter systemet består af de følgende hardwareblokke: Laser Distance Sensor, Step-motor Kredsløb, PSoC og Power Supply. Bdd for Smartlygter kan ses på figur 7.

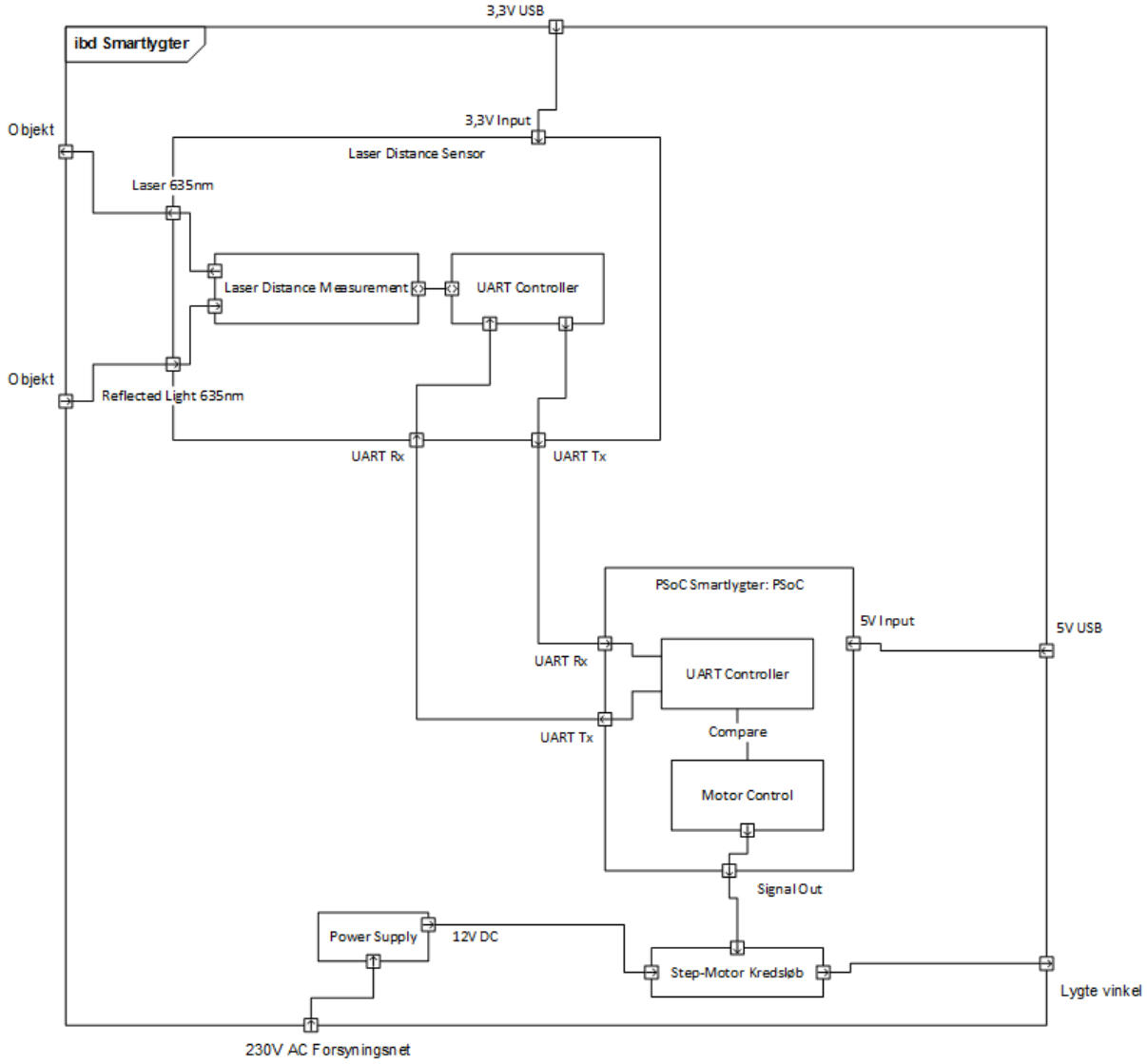


Figur 7: Bdd for Smartlygter

Laser Distance Sensor står for at for at måle afstanden. Dette skal indikere, hvor langt vores nærlys lyser.
Smartlygter PSoC:PSoC er den hardware enhed, som sørger for at behandle data modtaget fra **Laser Distance Sensor**.
Step-motor kredsløb er den hardwareblok, som giver step-motoren sin funktionalitet. Funktionaliteten bliver styret af **PSoC**'en.
Power Supply leverer spænding og strøm til step-motoren.

10.1.2 Interne hardwareblokke

For at få overblik over de interne forbindelser mellem de forskellige hardwareblokke i Smartlygter systemet, er der lavet et ibd. Dette ibd kan ses på figur 8.

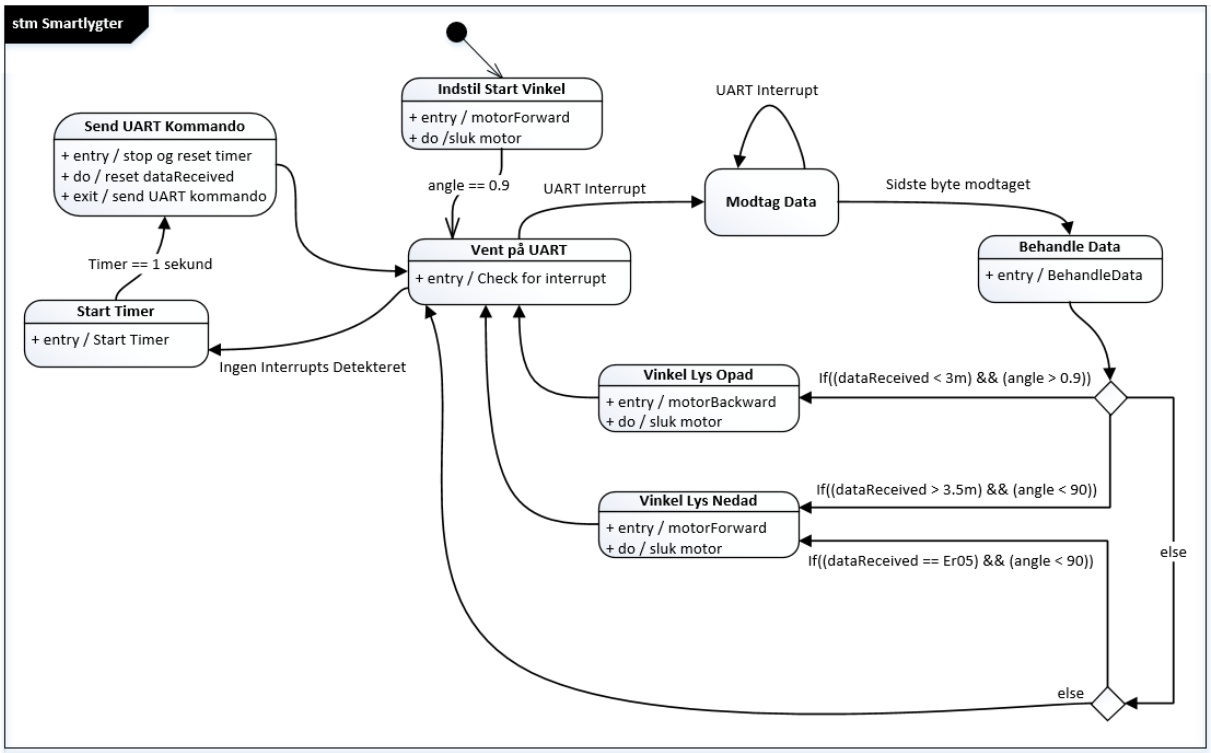


Figur 8: Ibd for Smartlygter

Laser Distance Sensor står for at sende en laserstråle afsted og måle på det reflekterede lys for at måle afstanden. Derefter sendes den udregnede målte værdi til **PSoC**'en.
Kommunikationen mellem **Laser Distance Sensor** og **PSoC** foregår via UART.
Smartlygter PSoC: PSoC er den microprocessor, som står for at styre behandlingen af signalet fra **Laser Distance Sensor**. Programmet styrer desuden også signalet, som bliver sendt for at styre **Step-motor Kredsløb**.
Step-motor kredsløb styrer step-motoren og dermed vinklingen af nærlyset Power Supply får sin forsyning fra el-nettet og transformerer indgangs AC spændingen til den nødvendige DC værdi for step-motoren

10.1.3 Afvikling af kode på PSoC

Koden på PSoC afvikles i et uendeligt loop se figur 9. Ved opstart af programmet indstilles vinklen på nærlyset, og herefter begynder den at vente på et UART interrupt.



Figur 9: State Machine Diagram for PSoC: Smartlygter

Indstil startvinkel Dette er den indledende state for PSoC programmet. Her bliver start vinklen på smartlygterne indstillet. Herefter forlader den staten.

Vent på UART Ved indgangen til denne state bliver der tjekket for UART interrupt, hvis der kommer et interrupt forlader den denne state og går til **modtag data**.

Modtag data I denne state bliver den modtagne UART data fra Laseren lagt over i et array, som hedder dataReceived.

Behandle data i denne state, har PSoC'en modtaget en UART besked, som skal behandles. Hvis den sendte besked er en målt længde, tjekkes der for, hvor langt der er blevet målt og medmindre, det er inden for det valgte interval, vil den gå over i en af de tre condition bestemte states.

Vinkel lys opad I denne state er der modtaget en målt længde fra laseren, som er mindre end det ønskede interval og lygterne vil derfor vinkles opad, så der kan lyses længere. Dette sker ved indgangen til staten. Efter dette slukkes motoren.

Vinkel lys nedad I denne state er der modtaget en målt længde fra laseren, som er højere end det ønskede interval, og lygterne vil derfor vinkles nedad, så der ikke lyses så langt. Motoren

vinkler lyset nedad ved indgangen til denne state og efter dette slukkes motoren.

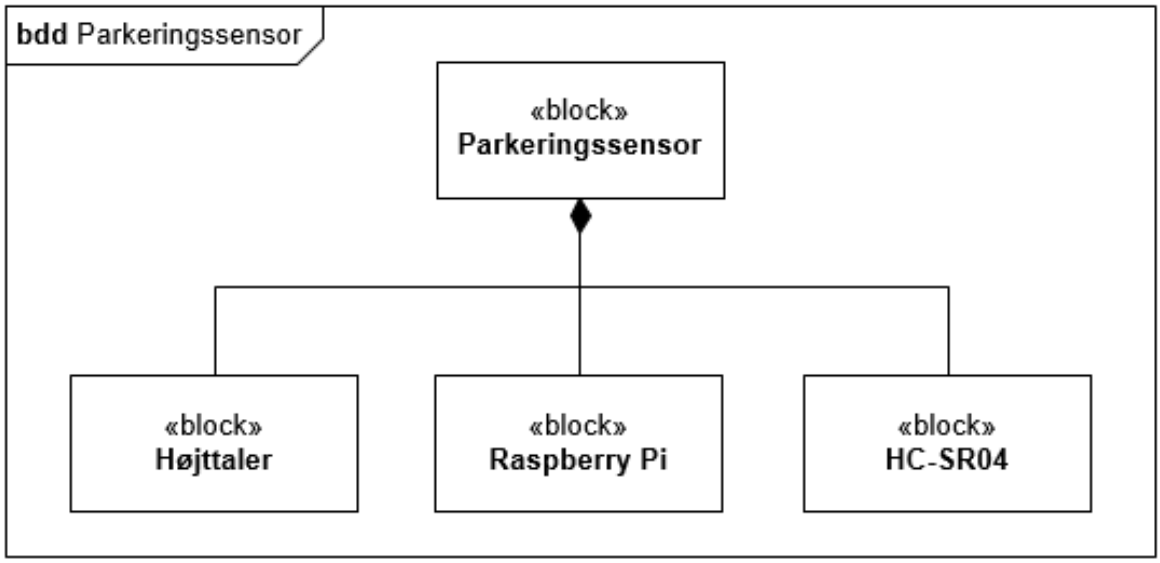
Start timer Indgangen til denne state sker, når der ikke modtages et UART interrupt. Ved indgangen til denne state bliver timeren startet

Send UART kommando Der ankommes til denne state, når timeren har talt 1 sekund. Ved indgangen til denne state resettes og stoppes der for timeren, og dataReceived arrayet for den modtagne UART besked bliver resat. Ved udgangen af denne state sendes der en kommando til Laseren, som laver en ny måling, og det hele starter forfra.

10.2 Parkeringssensor

10.2.1 Hardwareblokke

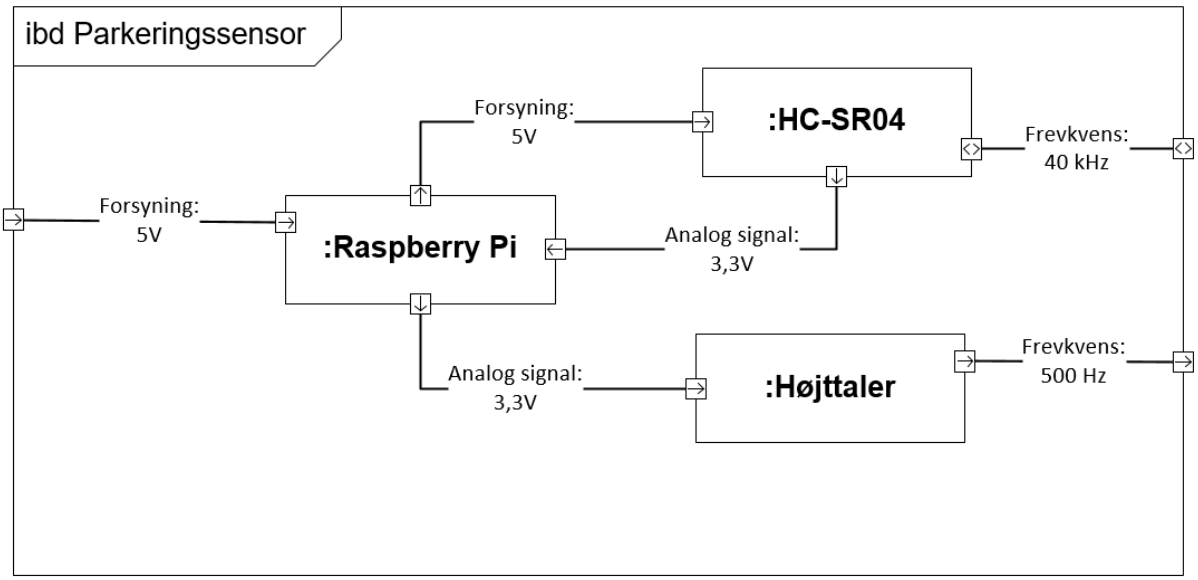
Parkeringssensoren består af følgende hardwareblokke: Raspberry Pi, højtaler og HC-SR04 se figur 10.



Figur 10: Bdd for parkeringssensor.

10.2.2 Interne hardwareforbindelser

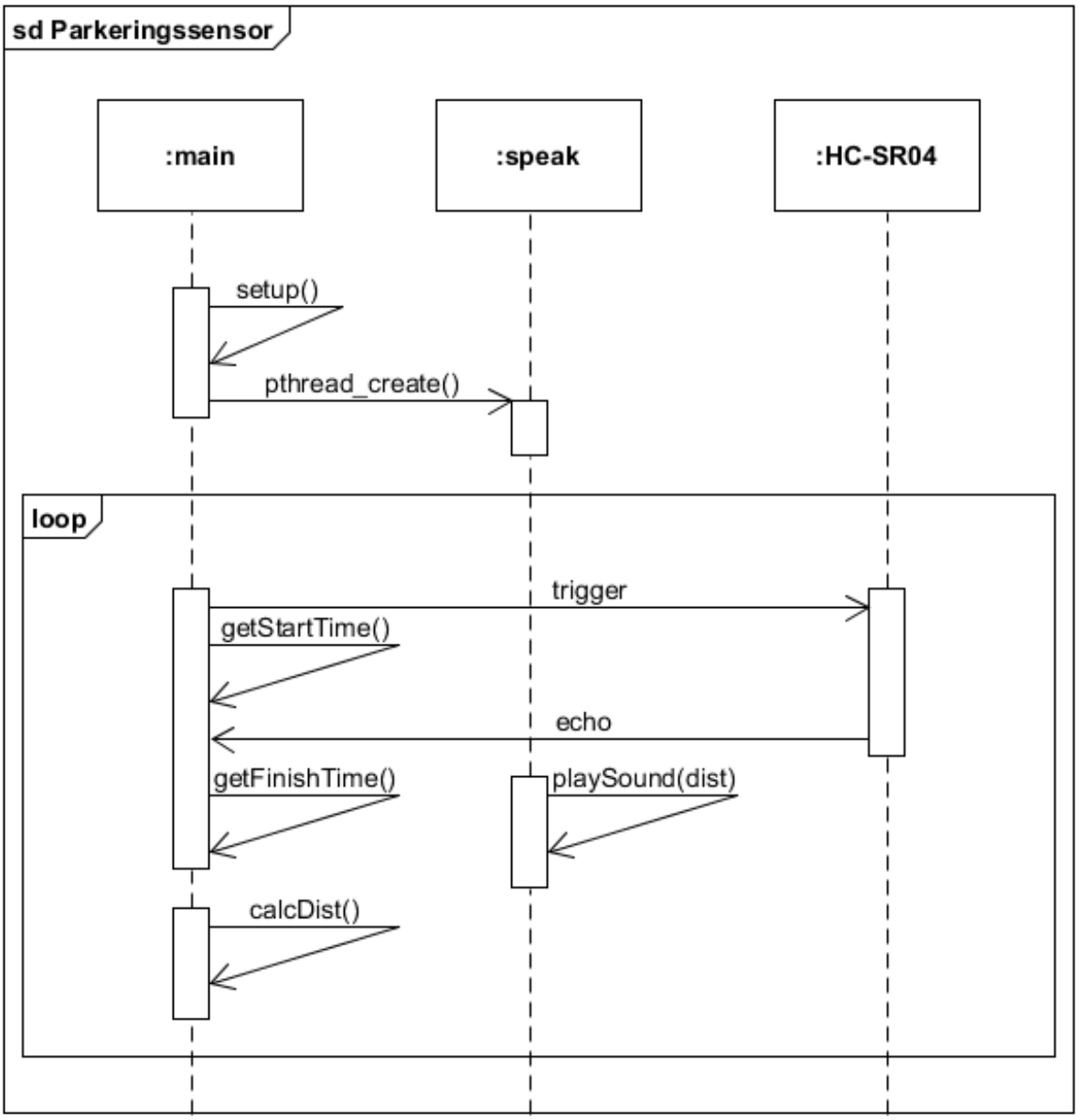
Parkeringssensoren består af følgende interne hardwareblokke: Raspberry Pi, højtaler og HC-SR04. Se figur 11. På figuren ses det, at der bliver sendt en 40 kHz frevkvens frem og tilbage til HC-SR04, det er en højfrevkvent lyd der bliver brugt til at måle afstanden fra modulet til et objekt med. Frevkvensen fra højttaleren på 500 Hz, er den tone der bliver afspillet ud af højtaleren, til advarsel for Bruger. Derudover ses både HC-SR04 og højttalerens spænding fra Raspberry Pi'en. De 3,3V der kommer fra HC-SR04 til Raspberry Pi'en er Echo fra HC-SR04, signalet som bliver brugt til at måle længden.



Figur 11: Ibd for parkeringssensor.

10.2.3 Afvikling af kode på Raspberry Pi

Sekvensdiagrammet for parkeringssensoren indeholder en main, speak og HC-SR04. Main sender først et signal til sonar som begynder at måle afstand, hvorefter sonar sender dataen tilbage. Dette foregår i en loop som først stopper når main bliver slukket. Når main får en afstand der er indenfor 1,5 m sendes en frekvens til højtaler i et interval, der bestemmes af afstanden til et objekt. Frekvensen bliver afspillet i et loop, så længe afstanden er i mellem 1,5 m og 1 m, vil intervallet være 1 sekund. Når sonaren måler en afstand der er mellem 1 m og 0,5 m bliver højttalernes loop på 0,5 s; under 0,5 m og 0,02 m er det 10 ms. Sekvensdiagrammet kan ses på figur 12.

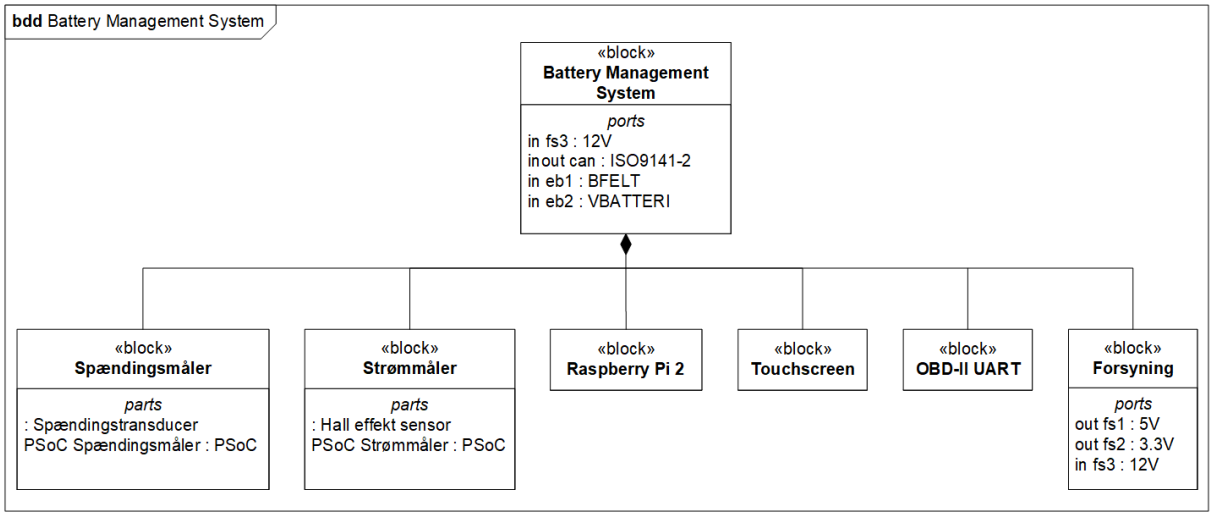


Figur 12: Sekvensdiagram for afvikling af software på Raspberry Pi.

10.3 Battery Management System

10.3.1 Hardwareblokke

Battery Management System består af følgende hardwareblokke: Spændingsmåler, Strømmåler, Raspberry Pi 2, Touchscreen, OBD-II UART og Forsyning. Bdd for Battery Management System ses på figur 13.

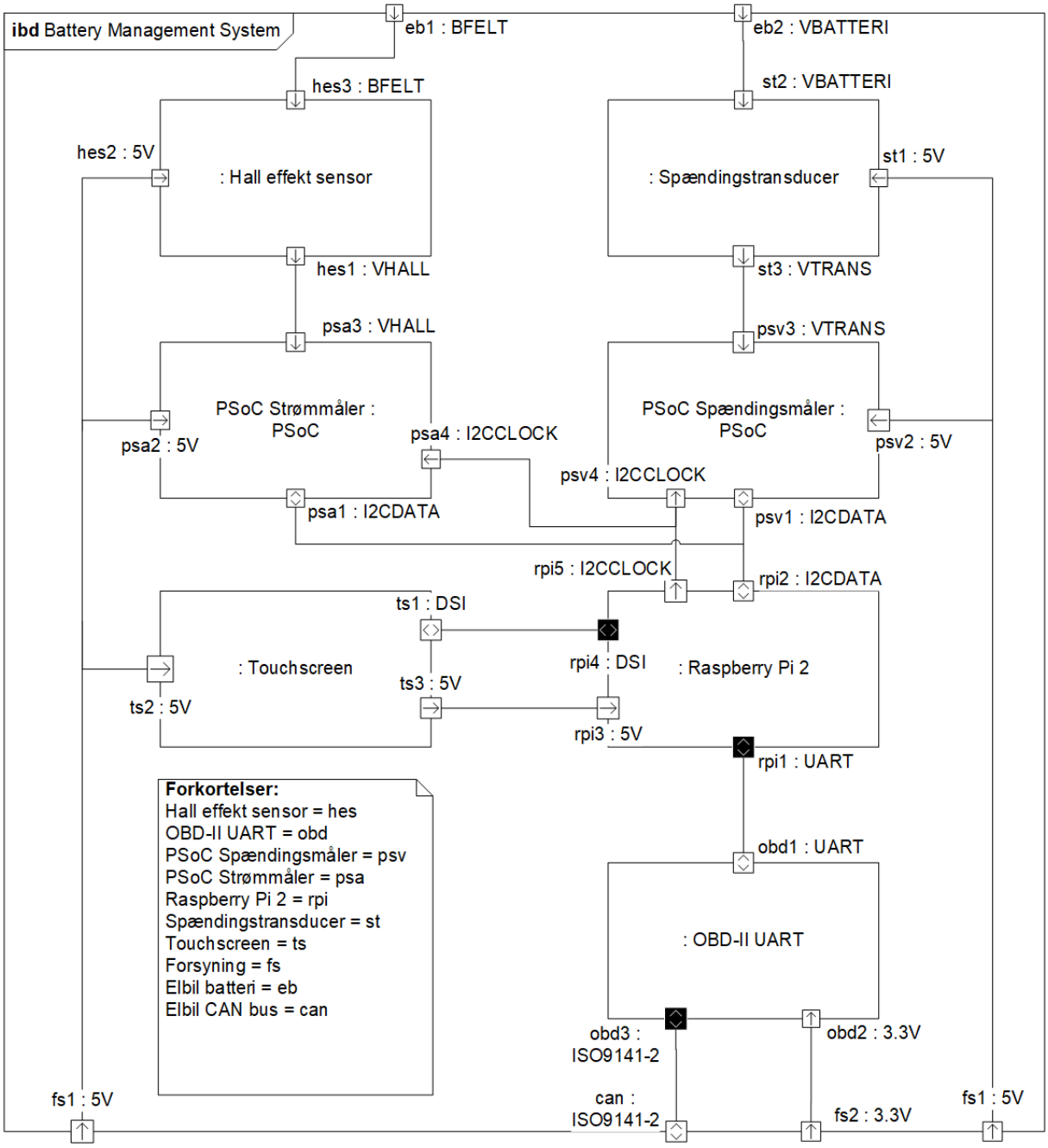


Figur 13: Bdd for Battery Management System.

Spændingsmåler er en hardwareenhed til måling af elbilens batterispænding. Spændingsmåler består af hardwareblokkene Spændingstransducer og PSoC. Spændingstransducer nedskalerer elbilens batterispænding, således at den kan måles med PSoC'ens ADC [9].
Strømmåler er en hardwareenhed til måling af strømtrækket fra elbilens batteri. Strømmåler består af hardwareblokkene Hall effekt sensor og PSoC. Hall effekt sensor måler strømtrækket fra elbilens batteri. PSoC måler udgangsspændingen fra Hall effekt sensor med en ADC.
OBD-II UART er en hardwareenhed, der muliggør kommunikation med elbilens On-board diagnostics system (OBD) [23]. OBD-II UART muliggør at information om elbilens hastighed kan hentes fra dens indbyggede microcontroller.
Touchscreen er en hardwareenhed til visning af systemets grafiske brugergrænseflade. Touchscreen er en 7“ Official Raspberry Pi berøringsfølsom skærm.
Raspberry Pi 2 er en hardwareenhed til integration og behandling af måledata fra Spændingsmåler, Strømmåler og OBD-II UART. Systemets grafiske brugergrænseflade til visning af data for brugeren kører også på Raspberry Pi 2 og vises på Touchscreen.
Forsyning er betegnelsen for systemets effektforsyning. Forsyning omsætter de ca. 12V fra elbilens cigaretstik til hhv. 5V og 3.3V. Dette skyldes, at nogle af systemets enheder kører på 5V mens andre kører 3.3V.

10.3.2 Interne hardwareforbindelser

En oversigt over de interne forbindelser mellem de forskellige hardwareblokke i Battery Management System kan ses på figur 14. På figuren ses også en oversigt over de forskellige forkortelser som er anvendt for portnavne i diagrammet. Som det fremgår er Battery Management System forbundet til en række enheder, som er eksterne til systemet: Hall effekt sensor og spændingstransducer er begge forbundet til elbilens batteri for at muliggøre måling af hhv. strøm og spænding. OBD-II UART er forbundet til elbilens OBD-stik for at muliggøre aflæsning af elbilens hastighed. Til intern kommunikation mellem systemets microcontrollere anvenes lavniveau-protokollerne UART og I2C. Det er værd at bemærke, at Raspberry Pi 2 agerer master over for de to slaveenheder PSoC Strømmåler og PSoC Spændingsmåler ved adressering på den samme I2C linje.



Figur 14: Ibd for Battery Management System.

Tabel 1: Blokbeskrivelse for Battery Management System

Blok-navn	**Funktionsbeskrivelse**	**Signal**	**Tolerance**	**Kommentar**
: Hall effekt sensor	Måler strømtrækket fra elbilens batteri og omsætter det til en spænding.	hes1 : VHALL	0 - 5 V	Udgangsspændingen fra Hall effekt sensoren.
		hes2 : 5V	4.75V - 5.25V	Forsyningsspænding.
		hes3 : BFELT	-	Størrelsen af B-feltet omkring ledningen hvorigennem strømmen fra elbilens batteri trækkes.
: OBD-II UART	Tillader kommunikation med elbilens CAN bus via et UART interface.	obd1 : UART	0V - 3.5V	Seriel dataforbindelse.
		obd2 : 3.3V	3.05V - 3.55V	Forsyningsspænding.
		obd3 : ISO9141-2	0V - 15V	Seriel dataforbindelse.
PSoC Spændingsmåler : PSoC	Sender den målte batterispænding videre til Rpi2.	psv1 : I2CDATA	0V - 3.5V	Seriel dataforbindelse.
		psv2 : 5V	4.75V - 5.25V	Forsyningsspænding.
		psv3 : VTRANS	0V - 5.25V	Udgangsspænding fra Spændingstransducer.
		psv4 : I2CCLOCK	0V - 3.5V	Clock til I2C forbindelse.
PSoC Strømmåler : PSoC	Sender det målte strømtræk videre til Rpi2.	psa1 : I2CDATA	0V - 3.5V	Seriel dataforbindelse.
		psa2 : 5V	4.75V - 5.25V	Forsyningsspænding.
		psa3 : VHALL	0V - 5.25V	Udgangsspænding fra Hall effekt sensor.
		psa4 : I2CCLOCK	0V - 3.5V	Clock til I2C forbindelse.
: Raspberry Pi 2	Behandler måledata og viser dem via Touchscreen.	rpi1 : UART	0V - 3.5V	Seriel dataforbindelse.
		rpi2 : I2CDATA	0V - 3.5V	Dataforbindelse til I2C kommunikation.
		rpi3 : 5V	4.75V - 5.25V	Forsyningsspænding.
		rpi4 : DSI	0V - 3.5V	Seriel dataforbindelse.
		rpi5 : I2CCLOCK	0V - 3.5V	Clock til seriel I2C forbindelse.
: Spændingstransducer	Nedskalerer elbilens batterispænding.	st1 : 5V	4.75V - 5.25V	Forsyningsspænding.
		st2 : VBATTERI	0V - 400V	Elbilens batterispænding.
		st3 : VTRANS	0V - 5.25V	Elbilens batterispænding nedskaleret.
: Touchscreen	På denne enhed vises systemets GUI.	ts1 : DSI	0V - 3.5V	Seriel dataforbindelse.
		ts2 : 5V	4.75V - 5.25V	Forsyningsspænding.
		ts3 : 5V	4.75V - 5.25V	Forsyningsspænding til Rpi2.
: Forsyning	Systemets effektforsyning.	fs1 : 5V	4.75V - 5.25V	Forsyningsspænding.
		fs2 : 3.3V	3.05V - 3.55V	Forsyningsspænding.
		fs3 : 12V	12V - 15V	Udgangsspænding fra elbilens 12V batteri.
: Eksterne forbindelser	Grænseflader mellem systemet og elbilen.	eb1 : BFELT	-	Størrelsen af B-feltet omkring ledningen hvorigennem strømmen fra elbilens batteri trækkes.
		eb2 : VBATTERI	0V - 400V	Elbilens batterispænding.
		can : ISO9141-2	0V - 14V	Seriel dataforbindelse.

10.3.3 Blokbeskrivelse

En blokbeskrivelse med beskrivelser og tolerancer for alle relevante signaler i Battery Management System kan ses på Tabel 1. Hvert signal er beskrevet ift. toleranceniveauer og dets funktionalitet.

10.3.4 Yderligere beskrivelse af de vigtigste hardwaresignaler

10.3.4.1 BFELT

BFELT er den størrelse som Hall effekt sensor måler. Princippet for en Hall effekt sensor er, at den måler størrelsen af det magnetfelt, som en ladning i bevægelse giver ophav til. I dette tilfælde anvendes en Hall effekt sensor til måling af den strøm, der trækkes ud af elbilens batteri. Denne strøm er proportional med det resulterende magnetfelt og magnetfeltet omsættes af Hall effekt sensoren til en tilsvarende proportional spænding betegnet VHALL.

10.3.4.2 VBATTERI

VBATTERI er spændingen over elbilens batteri. Da denne spænding er omkring 360V ved fuldt opladt batteri, er det nødvendigt at nedskalere den, for at den kan måles med PSoC'ens ADC. Dette gøres vha. Spændingstransduceren, der nedskalerer spændingen til et niveau mellem 0 - 5V. Denne spænding betegnes VTRANS.

10.3.4.3 ISO9141-2

ISO9141-2 er en kommunikationsstandard, der anvendes til kommunikation med elbilens On-board diagnostics system. Standarden minder om RS232 og har en baudrate på $10,4$ kbps. De nærmere detaljer omkring denne standard er ikke interessante for dette projekt. Ved anvendelse af OBD-II UART kan en række forskellige data om bilen forespørges ved kommunikation via UART. For dette projekts vedkommende, er det udelukkende bilens hastighed, der er interessant.

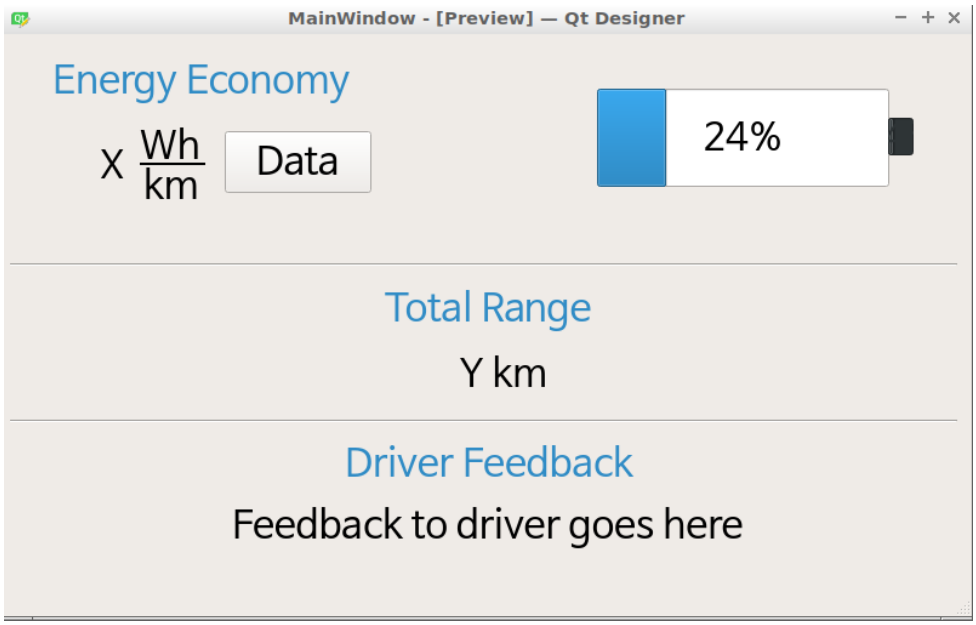
10.3.5 GUI

GUI er betegnelsen for Battery Management Systemets grafiske brugergrænseflade. Herigennem kan føreren af elbilen opnå en række informationer om elbilens status som tidligere beskrevet i afsnittet om systemets krav. Softwarearkitekturen for GUI er udviklet under hensyntagen til følgende:

1. Føreren af bilen skal via GUI'en opdateres løbende om følgende forhold:
    - Batteriets resterende kapacitet i procent.
    - Nuværende effektforbrug angivet som Wh/km.
    - Bilens resterende rækkevidde.
    - Feedback til forbedring af kørselsmønster.
2. Brugergrænsefladen skal være i stand til at modtage og behandle data fra tre andre enheder som beskrevet i afsnittet om hardwarearkitektur for Battery Management System.

For at opnå en bedre forståelse for hvordan GUI'en skal fungere i praksis, udvikles det grafiske design som det første. Dette gøres med udgangspunkt i den skitse af GUI'en, som kan ses på

figur 5. GUI'ens grafiske design udvikles vha. programmet QtDesigner [26]. QtDesigner tillader design af en brugergrænseflades udseende vha. simpel drag and drop af designelementer. GUI'ens endelige udseende fremgår af figur 15.



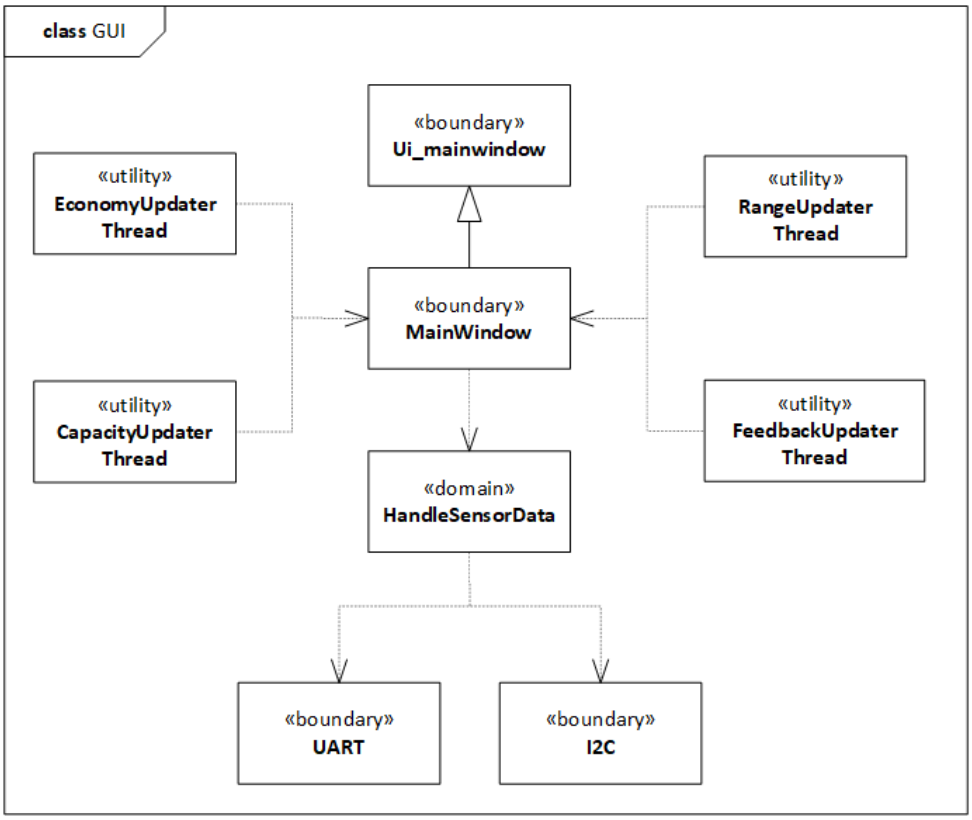
Figur 15: GUI'ens grafiske udseende udviklet ved brug af QtDesigner.

Det udviklede design viser, at GUI'en tilbyder brugeren fire typer information, som skal opdateres regelmæssigt:

1. Øverst til højre på figur 15 ses feltet til visning af energiforbruget (midlertidigt markeret med et "X").
2. Øverst til venstre ses en angivelse af batteriets tilbageværende kapacitet i procent (angivet stilistisk som et husholdningsbatteri).
3. Centralt ses bilens resterende rækkevidde angivet i km under overskriften "Total Range".
4. Nederst ses feltet til "Driver Feedback".

Da GUI'en regelmæssigt skal opdatere information for brugeren, er det nærliggende at anvende en tråd for hver af de fire elementer, der skal opdateres. Disse tråde implementeres som en klasse for hvert element, der skal opdateres. Disse benævnes hhv. EconomyUpdaterThread, CapacityUpdaterThread, RangeUpdaterThread og FeedbackUpdaterThread.
En trådet implementering tillader at hver af de fire elementer kan opdateres hver for sig med forskellig frekvens. F.eks. er det ikke nødvendigt at opdatere den resterende rækkevidde så ofte som energiforbruget. Klasser til implementering af grænseflader i form af I2C og UART oprettes ligeledes. En klasse til håndtering af den indkomne datastrøm fra disse interfaces er også påkrævet. Denne klasse benævnes HandleSensorData. I denne klasse skal de forskellige algoritmer til beregning af restkapacitet, rækkevidde osv. implementeres. Klassen MainWindow giver adgang til opdatering af GUI'ens forskellige grafiske elementer og kan til dels autogenereres vha. QtDesigner. Det er værd at bemærke, at MainWindow arver fra klassen Ui_mainwindow. Ui_mainwindow

indeholder autogenereret kode fra designet af GUI'en i QtDesigner. Det overordnede klassediagram for GUI'en kan ses på figur 16.

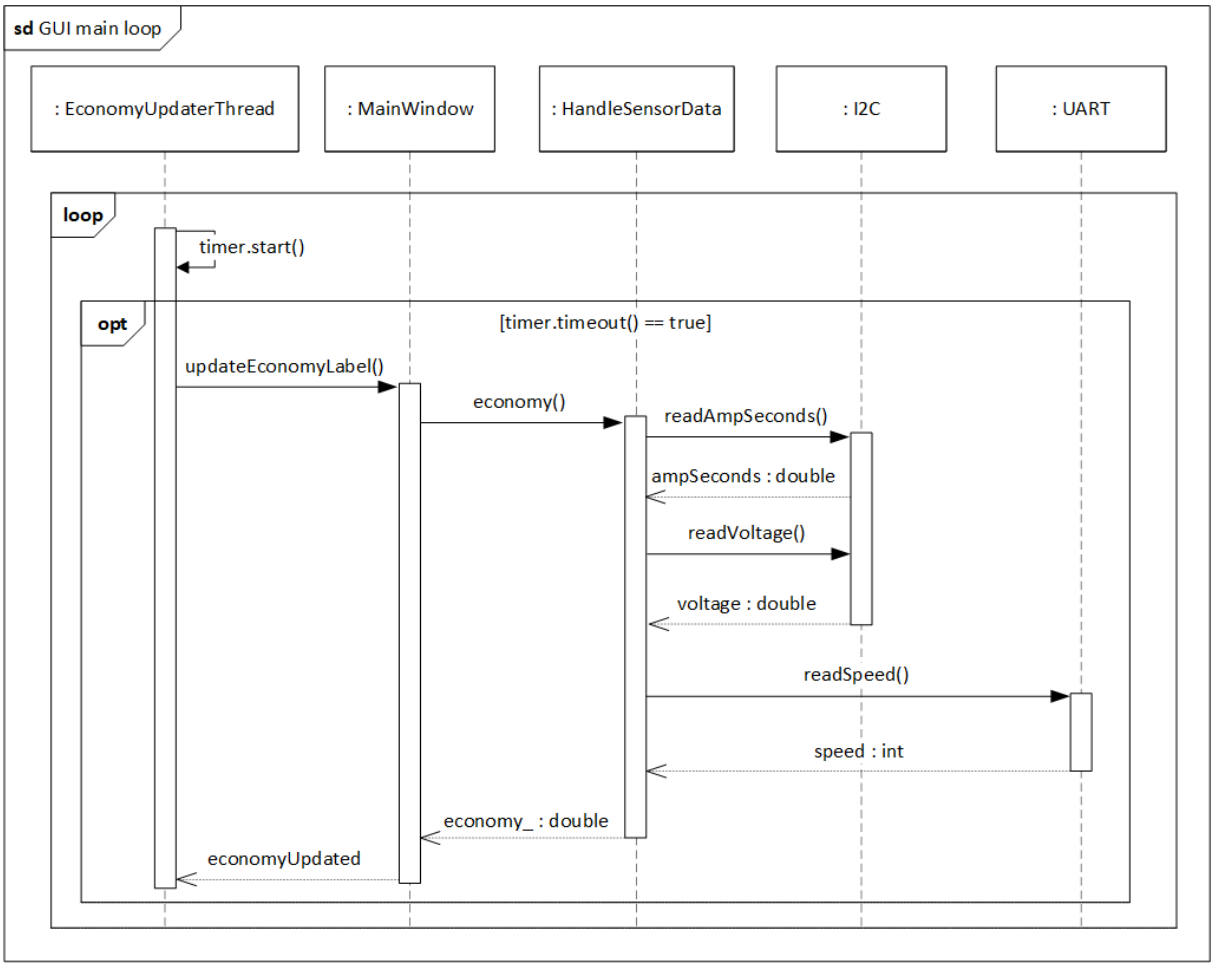


Figur 16: Klassediagram for GUI.

Bemærk at GUI'en har tre grænsefladeklasser: MainWindow udgør grænsefladen til brugeren. UART og I2C udgør grænseflader til andre enheder i Battery Management System. Den stiplede linje mellem klasserne i diagrammet angiver hvilke klasser, der anvender hinandens metoder.

10.3.5.1 Afvikling af kode på GUI

Koden på GUI afvikles i et uendeligt loop se figur 17. Ved opstart af programmet skal klasserne oprettes og de fire tråde startes. Dernæst opdateres hvert af de fire elementer (energiøkonomi, restkapacitet, rækkevidde og feedback) med forskelligt interval. Dette sker ved at applikationen først læser data via grænsefladeklasserne I2C og UART. Dernæst behandles de modtagne data i klassen HandleSensorData. Klassen MainWindow har funktioner til opdatering af den information, der vises på brugergrænsefladen. De fire trådklasser sørger for at denne opdatering sker med et bestemt interval. Sekvensdiagram for afvikling af kode på GUI'en ses på figur 17. For overskuelighedens skyld er kun én af trådene medtaget.



Figur 17: Sekvensdiagram for afvikling af main loop for GUI. Kun tråden til opdatering af energiøkonomi er medtaget.

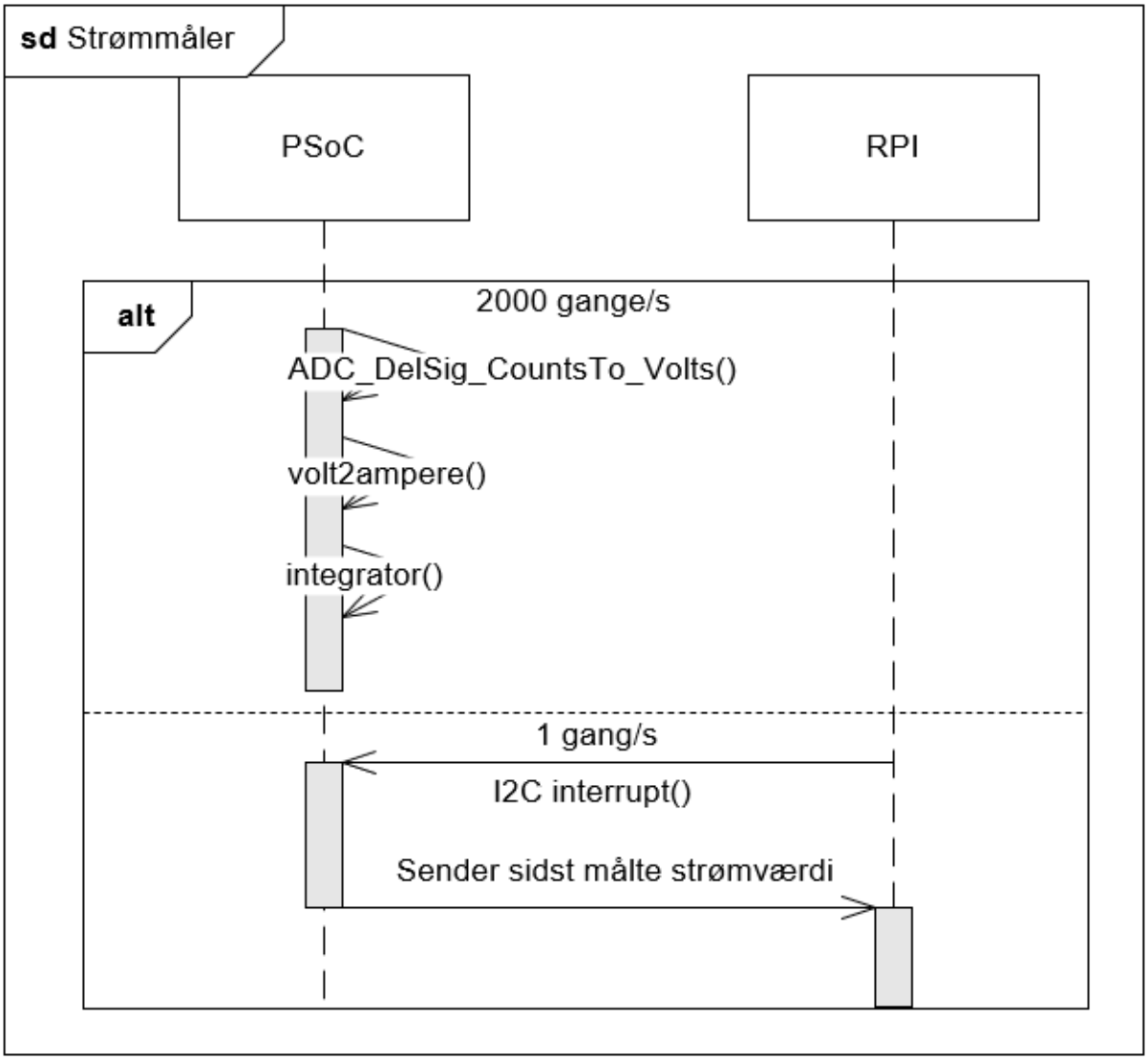
Afvikling af de andre tråde foregår på tilsvarende vis:

1. En timer startes med timer.start().
2. Når timeren har talt ned til nul dvs. timer.timeout() == 0 kaldes funktionen til opdatering af hhv. energiøkonomi, batterikapacitet, rækkevidde og feedback til brugeren.
3. Derefter startes timeren igen og loopet fortsætter.

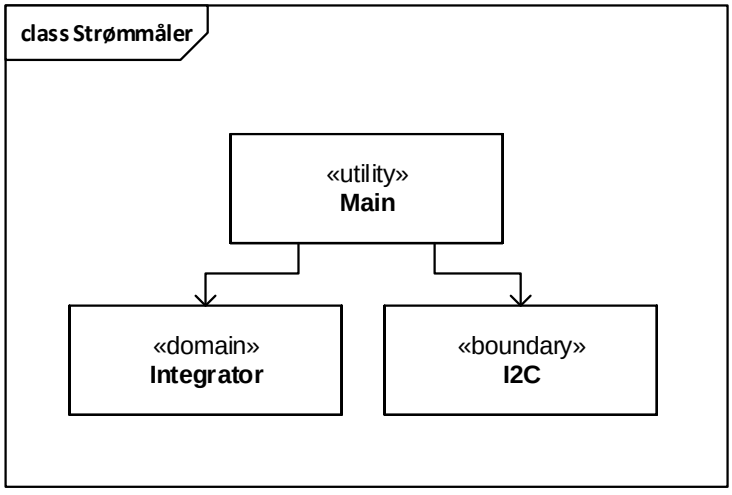
Fordelen ved denne fremgangsmåde er, at timerne kan tildeles forskellige værdier således at de forskelige informationer der vises på GUI'en, kan opdateres med forskellig frekvens.

10.3.6 Strømmåler

Strukturen af software for spændingsmåleren er blevet udtænkt gennem sekvens- og klassediagrammet vist på figur 19 og 18.



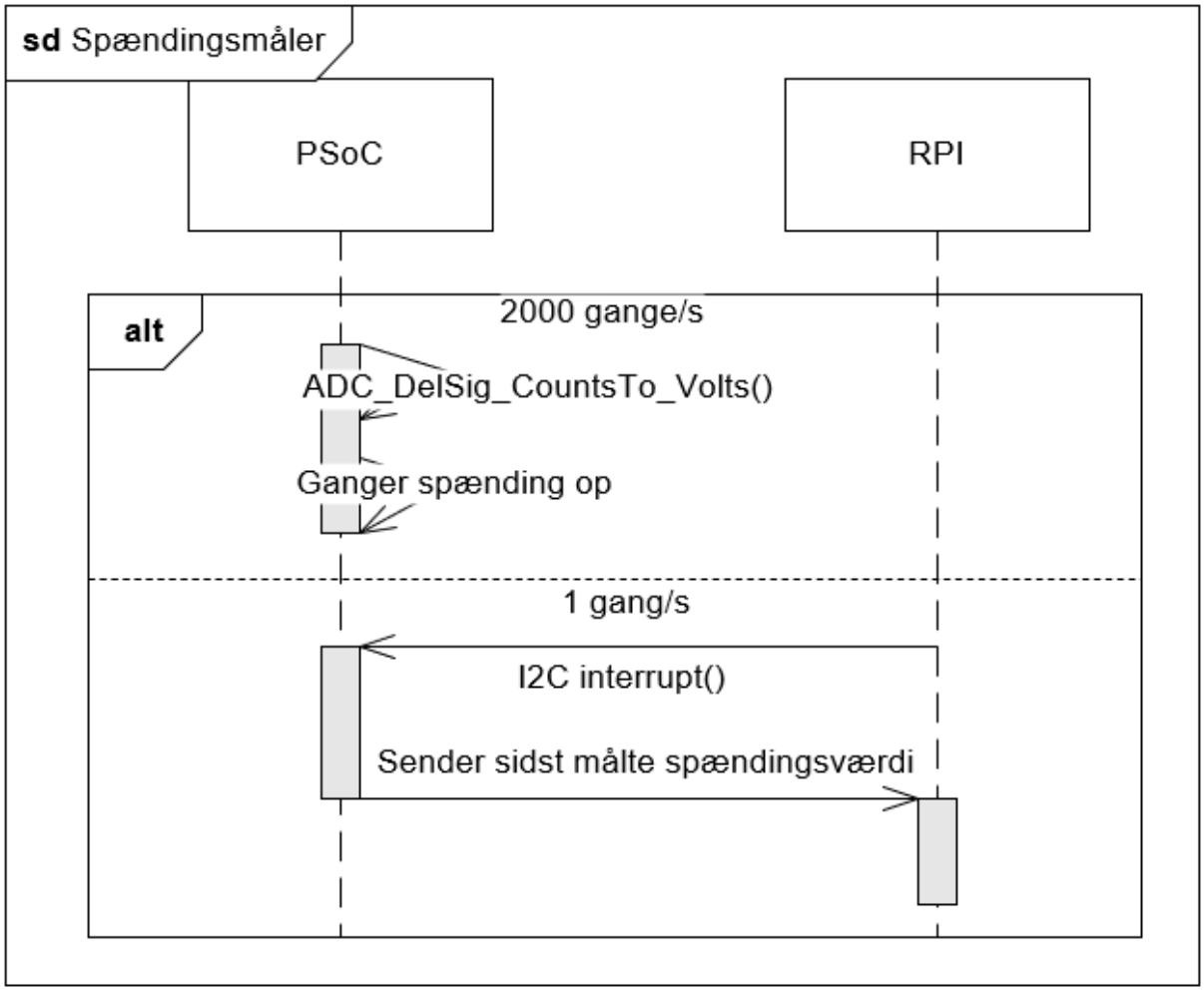
Figur 18: Sekvensdiagram for strømmåleren.



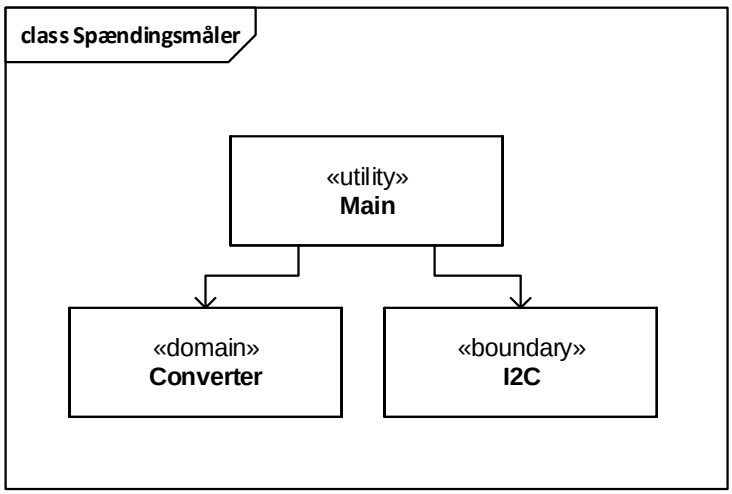
Figur 19: Klassediagram for strømmåleren.

10.3.7 Spændingsmåler

Eftersom elbilens batteri kan have en spænding på 360V, når det er fuldt opladt, kan spændingen ikke måles direkte af en PSoC, der kun kan tåle 5V. Derfor skal spændingen først justeres ned til et niveau, som en PSoC kan holde til. PSoC'en kan så derefter regne tilbage til den spænding, der faktisk er i batteriet. For at gøre justeringen lettere og simplere, er det blevet valgt, at 0-400V skal justeres ned til 0-5V. Dette gør vi via en spændingstransducer.
Herefter vil spændingen så blive målt af PSoC, der kan "forstærke"spændingen op igen. Dette gøres ved at PSoC'en indlæser spændingen som en talværdi og ganger dette med det antal gange, som transduceren dæmpede batteriets spænding med.
Strukturen af software for spændingsmåleren er blevet udtænkt gennem sekvens- og klassediagrammet vist på figur 21 og 20.



Figur 20: Sekvensdiagram for spændingsmåleren.



Figur 21: Klassediagram for spændingsmåleren.

10.4 Forsyning

DC strømforsyning til Hall Effekt Sensor, 2 x PSoC, RPi GPIO, RPi 7" touch screen og RPi 2 leveres af elbilens 12V cigarettænder, der befinder sig på instrumentbrættet. Cigarettænderen leverer som sagt 12V og 100W effektforsyning, og cigarettænderens udgangsstrøm findes ved Ohms lov:

$$I = \frac{P}{V} \tag{1}$$

$$I = \frac{100W}{12V} = 8.333A \tag{2}$$

Ved gennemgang af datablade for de forskellige dele opstår behovet for spændings- og strømregulering. Max. spænding for RPi GPIO er 3.3V [11] , RPi 7" touch screen 5V [37], RPi 2 5V [13], PSoC op til 5.5V [8], Hall Effekt Sensor 5V[14]. Vi skal derfor skabe et system, der regulerer spænding til de enkelte deles behov. Svaret er DC-DC spændingsregulering.

Ved en serieregulator løber strømmen så godt som direkte til belastningen – Den afsatte effekt er tilnærmelsesvist proportional med udgangsstrømmen [31]. Markedet er stort og prisen lav, så det kan simpelthen ikke betale sig at bygge en serieregulator selv. Værkstedet på Ingeniørhøjskolen har derfor et udvalg af spændingsregulatorer, der kan benyttes til opgaven. For at reducere antallet af spændingsregulatorer sammenlignes de enkelte deles strømforbrug og ved udregning af fælles strømforbrug, når man frem til flg.:

1. 1 x LD1117 spændingsregulator = 3.3V RPi GPIO
2. 1 x L7805CV spændingsregulator = 2 x PSoC + 1 x Hall Effekt Sensor
3. 1 x LM350 spændingsregulator = 1 x RPi 7" touch screen
4. 1 x L7805CV spændingsregulator = RPi 2

Af pladshensyn skal forsyningen designes på et enkelt board. Næste skridt er design og simulering i simuleringsprogrammet Multisim 14.1. I følgende afsnit skabes et kort overblik over

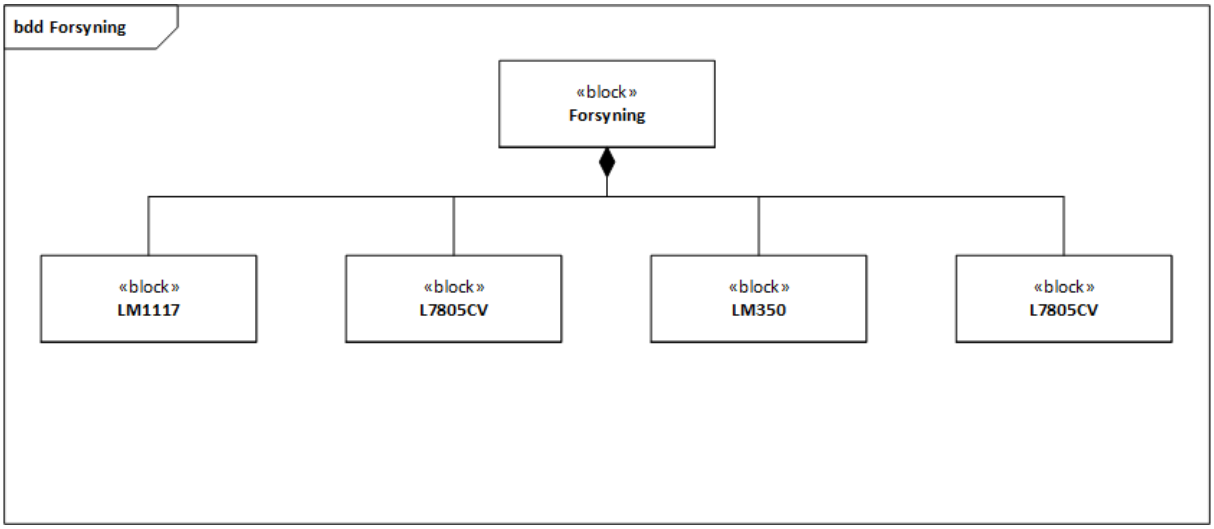
designprocedure, analyse, implementering og test af systemet. Uddybende detaljer henvises til bilag.

10.4.1 Hardwareblokke

Forsyningen består af følgende hardwareblokke:

1. LD1117 (3.3V, 800mA)
2. L7805CV (5V, 1.5A)
3. LM350 (5V, 3A)
4. L7805CV (5V, 1.5A).

Bdd for Forsyning ses på figur 22.

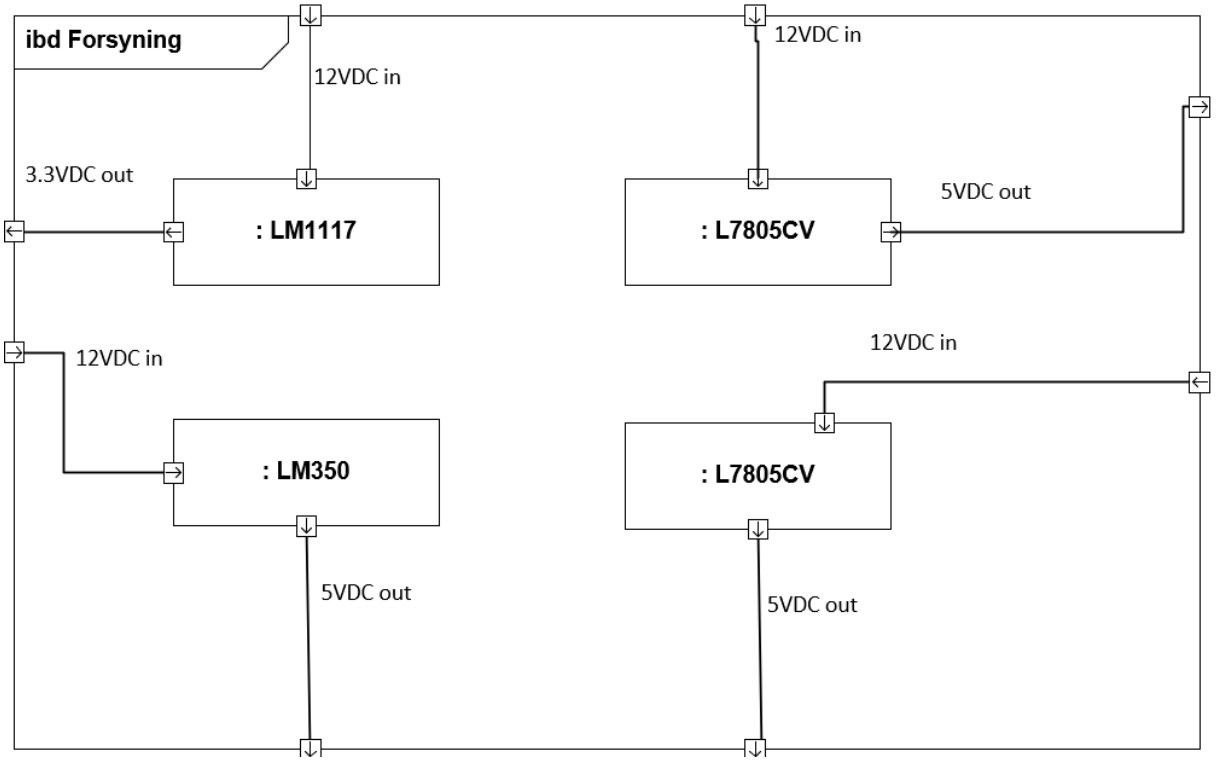


Figur 22: Bdd af Forsyning

LD1117 står for at forsyne RPi GPIO med 3.3V spænding. Denne regulator er valgt af hensyn til RPi databladets spændingskrav til GPIO.
L7805CV står for at forsyne 5V spænding til Hall Effekt Sensor samt 2 PSoC. Denne regulator er valgt af hensyn til de respektive datablades krav til spændingsforsyning.
LM350 står for at forsyne RPi 7" touchskærm med 5V. Denne regulator er valgt af hensyn til RPi 7" touchskærmens datablad, og dets krav til spændingsforsyning på 5V og strømkrav på min. 2.5A.
L7805CV står for at forsyne 5V spænding til RPi 2, der er monteret på touchskærmen. Denne regulator er valgt af hensyn til RPi 2s datablad og dets krav til spændingsforsyning på 5V. Grundet touchskærmens større strømforbrug ved opstart af systemet (ved at tænde elbilen mens hele systemet er tilsluttet), skete der et spændingsfald grundet det førnævnte store strømtræk. Touchskærmen meldte fejl om for lav spænding, og det var derfor nødvendigt at tilføje denne regulator.

10.4.2 Interne hardwareblokke

For at få et overblik over de interne forbindelser mellem de forskellige hardwareblokke i Forsyning, er der lavet et ibd. Dette ibd kan ses på nedenstående figur:



Figur 23: Ibd af Forsyning

RPi GPIO som forsynes af LD1117 står for kommunikation mellem Hall Effekt Sensor og RPi GPIO.
Hall Effekt Sensor, der forsynes af LM7805 sammen med 2 PSoCs, er systemets strømsensor, der måler elbilens batteri for strøm.
PSoC Spændingsmåler, som forsynes af LM7805, står for simulering af spændingsmåleren. Dette er grundet de høje spændinger, elbilens batteri opererer. Elbilen er gruppens rådighed hver torsdags, og PSoC1 anvendes derfor som simuleringsmiljø, når elbilen ikke er tilgængelig.
PSoC Strømmåler, der også forsynes af førnævnte LM7805, anvendes til afvikling af SmartLygter-programmet i et uendeligt loop.
RPi 7" touchskærm, der forsynes af LM350, der systemets GUI, der interagerer med brugeren. GUI'en viser relevant data for brugeren, der optimerer sin kørsel ud fra GUI'ens data.
RPi 2, der forsynes af LM7805, afvikler touchskærmens GUI-program.

11 Design og implementering

11.1 Smartlygter

Smartlygter er en prototype af automatiseret nærlys. Da det kun er en prototype, forventes den ikke implementeret på bilen, og test rækkeviden er kun en tiendedel af de specifikationer som der er på bilens nærlys. Det vil sige, at hvis der normalt vil skulle måles mellem 30m og 35m, bliver der ved prototypen kun målt mellem 3m og 3.5m.

11.1.1 Laser Distance Sensor

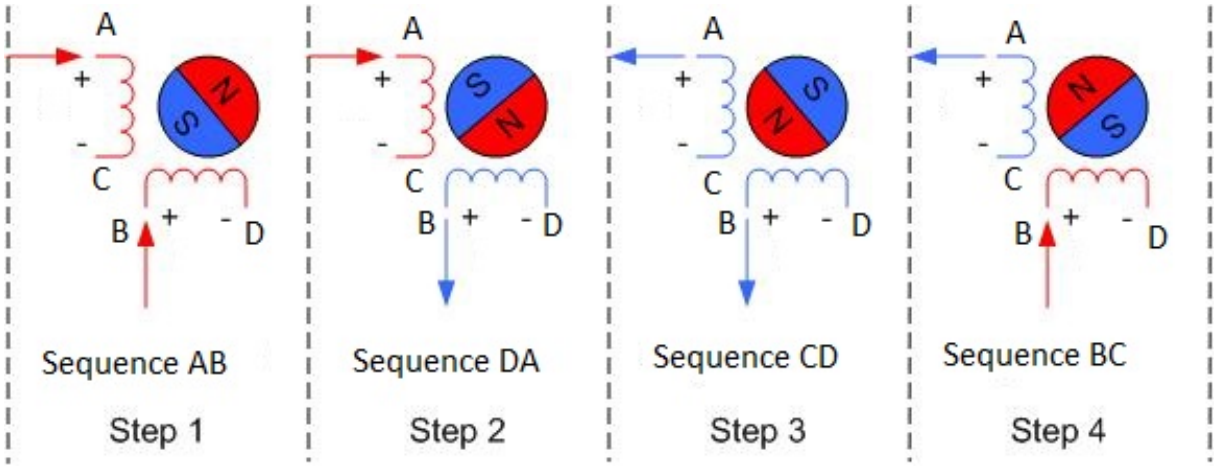
I designet af Smartlygter er der blevet brugt en Laser Distance Sensor 701A. Dette module benytter time-of-flight princippet til at måle afstand. Time-of-flight princippet går ud på at sensoren udsender en laserstråle ved at modtage en kommando. Derefter rammer denne laserstråle et fjernt eller nært objekt, som reflekterer laserstrålen tilbage til sensoren. Herefter bruger sensoren tiden det tog laserstrålen fra at blive udsendt til den vendte tilbage, samt lysets hastighed til at udregne afstanden. Udregningen af distance bliver hermed målt med følgende formel:

$$\text{Distance} = \frac{c \cdot t}{2}$$

Hvor c er lysets hastighed i km per sekund, t er tiden i sekunder og det bliver divideret med to, da laserstrålen både skal frem og tilbage.
Funktionaliteten af Laser Distance Sensor 701A i vores Smartlygter design skal symbolisere, at vi måler, hvor langt nærlyset i en bil lyser. Hvis dette nærlys lyser for langt, vil det sandsynligvis være fordi bilen er på vej over en bakke, og dermed skal nærlyset vinkles nedad.
Denne Laser Distance Sensor 701A har en række UART kommandoer installeret, som er benyttet til at få den rigtige funktionalitet. Disse kan findes i databladet [16].

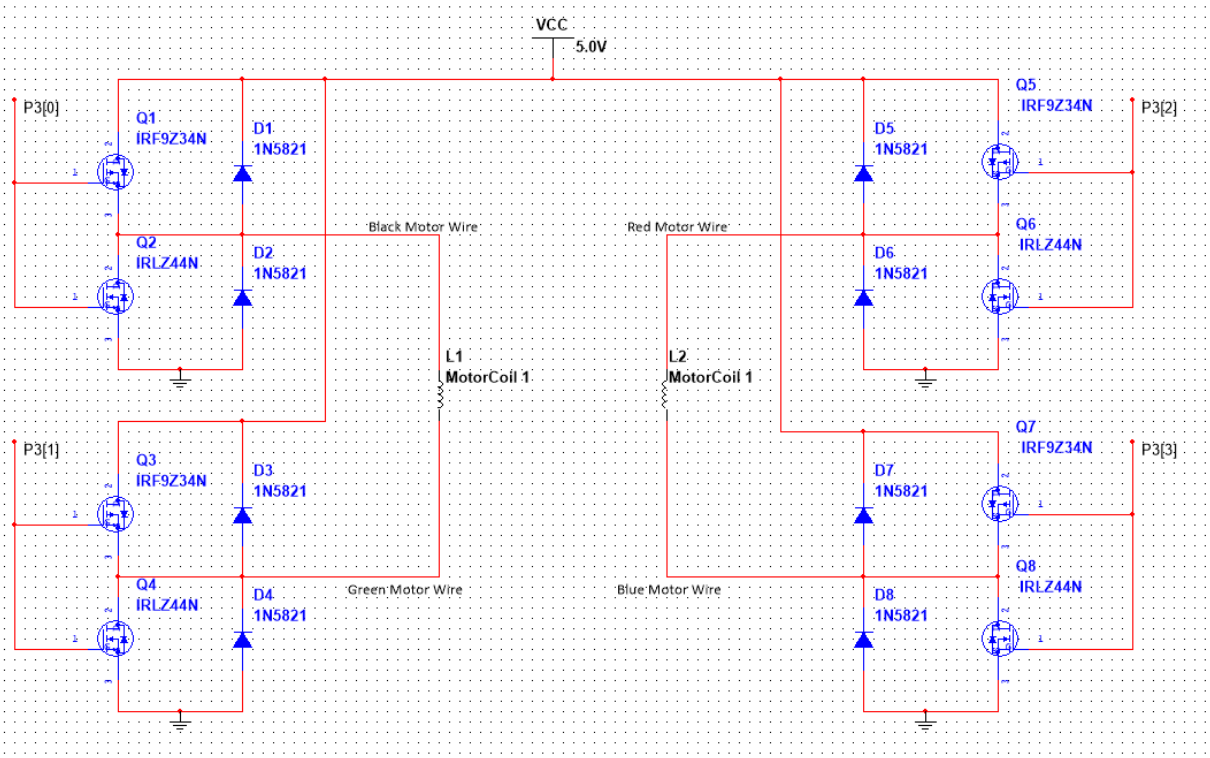
11.1.2 Step-motor

Til at vinkle nærlyset, er der brugt en step-motor. Den valgte step-motor er en fire-lednings bipolar step-motor med to faser. Den kan ved hjælp af de to faser skabe drejningsmoment og dermed rotere.



Figur 24: Bipolar Step-motor. Modificeret fra [3]

For at skabe et drejningsmoment skal der først sendes strøm igennem A og B. Dette gør, at de to spoler vil lade deres pol, så sydpolen fra rotoren bliver tiltrukket af de to poler fra spolerne. Herefter bliver det A og D der skal sendes strøm igennem, som gør at polariteten igennem BD spolen ændrer sig, og dermed tiltrækker nordpolen fra rotoren. Herefter bliver der sendt strøm igennem C og D. Dette gør nu, at de to spoler begge har ændret deres polaritet, og at nordpolen fra rotoren bliver tiltrukket af de to poler. Ved at køre de 4 steps igennem bliver der skabt et drejningsmoment, og motoren roterer.
Selve denne funktionalitet kan styres ved hjælp af digital styring. Her er det dog påkrævet, at der er tilsat et kredsløb, som kan styre hvilken vej strømmen skal gå igennem hver enkelt spole og dermed polariteten af denne. step-motor kredsløbet kan ses på figur 25.



Figur 25: Step-motor Kredsløb

Dette kredsløb består af mosfets, som skal styre hvilken vej strømmen løber igennem spolerne i motoren. Disse sidder i par, med henholdsvis en P-channel mosfet først og derefter en N-channel. Når der bliver sendt 5V fra PSoC'en for eksempel i P3[0], vil der blive lukket for P-channel mosfetten men åbnet for N-channel mosfetten. Så det vil sige, at hvis der for eksempel lukkes for P3[0], så der var 0V der og åbnes for P3[1], så der er 5V, så ville strømmen løbe igennem Q1 mosfetten, derefter igennem spolen, da Q2 mosfetten er lukket og til sidst ned gennem Q4 mosfetten og få jordforbindelse.
Ud over mosfets er der dioder i kredsløbet. Disse er det, man kalder flyback dioder, og de skal sørge for, at der ikke kommer høje induktive spikes fra motoren. Dette sørger for at spændingen ikke stiger så meget, at det kan ødelægge de forskellige mosfets.
For at dette kredsløb skal fungere, er der blevet brugt en PSoC til åbne og lukke for de forskellige mosfets, således at der dannes drejemoment i motoren. Ud fra kendskabet til motoren, er der blevet lavet en halv-step funktion. Halv-step sekvensen kan ses i Tabel 2.

Step	A. Sort	B. Rød	C. Grøn	D. Blå
1	1	0	0	0
1	1	1	0	0
1	0	1	0	0
1	0	1	1	0
1	0	0	1	0
1	0	0	1	1
1	0	0	0	1
1	1	0	0	1

Tabel 2: Step-motor sekvens tabel

På tabellen kan det ses, hvilke mosfet transistorer der åbnes og lukkes for ved hjælp af PSoC programmet. Dette vil dermed få motoren til at rotere. Et nul symboliserer, at der ikke løber strøm igennem, og et 1 tal symboliserer, at der løber strøm igennem. Udover dette er der implementeret et delay mellem hvert enkelt step, da dette er nødvendigt, for at motoren kan opnå den ønskede trækkraft. For at stoppe motoren bliver der slukket for alle N-channel mosfets, hvilket vil sige, at der bliver sendt 0V i portene.
Grunden til at der er valgt halv-step og ikke fuld-step, er at hvert enkelt step havde en vinkeldrejning på 1,8 grader per step, og man vil gerne kunne styre ned til 0,9 grader per step, hvilket kan opnås med halv-step. Denne funktionalitet er vigtig, da Smartlygterne starter med at være vinklet vandret, og ved at vinkle dem 0.9 grader nedad, passer det med at de falder med 1%, som er minimums specifikationen på bilens nærlys [33].

11.2 Parkeringssensor

11.2.1 Sonar

Lyd i sin essens er trykbølger. Når lyd anvendes, arbejder vi derfor med bølger. Bølger måles ved deres bølgelængde, som er det man kalder "én hel bølge" altså fra bølgetop til bølgetop. Når

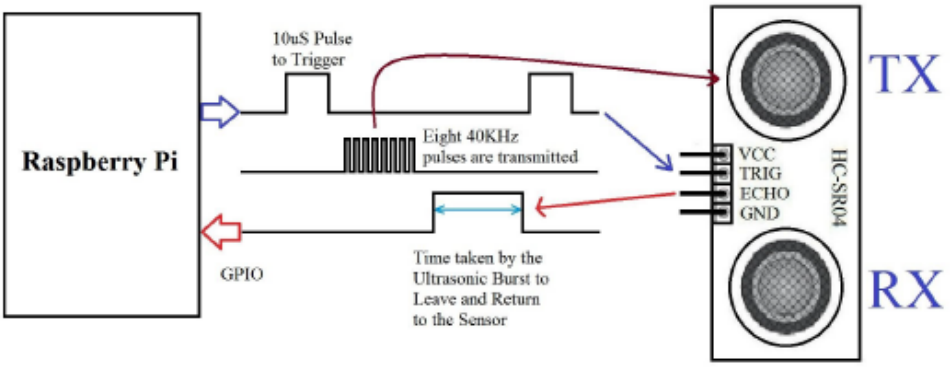
der kendes til bølgelængden, kan man udregne det frekvensspektrum, der arbejdes med. Dette gøres ved at tage lydens hastighed og dividere denne med bølgelængden. Vi arbejder uden for det menneskeligt hørbare område altså uden for 20 Hz og 20.000 Hz. Dette område omtales i daglig tale som ultralyd.
Sonarer er næsten udelukkende ultralydsbaserede i dag. Der er dog nogle anlæg, som anvender meget høje lyde samt meget dybe rungende lyde. Alt dette bestemmes ud fra hvilken overflade lyden skal returnere fra, samt hvilket miljø bølgerne skal rejse igennem. I vores tilfælde arbejder vi med luft. Det er almindeligt kendt at lydens hastighed i luft er ca. 340 meter/sekund. Måden vores sonar fungerer på, er ved at udsende et sådant højfrekvent signal, som reflekteres når denne møder en overflade. Sonaren opfanger så signalet, der kommer retur, ofte kaldet ekko, og tiden der er afsat måles så, og man kan ud fra dette så fastslå distancen til objektet, hvorfra lyden blev reflekteret. Hvis ikke man anvendte ultra lyd, ville der komme et retursignal hver gang en lyd blev opfanget, som var inden for det hørbare frekvensspektrum, og dette ville dermed forvirre diverse systemer, man udvikler med sonar.

11.2.2 HC-SR04

I designet af "Parkeringssensor" er der anvendt modulet HC-SR04. For at kunne advare forbrugeren imod uheld under parkering af bilen er modulet HC-SR04 anvendt til implementering af parkeringssensor. Denne er dernæst koblet til en Raspberry Pi 3b, som agerer CPU for modulet. Yderligere er der koblet en højtaler på, som ellers normalt ville være bilens højtaler(e) man anvendte. Tilsammen danner disse elementer samt den software, som booter når Pi'en opstartes, et alarmeringssystem, der kontinuerligt måler om bilen er inden for rækkevidde af et trusselsudgørende objekt.
HC-SR04 sonarmodulet har 4 pins, disse anvendes alle:

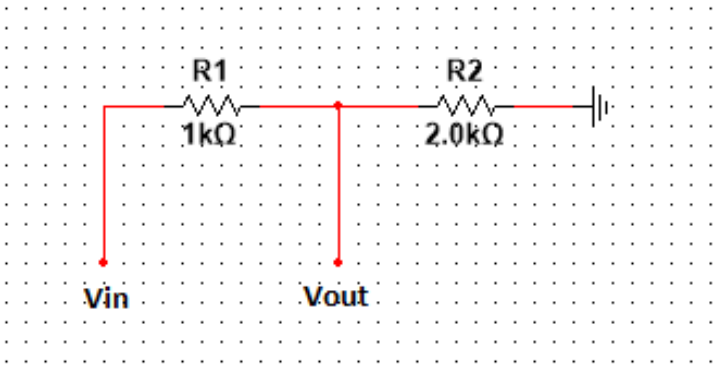
- Vcc (5V DC)
- Trig (Trigger pulse input)
- Echo (echo pulse input)
- GND (ground)

Måden HC-SR04 fungerer på er forholdsvis simpel. Modulet tilsluttes VCC og ground på vores Pi. Dernæst tilsluttes Trig og Echo pins som defineres ud fra Raspberry Pi's pin mapper. Der er brugt pin 16 og pin 18 - disse pins kunne have været hvilken som helst GPIO pin, så længe det matcher i koden. Trig bruges til at sende en 10 μs puls fra Raspberry Pi til HC-SR04, som får modulet til at sende otte 40KHz ultralydssignaler afsted. Disse modtages så igen af receiveren og via ECHO sendes info tilbage til Raspberry Pi.



Figur 26: HC-SR04, Sonar forløb. Tilpasset fra [4].

HC-SR04 modulet trækker 5V DC fra Vcc på Pi, men da Pi'ens GPIO Pins kun må tage 3,3V, skal disse beskyttes ved at sænke 5V signalet. Til dette anvendes en lille spændingsdeler. Dette semester indebar dog også viden om hvordan Zener dioder kan regulere spænding i et kredsløb, men det ville hverken være en bedre, eller simplere løsning. Så der blev anvendt to modstande, R1 og R2 sat i serie, som danner spændingsdeleren. Det er Echo signalet, som Pi'en skal beskyttes imod, da den ellers ville modtage 5V fra HC-SR04 modulet.



Figur 27: Spændingsdeler

Kredsløbet for spændingsdeleren er bestemt ud fra en 1kΩ modstand. Værdien udregnes bare for den anden modstand, eftersom vi kender indgangsspændingen samt den udgangsspænding, som vil opnås.

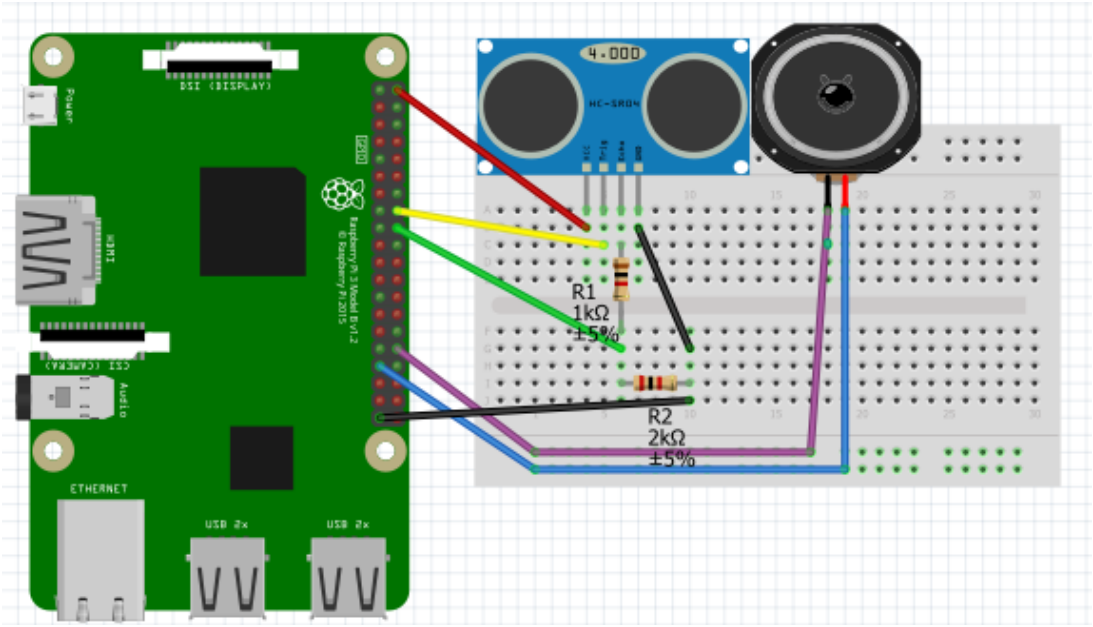
$$V_{in} := 5\ V \qquad V_{out} := 3.3\ V \qquad R1 := 1000\ \Omega$$

$$R2 := \frac{V_{out}}{V_{in}} = \frac{R2}{R1 + R2} \xrightarrow{solve, R2} 1941.1764705882352941 \cdot \Omega$$

$$R1 = 1000\ \Omega \qquad R2 := 1941\ \Omega$$

Figur 28: HC-SR04 Spændingsdeler

Efter at have udregnet på spændingsdeleren er det tid til at opbygge kredsløbet, og forbinde det med Pi'en. Til illustrering af opbygningen, er et opensource program ved navn Fritzing, blevet anvendt. [5].

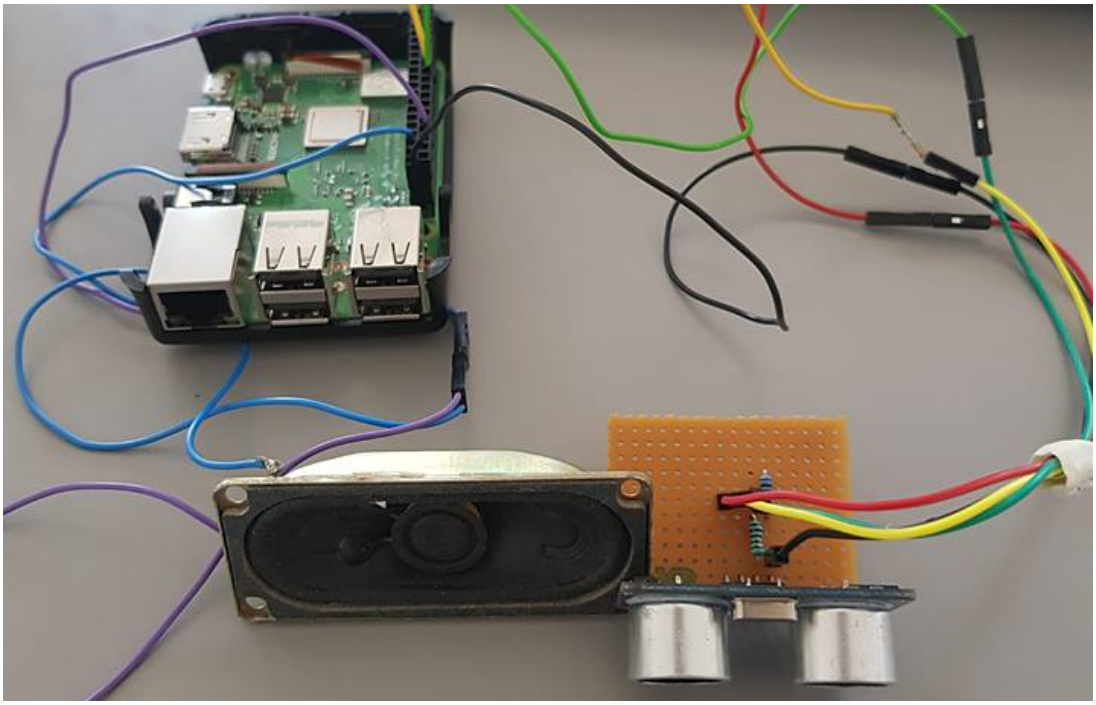


Figur 29: Overblik af sonar kredsløb

Som det fremgår af ovenstående figur, har HC-SR04 modulet 4 pins som tidligere forklaret: Vcc, Trig, Echo og Ground. Disse skal forbindes til Pi på den rigtige måde. Nogle pins er intetsigende, så længe de passer i den tilhørende software hvorimod andre har begrænsede muligheder.

"Pin Map" er således opbygget:

- Vcc er forbundet til Pin 2 (rød ledning, øverst til højre)
- Trig er forbundet til Pin 16 (gul ledning, GPIO 23)
- Echo er forbundet til Pin 18 (grøn ledning, GPIO 24)
- GND er forbundet til Pin 39 (sort ledning, nederst til venstre)



Figur 30: Selve realiseringen af vores Parkeringssensor ser således ud.

11.2.3 Parkeringssensor software

Main() for parkeringssensoren ses i Listing 1 på side 46. Her bliver hardwaren først opsat gennem funktionen setup(). Efter opsætningen oprettes en ny tråd som bliver brugt til speak(). I main's while loop måles afstanden, hvorefter den bliver omregnet til cm og derefter printet.

Listing 1: Udsnit af centimeter() hvor afstanden bestemmes

```
1  int main() {
2
3      setup();
4      pthread_t thr1;
5      pthread_create(&thr1, NULL, speak, NULL);
6
7      while(1) {
8          digitalWrite(TRIG, LOW);
9                      //digitalWrite saetter output pin til at vaere enten high/low
10         dist = centimeter();
11         printf("Afstand til objekt: %dcm\n", dist);
12         delay(500);
13     }
14     return 0;
15 }
```

Funktionen for centimeter() som ses i Listing 2 på side 47 er opbygget ved at der først bliver sendt et puls/TRIG af Sonaren, derefter kommer der et delay på 20ms og pulsen bliver sat low igen. Efter ventes der på at returpulsen/ECHO modtages. Når ECHO er modtaget læser den det modtagede data, hvorefter dataen bliver oversat til centimeter/distance som bliver sendt retur.

Listing 2: Udsnit af centimeter() hvor afstanden bestemmes

```
int centimeter() {              //funktion til afstandsbemaaling

    //Sender TRIG puls
    digitalWrite(TRIG, HIGH);           //Setting TRIG HIGH
    delayMicroseconds(20);              //Added delay to avoid reading overload
    digitalWrite(TRIG, LOW);            //Setting TRIG LOW

    //Venter paa ECHO start
    while(digitalRead(ECHO) == LOW);    //while loop checking if ECHO is LOW

    //Venter paa ECHO slut
    long startTime = micros();      //
    while(digitalRead(ECHO) == HIGH);   //While loop checking if ECHO is HIGH
    long travelTime = micros() - startTime; //

    //Regner afstand i cm
    int distance = travelTime / 58;

    return distance;
}
```

11.3 Battery Management System

11.3.1 GUI

11.3.1.1 Valg af programmeringssprog

Som nævnt i analyseafsnittet er Qt blevet valgt til implementering af GUI'en. Da GUI'en ønskes udviklet i programmeringssproget Python, anvendes PyQt sammen med python3 til den egentlige programmering. QtDesigner blev i arkitekturfasen anvendt til at designe GUI'ens grafiske udseende. Med PyQt kan den autogenererede C++ kode fra dette design omdannes til Python kode. Da Python er et højniveau sprog, er det særlig velegnet til relativt hurtigt at implementere en fungerende brugergrænseflade.

11.3.1.2 Implementering af grænseflader

For at GUI'en kan fungere, skal den være i stand til at kommunikere med de tre andre CPU'er, der er en del af Battery Management System. Dette varetages af de to grænsefladeklasser I2C og UART. I2C klassen læser 16 bytes sv.t. to doubles fra strømmåleren via et kald til funktionen readAmpSeconds(). Funktionen readVoltage() læser tilsvarende 8 bytes fra spændingsmåleren. I2C anvender en i2c protokol med en hastighed på 100 kbps. Via UART klassen kommunikerer GUI'en med SparkFun OBD-II UART. Ved et kald til funktionen readSpeed() returneres elbilens nuværende hastighed til GUI'en.

11.3.1.3 Implementering af algoritme til beregning af restkapacitet

Den resterende kapacitet i elbilens batteri beregnes som beskrevet i analyseafsnittet. I klassen HandleSensorData implementeres beregning af restkapaciteten i funktionen economy(). Dette

skyldes at bilens energiøkonomi er den størrelse, som skal opdateres oftest for brugeren. Derfor er HandleSensorData.economy() den funktion, som benytter grænsefladeklasserne til at læse værdier for spænding, kapacitet i $A \cdot s$ samt bilens hastighed. I Listing 3 på side 48 ses et udsnit af koden for funktionen.

Listing 3: Udsnit af HandleSensorData.economy() hvor resterende kapacitet beregnes.

```
1 self.ampHours = self.i2c_interface.readAmpSeconds()
2 self.voltage = self.i2c_interface.readVoltage()
3 self.speed = self.uart_interface.readSpeed()
4
5 if not (self.ampHours or self.voltage) == "READ ERROR":
6     self.wattHours = (self.ampHours * self.sampleTime - self.ampHours_old) * self.
          voltage
7     if self.capacityLeft_ > self.wattHours:
8         self.capacityLeft_ -= self.wattHours
9     self.ampHours_old = self.ampHours * self.sampleTime
```

Som det fremgår læser funktionen først den forbrugte kapacitet, spænding og bilens hastighed. Hvis værdierne er aflæst korrekt omregnes den aflæste spænding og strøm til Watt-timer. Hvis der stadig er kapacitet tilbage på batteriet, trækkes det beregnede antal Watt-timer fra bilens totale kapacitet.

11.3.1.4 Implementering af algoritme til beregning af bilens rækkevidde

Bilens rækkevidde beregnes ud fra algoritmen, der er beskrevet i analyseafsnittet. Denne algoritme implementeres også til dels i funktionen HandleSensorData.economy(). Gennemsnittet af effekt og hastighed for det seneste minut beregnes som på Listing 4 på side 48.

Listing 4: Udsnit af HandleSensorData.economy() hvor gennemsnittet af effekt og hastighed beregnes.

```
1     self.powerAvg = self.alpha * (self.wattHours/self.sampleTime) + (1-self.alpha) *
          self.powerAvg
2
3 if not self.speed == "READ ERROR":
4     self.speedAvg = self.alpha * self.speed + (1-self.alpha) * self.speedAvg
```

Som det fremgår beregnes den gennemsnitlige effekt og hastighed med anvendelse af den værdi af α, der blev fundet i analysen. Den gennemsnitlige effekt og hastighed anvendes nu sammen med batteriets resterende kapacitet til at beregne bilens resterende rækkevidde. Dette gøres i funktionen HandleSensorData.estimateRange() se Listing 5 på side 48.

Listing 5: Bilens resterende rækkevidde estimeres i HandleSensorData.estimateRange().

```
1 def estimateRange(self):
2
3     self.range = self.speedAvg * self.capacityLeft_ / self.powerAvg
```

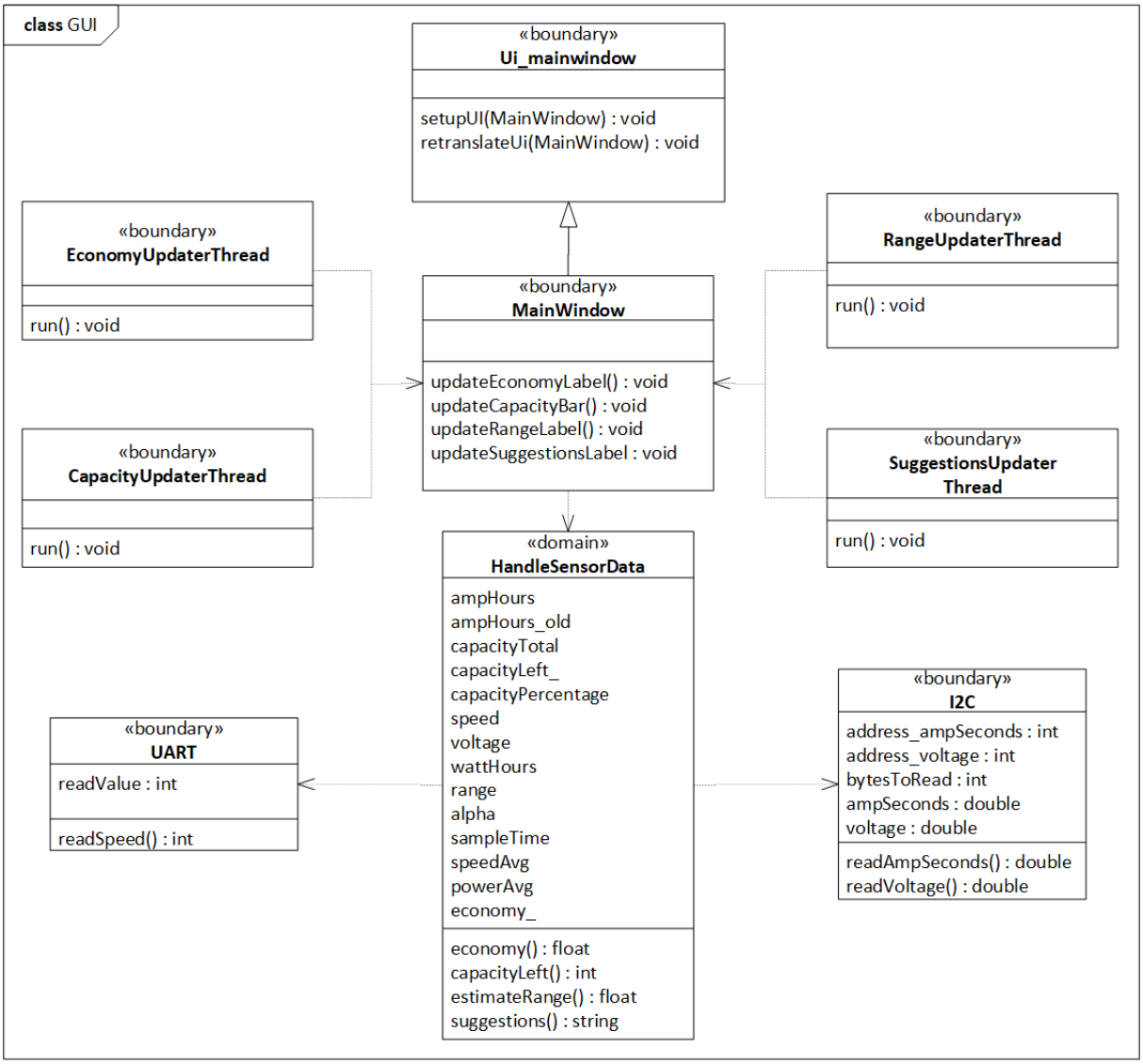
```
4
5       return self.range
```

11.3.1.5 Implementering af feedback til brugeren

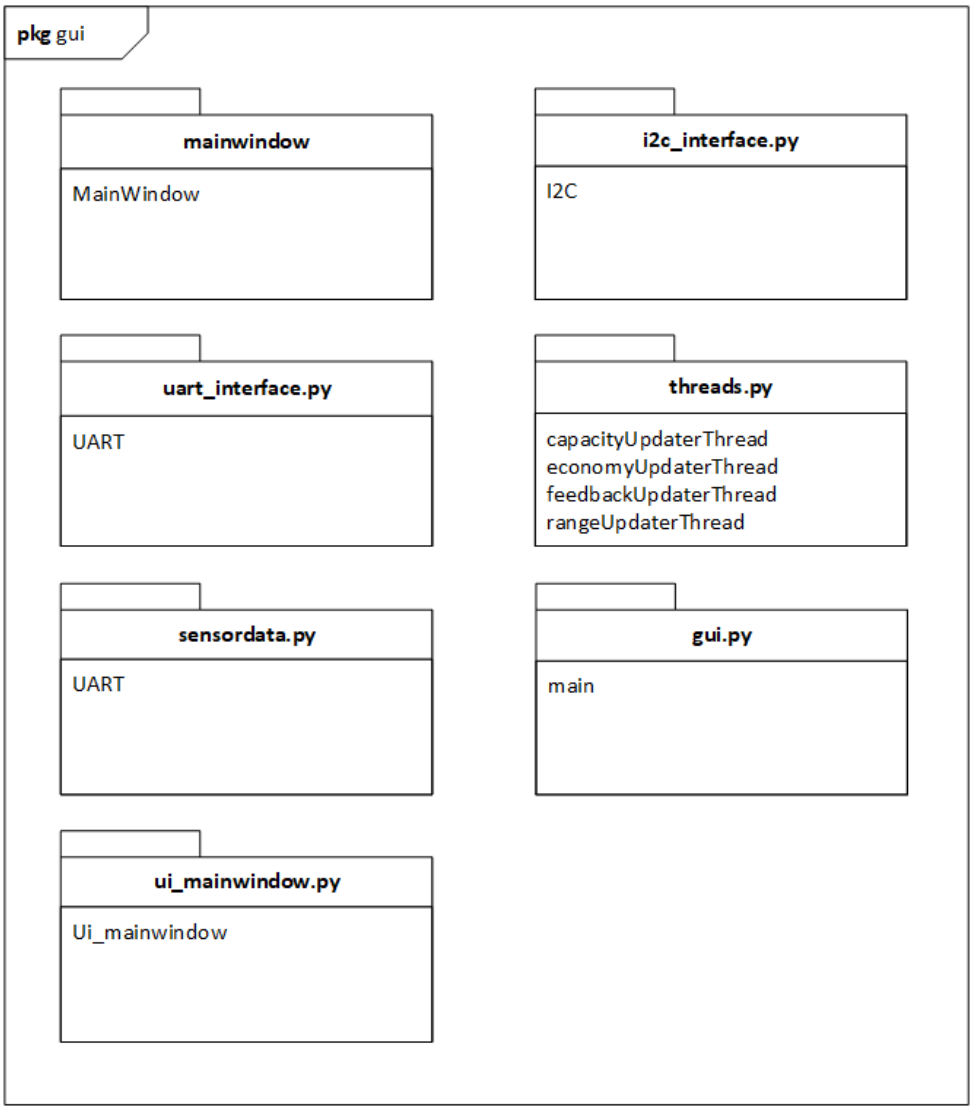
Da det ikke var muligt at implementere hall effekt sensoren på bilen, er feedback til brugeren ikke implementeret i det omfang, som egentlig var tiltænkt. Dette kræver nemlig at bilens effektforbrug testes ved forskellige hastigheder. Feedback til brugeren i den endelige implementering består blot af en beskeden "Battery Capacity below 20%!", der vises hvis den resterende kapacitet af batteriet kommer under 20%.

11.3.1.6 Endeligt klassediagram

Design af GUI'en leder frem til det endelige klassediagram, som kan ses på figur 31. GUI'ens organisering i python-moduler kan ses på figur 32.



Figur 31: Endeligt klassediagram for GUI hvor alle metoder og variable er medtaget.



Figur 32: Pakkediagram som viser hvordan software til GUI'en er organiseret i python-moduler. gui.py indeholder programmets main-funktion.

11.3.2 Pålidelig kommunikation

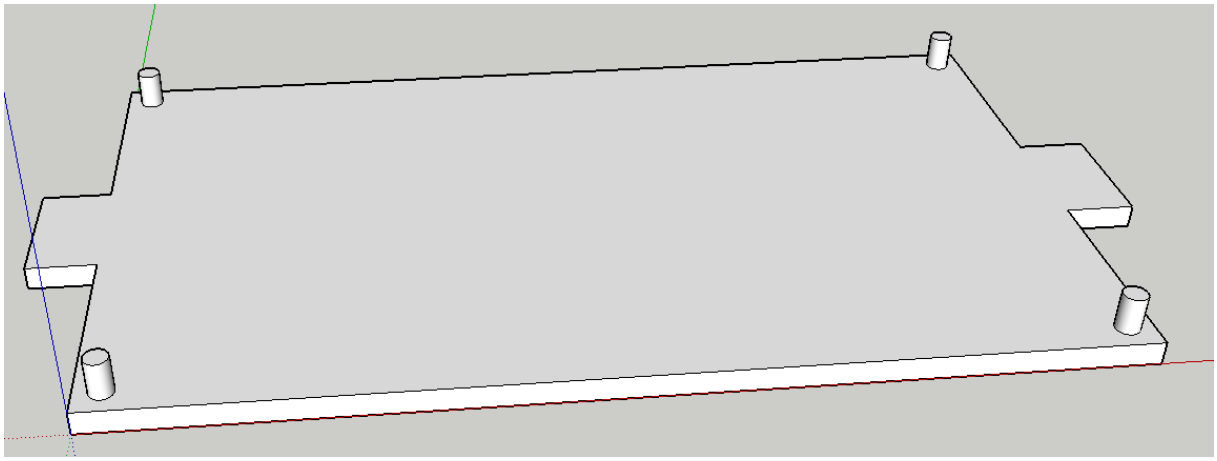
For at opnå en pålidelig kommunikation mellem Raspberry Pi og PSoC, er der anvendt en modificeret I2C kommunikation. Standard I2C protokol er blevet brugt, men der er indført en checksum, som gør det muligt at verificere om det rigtige er blevet sendt og modtaget. Der bliver sendt 64 bits data fra PSoC Strømmåler til Raspberry Pi 2 i form af en double. Checksummen til denne bliver udregnet ved at dele de 64 bits op i fire 16 bits integers, som bliver lagt sammen, dette kaldes for sum. Herefter bruges første komplement af sum og dette bliver checksummen. Denne checksum bliver sendt foran data bytes.
På modtagersiden, bliver sum af data bits igen udregnet. Efter dette bliver summen lagt sammen med den modtagne checksum, og da checksummen er første komplement af summen, skulle det gerne resulterer i ettaller i alle bits. Hvis dette er sandt bruges de modtagne data bits, ellers bliver der spurgt om en retransmission. Dermed er der implementeret både linklag i form af I2C standard og transportlag med retransmission og checksum.

11.3.3 Fremstilling af skærmkasse til GUI

Der er til skærmen blevet fremstillet en kasse. Denne kasse er blevet tegnet i et 3D-tegneprogram, ved navn "SketchUp". "SketchUp"er et gratis program, som udbydes af Google [6].
For at kunne lave kassen er der til at starte med blevet set på skærmens dimensioner. Det viste sig dog, at afstandene for skærmen var blevet forstået forkert, hvilket var skyld i at kassen blev for stor. Dette gjorde at skærmen sad løst i kassens udhuling til skærmen. Dette blev efterfølgende opdaget, og der blev lavet en ny skærm med de egentlige størrelser for både skærm og skærmkasse.
Skærmkassen forventes ikke at blive udsat fysiske belastninger, såsom at blive kastet med eller tunge objekter ligges på kassen, men derimod at kassen blot vil blive placeret i bilen således at brugeren kan benytte sig af systemet. Derfor er kassen ikke blevet lavet særlig tyk, da dette ville forlænge fremstillingstiden.
Kassen er derudover blevet lavet med 4 huller, et hul i hvert hjørne, hvor disse huller skal bruges til at holde skærmkassens låg på plads, som så holder skærmen på plads. Skærmkassen ses herunder:

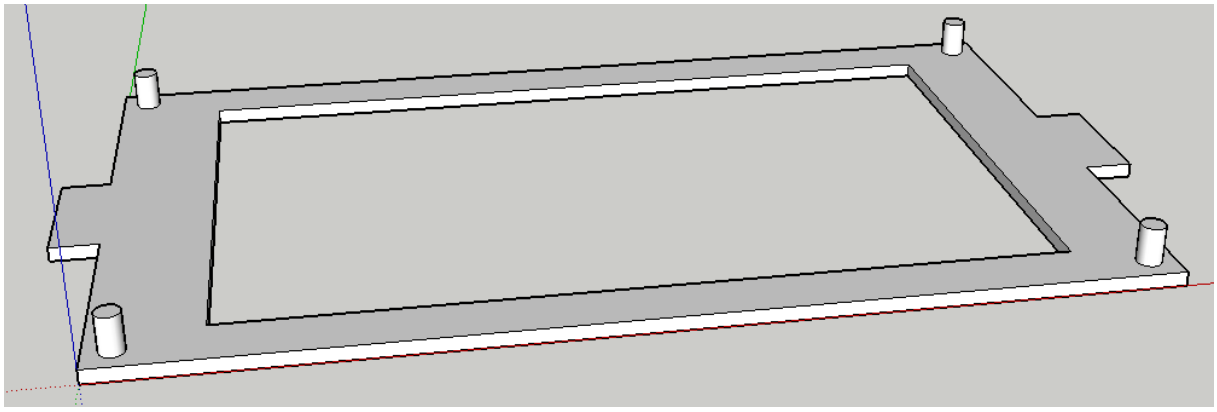
Figur 33: Den endelige skærmkassen, som skærmen skal ligge/sidde i

Derudover er der blevet lavet et låg, som er gennemsigtigt. Dette er gjort med henblik på at man selvfølgelig skal se, hvad der står på skærmen. Låget blev i første omgang lavet således at det lignede en plade, som har til formål at beskytte skærmen fra eksempelvis ridser eller fra at falde ud af kassen. Se figuren nedenfor:



Figur 34: Låget til skærmkassen, set fra 3D-tegneprogrammet

Dog skal der her gøres opmærksom på, at der på daværende tidspunkt ikke var blevet taget højde for, at skærmen har touch-interface. Med en gennemsigtig plade, som låg, vil man kunne se hvad der er på skærmen, men ikke bruge touch-funktionen. Hvis skærmens touch-funktion skal bruges, er det ikke praktisk at man ikke kan røre skærmen. For at løse dette problem, har låget fået nogle ændringer. Låget er blevet gjort tyndere, hvor den oprindeligt var 5mm, og der blev lavet en udhuling, som er åben således at man kun kan røre indenfor skærmens touch-funktions felt. Denne udhuling skal laves så den stadig holder skærmen på plads, selvom der nu er mindre overflade til at holde skærmen på plads.

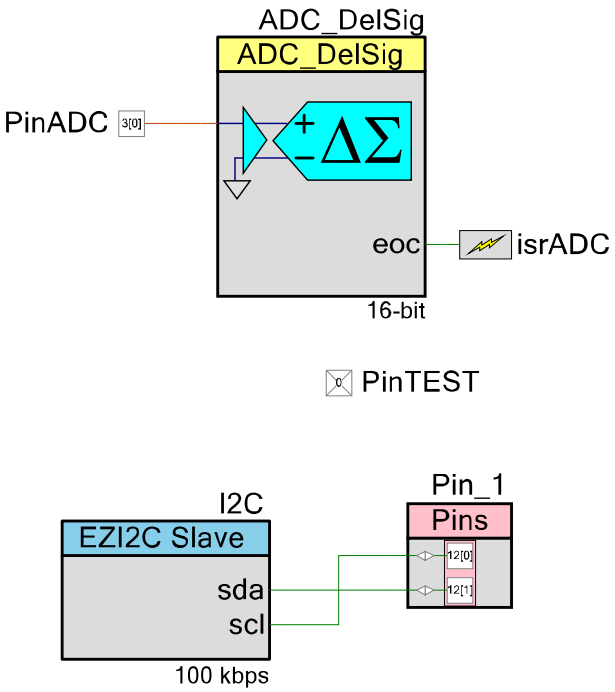


Figur 35: Låget til skærmkassen efter ændringer, set fra 3D-tegneprogrammet

Foroven ses et billede af det nye låg til skærmkassen, hvor ændringerne er lavet. Det kan herpå ses, at området, hvor feltet til touch-interface befinder sig, er nu ledig og åbent. Det er derfor nu muligt at røre på skærmen og benytte sig af skærmens touch-interface. Derudover kan det også ses at tykkelsen af låget er blevet mindsket, så den er 3mm.

11.3.4 Strømmåler

Strømmåleren har ud fra arkitekturen to funktionaliteter, der skal implementeres, for at den vil fungere tilfredsstillende. Disse funktionaliteter er måling af strømmen og kommunikation via I2C til Rpi2.



Figur 36: TopDesign af strømmåleren og spændingsmåleren.

11.3.4.1 Måling af strøm

Til måling af strømmen sættes en hall effekt sensor op til en ADC i PSoC. Hall effekt sensoren omsætter så en målt strøm til en bestemt spænding, og denne spænding kan derefter konverteres tilbage til den egentlige strøm med en formel givet i hall effekt sensorens datablad [14].
ADC'en er en Delta-Sigma ADC opsat med en samplingsfrekvens på 2 kHz. Dette gøres, da Delta-Sigma'en er bedre til moderate præcise målinger med moderate frekvenser end f.eks. en SAR ADC. Det er nødvendigt at måle spændingenma præcist, da en lille spændingsændring kan medføre en markant ændring i den målte strøm.
Det målte resultat indsættes i en ligning, der konverterer spændingen til den tilsvarende strøm, der løber igennem kablet, hvor hall effekt sensoren er opsat.

Listing 6: Konverteringen fra spænding til strøm foretages af funktionen her. Koden er designet ud fra sammenhængen givet fra databladet.

```
1 float volt2ampere( float input ) {
2     return invG * ( input - offset);
3 }
```

Disse resultater kan derefter bruges til at bestemme integralet af strømmen, der løber igennem kablet. Beregning af integralet er nødvendigt, da det skal bruges til beregning af batteriets kapacitet. Integralet af strømmen udregnes med trapez-reglen og implementeres som et filter ved brug af en differensligning i koden. Implementeringen gennem trapez-reglen medfører en konstant underestimering af værdien af det endelige integral. Dette accepteres blot, og vil ikke blive håndteret i koden.

Listing 7: Implementeringen af integralet som et filter

```
1 double integrator ( float input ) {
2     double output = outputDelay + 0.5*input + 0.5*inputDelay;
3
4     outputDelay = output;
5     inputDelay = input;
6
7     return output;
8 }
```

11.3.4.2 Kommunikation gennem I2C

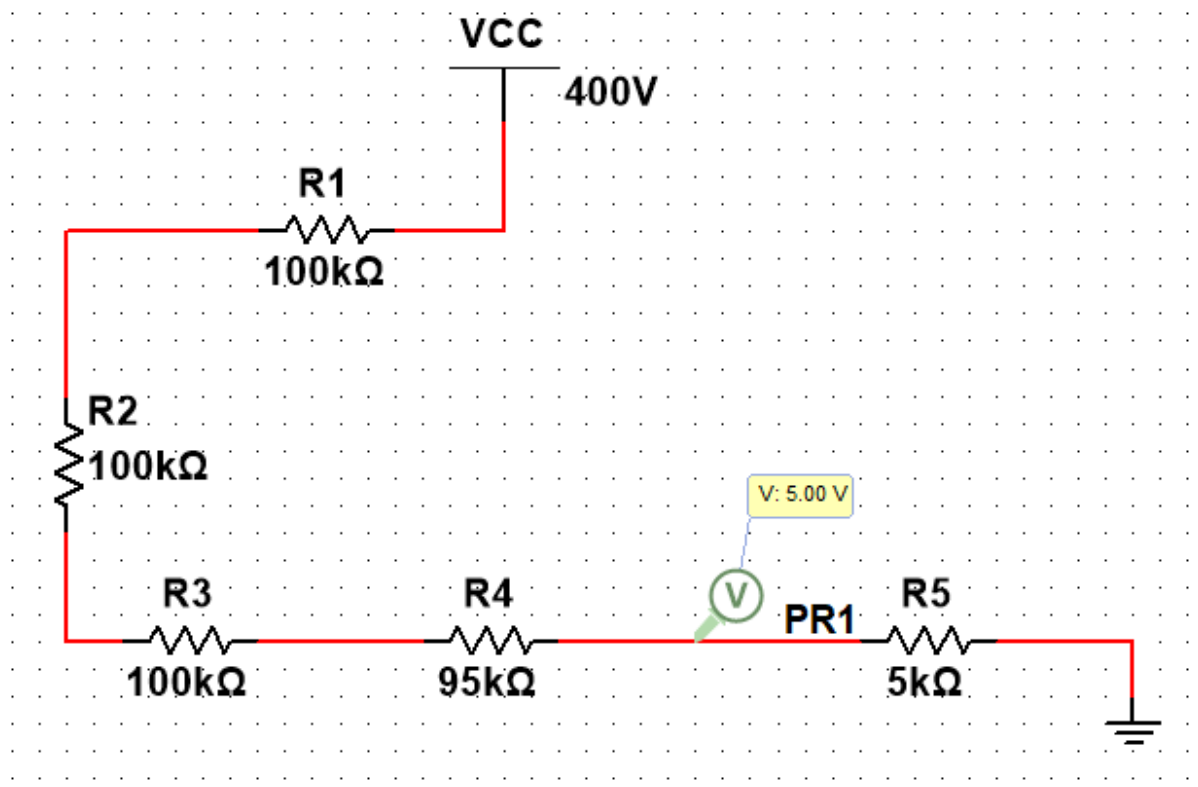
I programmet bliver der i PSoC'en opsat en I2C-komponent med en bestemt adresse. Programmet venter så i et loop på besked fra masteren om, at den skal videresende det seneste måleresultat. I2C'en er opsat på samme måde i designet for spændingsmåleren.

11.3.5 Spændingsmåler

Spændingsmåleren har ud fra arkitekturen tre funktionaliteter, der skal implementeres for at den vil virke fyldestgørende. Disse funktionaliteter er nedkonvertering af spænding, måling af spænding og kommunikation gennem I2C til RPi. Den sidste funktionalitet bliver implementeret på samme måde som i strømmåleren.

11.3.5.1 Nedkonvertering af spænding

Eftersom spændingsmåleren skal trække så lidt strøm som muligt, er det blevet valgt, at der skal bruges modstande i kΩ. Da en spænding på 400V skal justeres ned til 5V, hvilket er en dæmpning på 80 gange, bygges der en spændingstransducer, som har en total modstand på 400kΩ fra 400V til jord. Denne totale modstand er delt op, så der kan måles hen over de sidste 5kΩ, hvilket giver de ønskede 5V.



Figur 37: Multisim diagram af spændingstransduceren.

Som det kan ses på figur 37, er der også tilføjet en forstærkning. Denne er blevet nødvendig, efter at kredsløbet er blevet testet.

11.3.5.2 Måling af spænding

Til måling af spændingen bruges PSoC'ens indbyggede ADC. Denne vil konvertere den analoge spænding på målebenet til et tal, der kan bruges i softwaren. Der er to muligheder for valg af ADC nemlig SAR og Delta-Sigma. Den sidstnævnte blev valgt, da denne måling ikke er nødvendig at foretage så ofte. Der var derfor ingen gevinst ved at bruge en SAR over en Delta-Sigma.
Samplingsfrekvensen af ADC blev sat til det samme som for strømmåleren nemlig 2 kHz. Dette er muligvis en for høj frekvens til dette formål, men det er alligevel en lav samplingsfrekvens for PSoC'en.
Grundet den analoge hardware, der mindsker spændingen med en faktor 80 inden den når ADC'en, er det nødvendigt at lave lidt databehandling af målingerne. Her ganges faktoren på målingen, så det egentlige resultat opnås. Denne faktor kaldes for konverteringskoefficienten.

11.4 Forsyning

Hardware design baseres på undervisningen på 4. semester. Igennem undervisningen i Analog System Design (ASD) og Elektromagnetisk Kompatibilitet (EMC), lærer de studerende om de teoretiske og analytiske baggrunde af hardware og designprocedure igennem undervisning, og afprøvning af teoristoffet igennem opgaveregning og laboratorieøvelser.
Baseret på disse erfaringer fra 4. semesters undervisning, blandet med tidligere semestres projek-

ter, designes hardware til dette projekt. Målet for design af hardware er en funktionel prototype, der skal anvendes i projektet, samt eventuelle fremtidigt arbejde/forbedringer.

11.4.1 Specifikation af range

Der er opstillet følgende krav til range af spænding til spændingsregulatorerne:

Bloknavn	Funktionsbeskrivelse	Signal	Tolerance/Range	Kommentar
LD1117	Spændingsregulator omsætter 12VDC til 3.3V 800mA	12VDC	3.135VDC-3.465VDC	Udgangsspændingen fra spændingsregulator
LM350	Spændingsregulator omsætter 12VDC til 5VDC 3A	12VDC	4.75VDC-5.25VDC	Udgangsspændingen fra spændingsregulator
L7805	Spændingsregulator omsætter 12VDC til 5V 1.5A	12VDC	4.75VDC-5.25VDC	Udgangsspændingen fra spændingsregulator
L7805	Spændingsregulator omsætter 12VDC til 5V 1.5A	12VDC	4.75VDC-5.25VDC	Udgangsspændingen fra spændingsregulator

Tabel 3: DC spænding range tabel

11.4.2 5V spændingsregulering - L7805CV

DC strømforsyning for både PSoC [8] og RPi [13] leveres af bilens cigarettænder, som forsyner med 12V spænding og 100W effekt. Ved ekstern strømforsyning, kan PSoC kan forsynes ml. 3.3V til 5V, hvorimod Rpi 2 kan forsynes med 5V.
Forsyningen omsættes derfor til 5V forsyning ved hjælp af en spændingsregulator. En serieregulator, hvor den optagne strøm stort set er den samme som udgangsstrømmen til belastningen og den afsatte effekt er tilnærmelsesvis proportional med udgangsstrømmen [31], foretrækkes derfor i dette tilfælde. En serieregulator nedsætter spændingen fra en ustabil effektforsyning til en stabil og kortslutningssikker forsyningslinje. Udgangsspændingen ligger på en fast værdi over referencebenet, der kaldes COM eller ground. Serieregulatoren har et lavere internt forbrug på 8mA for 78-typerne. Effekttabet er stort set proportionalt med udgangsstrømmen I_O.
Valget af regulator faldt på L7805CV, da dens udgangsspænding er 5V, og har termisk beskyttelse ved overbelastning, hvilket sikrer microcontrollerne mod overbelastning, samt mulighed for forsyning med over 1.5A. [40]

11.4.3 3.3V spændingsregulering - LD1117

Spænding for RPi GPIO max. 3.3V[11], så en spændingsregulator, der omsætter 12VDC til 3.3VDC skal findes. LD1117, som værkstedet råder over, leverer 3.3VDC spænding og op til 800mA strøm, hvilket passer til RPi GPIOs behov.

11.4.4 5V spændingsregulering - LM350

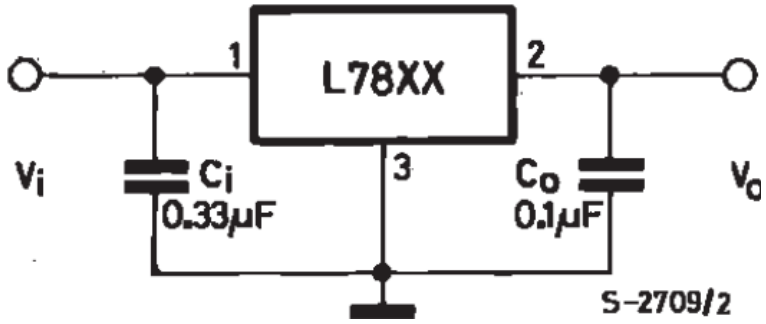
RPi 7"touch screen kræver 5V og min. 2.5A. [37]. LM350 spændingsregulatoren til touchskærmen anvendes, da den leverer 5V og 3A.[38].

11.4.5 5V spændingsregulering - L7805CV

RPi 2, som er monteret på 7" touch skærm kræver også 5V og 200mA [13] og 650mA under tunge udregninger. Strømtrækket på L7805CV udgør 1.5A, og lukker automatisk ned hvis den bliver overbelastet. For at forebygge driftsstop, må strømtrækket fordeles. Da den første L7805 allerede forsyner 2 x PSoC og 1 x hall effect sensor, ansås det ikke forsvarligt at implementere RPi 2 i denne spændingsregulator. Vi forsøgte med at implementere RPi 2 og 7"touch screen i samme LM350-spændingsregulatoren, men touchskærmen meldte fejl grundet for lav spænding og for stort strømtræk. Vi valgte derfor at adskille RPi 2 og 7"touch skærm med hver sin spændingsregulator for at forebygge driftsstop.

11.4.6 Designprodedure

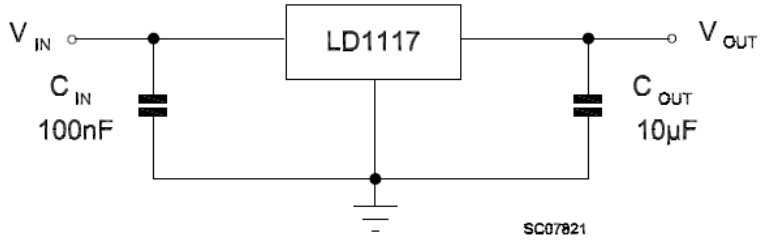
DC range er nu specificerede, og komponenter skal nu specificeres med inspiration af spændingsregulatorernes datablade.



Figur 38: L7805CV spændingsregulator fra datablad
[40]

Databladets figur anvendes som inspirationskilde. Komponentliste:

- 1 x 12V DC forsyning
- 1 x 0.33 μF kondensator (C_1)
- 1 x L7805CV spændingsregulator
- 1 x 0.1 μF kondensator (C_2)
- 3 x USB Type A hun stik til 2 x PSoc og 1 x RPi

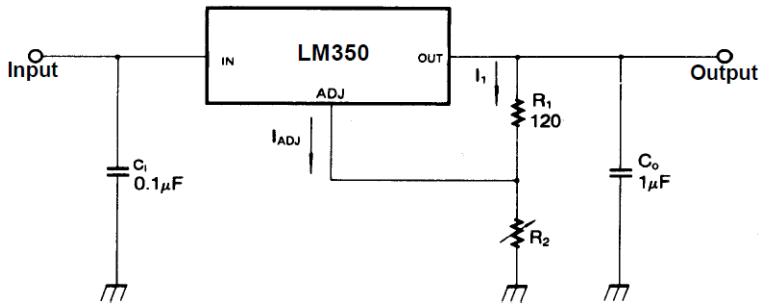


Figur 39: LD1117 spændingsregulator fra datablad
[39]

Databladets figur anvendes som inspirationskilde. Komponentliste:

- 1 x 12V DC forsyning

- 1 x 100 nF kondensator (C_1)
- 1 x L7805CV spændingsregulator
- 1 x 10 μF kondensator (C_2)

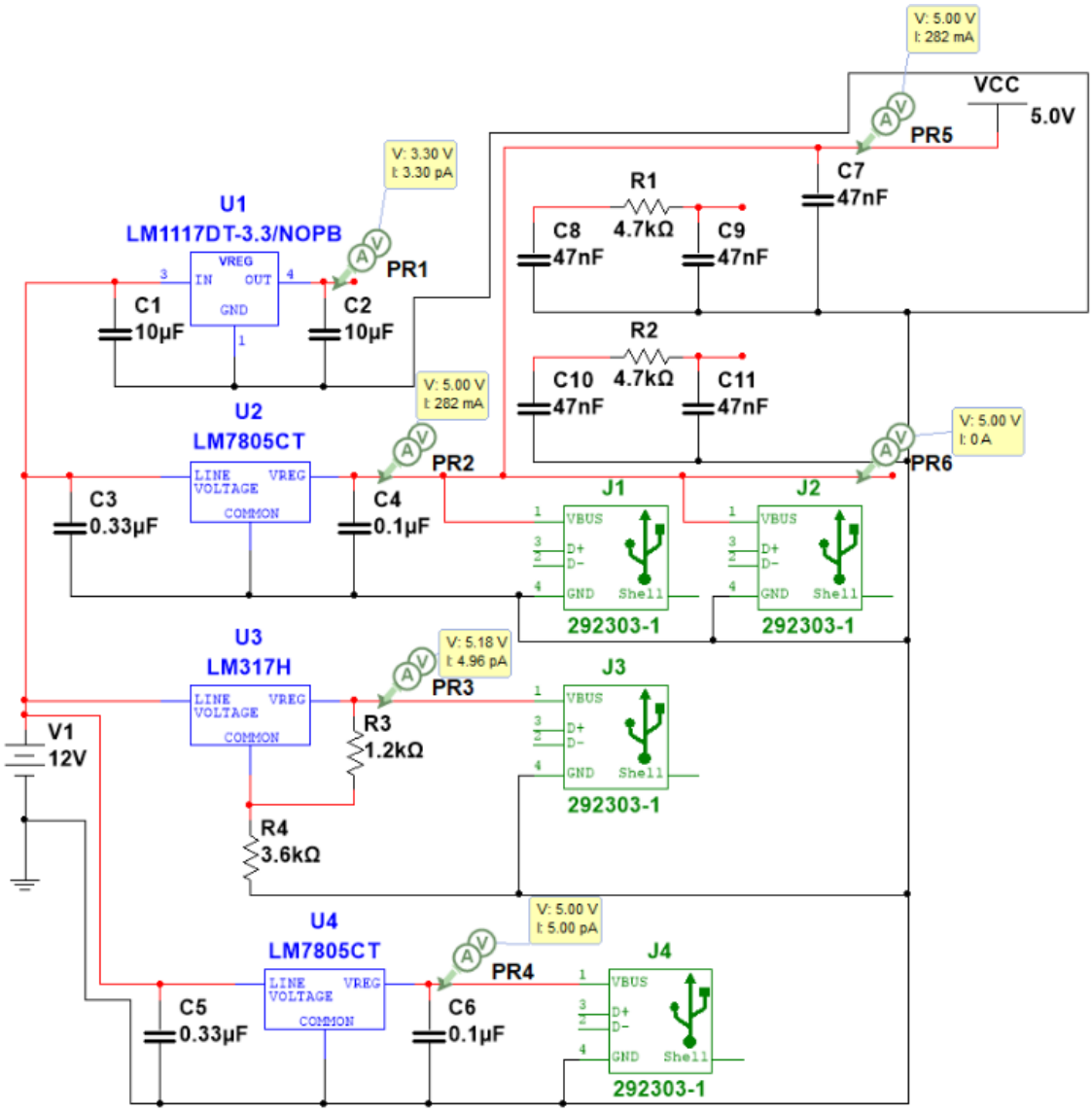


Figur 40: LM350 spændingsregulator fra datablad
[38]

Databladets figur anvendes som inspirationskilde. Komponentliste:

- 1 x 12V DC forsyning
- 1 x 1.2 kΩ modstand (R_1)
- 1 x 3.6 kΩ modstand (R_2)
- 1 x L7805CV spændingsregulator

Inspireret af databladende, implementeres kredsløbet i Multisim og simuleres, inden det implementeres i et veroboard:



Figur 41: Multisim-simulation af Forsyning

Spændingen er nu reguleret og strømforbrug er inden for cigarettænderens rammer. Designkriterierne er opfyldt. Kredsløbet bygges, så målinger kan foretages.

12 Resultater

Projektets fokus har været at udvikle en række delsystemer til forbedring af brugeroplevelsen i en elbil. I forbindelse med udvikling af EVManager er der udviklet en accepttest, der fungerer som en test af om det færdige system lever op til de krav, der er blevet fastlagt. Ved afslutningen af arbejdet med produktet blev accepttesten gennemgået og resultaterne noteret. Resultatet af denne accepttest er opsummeret i nedenstående tabeller, der viser accepttest for hhv. funktionelle og ikke-funktionelle krav. Se Tabel 4 og Tabel 5.

12.1 Resultater for accepttest af funktionelle krav

Test:	**Resultat:**	**Kommentarer:**
Use case 1: Opstart af Battery Management System	Ikke godkendt.	Da dele af Battery Management System ikke er implementeret på Elbilen, kan denne use case ikke godkendes. Use casen er dog gennemført ved brug af en simulering, hvor de ikke-implementerede dele er simuleret som stubbe.
Use case 2: Batteristatus og optimering af kørsel	Ikke godkendt.	Da dele af Battery Management System ikke er implementeret på Elbilen, er det ikke muligt at godkende denne use case. Use casen er dog gennemført med succes, hvor de ikke-implementerede dele er simuleret som stubbe.
Use case 3: Smartlygter	Godkendt.	Hvert punkt i accepttesten er blevet opfyldt. Smartlygte prototypen virker efter hensigten.
Use case 4: Parkeringssensor	Godkendt.	Hvert punkt i accepttesten er blevet opfyldt. Parkeringssensor prototypen virker efter hensigten.
Use case 5: Afslut Battery Management System	Godkendt.	Battery Management System afsluttes som forventet.

Tabel 4: Resultater for accepttest af funktionelle krav.

12.2 Resultater for accepttest af ikke-funktionelle krav

Test:	**Resultat:**	**Kommentar:**
Krav 1: Battery Management System skal kunne måle strømme op til 300 A DC.	Godkendt.	Den bestilte Hall effekt sensor kan måle strømme op til lige over 300 A selvom det af databladet fremgår, at den kan måle op til 750 A.
Krav 2: Battery Management System skal kunne måle spændinger op til 400 V DC.	Godkendt.	Spændingstransduceren i Battery Management System muliggør måling af spændinger op til 400 V.
Krav 3: Smartlygter skal kunne justere lysvinklen med en opløsning på 2 °.	Godkendt.	Step-motoren formår at få vinkel justeringen af smartlygterne ned på 0.9° ved hjælp af halv-step
Krav 4: Smartlygter skal kunne måle afstande fra 1 m til 40 m.	Godkendt.	Laser Distance Sensor 701A formår at måle afstande under 1 m og helt op til 50 m
Krav 5: Smartlygter skal være i stand til at sende data om målt afstand mindst 1 gang i sekundet.	Godkendt.	Laser Distance Sensor 701A formår at sende data om den målte afstand hurtigere end 1 gang i sekundet
Krav 6: Parkeringssensor skal kunne måle afstande fra 2 cm til 4 m.	Godkendt.	Efter test med parkeringssensoren vides der at den kan måle fra 2 cm til 500 cm.
Krav 7: Parkeringssensor skal kunne måle afstand med en afvigelse på mindre end 10%.	Godkendt.	Parkeringssensoren her en afvigelse på 0,26 % ud fra 200 målingere.

Tabel 5: Resultater for accepttest af ikke-funktionelle krav.

13 Diskussion

Dette projekt har resulteret i tre prototyper af delsystemer til optimering af en elbil. De forskellige prototyper virker optimalt med få afvigelser. Kravene for et 4. semesterprojekt er alle inkorporeret i projektet. Funktionaliteten af produkterne er opnået ved hjælp af viden fra størstedelen af 4. semester fag.

Et fokuspunkt for implementeringen gennem projektet, var at kunne sende og modtage signaler mellem delenhederne. Dette er i høj grad lykkedes, og kommunikation mellem PSoC, Raspberry Pi og diverse sensor er testet og fungerer optimalt.

Ved implementeringen af hardware lå fokuspunktet på at få individuelle hardwaredele til at virke hver for sig. Dette lykkedes ved hjælp af enhedstest og små testprogrammer enten på Raspberry

Pi eller PSoC. Ud fra dette er den samlede software i de forskellige prototyper blevet dannet, og dermed har testene dannet grundlag for de endelige softwareløsninger, der er implementeret i systemet.
I systemet er der blevet brugt en del sensorer, til at måle på omverdenen. Funktionaliteten af disse har fungeret efter hensigten, og har givet fornuftige og præcise resultater inden for de specifikationer vi har haft for dem.
Softwaren til GUI er implementeret med fokus på at det underliggende software skal kunne håndtere at modtage data fra PSoC, imens GUI kører og opdaterer data løbende. Til dette er der blevet brugt trådet programmering samt Qt til design. Da Qt ikke er en del af den almene undervisning, har dette taget tid at sætte sig ind i. GUI'en fungerer optimalt og efter hensigten.

14 Konklusion

Denne projektrapport har præsenteret resultaterne af et projekt omhandlende forbedring af de elektriske systemer i en elbil. Projektet har fokuseret på tre uafhængige delsystemer. Battery Management System sætter føreren af elbilen i stand til at opnå information om energiøkonomi, batteriets resterende kapacitet, elbilens rækkevidde og feedback om kørselsmønster. Derudover er der udviklet to prototyper i form af selvjusterende nærlys kaldet Smartlygter og en Parkeringssensor til assistering ved parkering.
Battery Management System nåede ikke at blive implementeret på elbilen pga. leveringsproblemer med den Hall effekt sensor, der skulle anvendes. Derfor var det nødvendigt at simulere de forskellige enheder i form af spændingstransducer, strømmåler og elbilens CAN bus. En fungerende brugergrænseflade blev implementeret og software til de forskellige enheder blev integreret med denne. Simulering af Battery Management System viste, at systemet fungerede som forventet. Dog kunne use case 1 og 2 ikke godkendes, da implementeringen på elbilen ikke var fuldendt.
Parkeringssensor blev udviklet som en separat prototype til at assistere føreren af elbilen med parkering. Dette system var ikke tiltænkt direkte implementering på elbilen men i stedet som et proof-of-concept delsystem. Parkeringssensoren blev implementeret og fungerede i overensstemmelse med de opstillede funktionelle og ikke-funktionelle krav.
Smartlygter blev udviklet som en separat prototype til automatisk justering af nærlys på elbilen. Dette system var ligeledes udtænkt som en separat prototype, hvor målet ikke var implementering på elbilen. Smartlygter bestod alle de opstillede funktionelle og ikke-funktionelle ved gennemførelse af accepttesten.
Ved arbejdet med projektet har gruppen haft en del udfordringer med at få centrale komponenter hjem. Dette har forhindret den endelige implementering af Battery Management System på elbilen. Projektgruppen har dog været i stand til at løse dette problem delvist ved at simulere tilstedeværelsen af de manglende enheder. Derved har det alligevel været muligt at bekræfte funktionaliteten af centrale dele af systemet og foretage de nødvendige tests.

15 Fremtidigt arbejde

Målet med dette projekt var overordnet at udvikle en række produkter, der skulle gøre elbiler mere attraktive for potentielle købere. Disse produkter bestod af et Battery Management System med tilhørende grafisk brugergrænseflade, selvjusterende forlygter samt en parkeringssensor. Alle tre produkter blev implementeret succesfuldt som fungerende prototyper. En del ting nåede dog ikke at blive implementeret.

15.1 Battery Management System

For Battery Management Systemets vedkommende mangler der en egentlig måling af elbilens totale kapacitet. Uden denne måling er den totale batterikapacitet i princippet ukendt idet den elbil, der testes på er fra 2012 og ganske givet ikke længere har den i databladet oplyste kapacitet på 16 kWh. Det har derfor været nødvendigt ved test af software, at antage at kapaciteten af batteriet har været som hvis den var ny. Desuden mangler Battery Management System en måde hvorpå det kan detekteres om batteriet er blevet opladt eller afladt uden at systemet har været tilsluttet. Pga. placeringen af strømsensoren på bilen er det nemlig ikke muligt at måle den strøm, der løber ind i batteriet ved opladning. Løsningen på dette problem er nævnt i analyseafsnittet, men den er ikke implementeret. Den største mangel ved Battery Management System er den manglende implementering på bilen. Dette skyldes systemets afhængighed af en strømsensor, der er godkendt til at kunne sidde under bilen. Leveringstiden på denne strømsensor viste sig at være omkring 2 måneder. Gruppen kunne også have tænkt sig at forbedre algoritmen til estimering af bilens rækkevidde. Algoritmen beregner bilens rækkevidde udelukkende ud fra den resterende batterikapacitet samt den gennemsnitlige hastighed og effektforbruget i det seneste minut. Algoritmen kunne forbedres ved at tage hensyn til flere faktorer såsom temperatur, der påvirker batteriets kapacitet. En forbedring kunne også være en vægtning af bilens tidligere kørselsmønstre således at et mere nøjagtigt estimat opnås.

15.2 Parkeringssensor

Parkeringssensoren blev implementeret som en selvstændig prototype. Dvs. målet var ikke at den skulle implementeres på elbilen. Dog kunne projektgruppen godt have ønsket sig, at data fra parkeringssensoren i form af afstanden til målte objekter kunne overføres til GUI'en. Evt. med en visuel indikation i form af en rød, gul eller grøn farve ift. hvor tæt bilen befinder sig på det givne objekt. Endvidere kunne parkeringssensoren forbedres ved at implementere endnu en sensor til anvendelse på fronten af bilen og kombinere dataindsamling fra disse to enheder på samme microcontroller. Parkeringssensoren kunne også forbedres ved tilføjelse af et bakkamera således at brugeren får visuel feedback.

15.3 Smartlygter

Smartlygter blev implementeret som en selvstændig prototype. Dvs. målet var ikke at den skulle implementeres på elbilen, da gruppen vurderede at dette kunne risikere at gøre skade på bilen. Laseren som blev anvendt til afstandsbestemmelse havde var ikke i stand til at opdatere den

givne afstand mere end ca. én gang i sekundet. Dette resulterede i et system med et vist delay ift. at regulere vinklen af lyset. Systemet kunne forbedres ved at anvende en laser, der er i stand til at reagere hurtigere. Dette ville føre til en væsentlig forbedring af systemet idet reguleringen af forlygternes vinkel ville kunne foretages hurtigere. En anden forbedring af systemet kunne være anvendelse af en mere præcis step-motor. Dvs. en step-motor hvor usikkerheden på antal grader, den bevæger sig per step er mindre. Det havde også været interessant at koble Smartlygter til GUI'en således at den automatiserede regulering af nærlyset kunne slås til eller fra ved tryk på en knap.

Referencer

[1] URL: https://en.wikipedia.org/wiki/Systems_Modeling_Language (sidst set 27.05.2018).

[2] URL: https://en.wikipedia.org/wiki/Unified_Modeling_Language (sidst set 27.05.2018).

[3] URL: http://k2.arduino.vn/img/2016/12/27/0/3422_88215469-1482851833-0-dieu-khien-pha.jpg (sidst set 27.05.2018).

[4] URL: https://electrosome.com/wp-content/uploads/2014/08/Working-of-HC-SR04-Ultrasonic-Sensor.jpg (sidst set 27.05.2018).

[5] URL: https://en.wikipedia.org/wiki/Fritzing (sidst set 27.05.2018).

[6] URL: https://www.sketchup.com (sidst set 27.05.2018).

[7] International Energy Agency. Global EV Outlook 2017. URL: https://www.iea.org/publications/freepublications/publication/GlobalEVOutlook2017.pdf (sidst set 01.03.2018).

[8] Cypress Semiconductor Corporation. PSoC 5LP CY8C58LP Family Datasheet. URL: http://www.cypress.com/file/45906/download (sidst set 16.04.2018).

[9] Delta Sigma Analog to Digital Converter (ADC_DelSig). Cypress.

[10] Distance Sensor VDM100-50. URL: https://www.pepperl-fuchs.com/usa/en/classid_53.htm?view=productdetails&prodid=63916 (sidst set 27.05.2018).

[11] RASPBERRY PI FOUNDATION. General Purpose Input/Output pins on the Raspberry Pi. URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md (sidst set 16.04.2018).

[12] iMiEV specifications. URL: https://www.mitsubishi-motors.com/en/showroom/i-miev/specifications/ (sidst set 27.05.2018).

[13] Introducing the Raspberry Pi 2 - Model B. adafruit learning system. https://cdn-learn.adafruit.com/down
the-raspberry-pi-2-model-b.pdf.

[14] kem.com. LEM. URL: https://www.lem.com/sites/default/files/products_datasheets/dhab_s_133.pdf (sidst set 16.04.2018).

[15] Laser Distance Sensor 701A-40 datasheet. Chengdu JRT Meter Technology.

[16] Laser Distance Sensor 701A-40 Manuel. Chengdu JRT Meter Technology.

[17] EV Propulsion llc. SOME BASIC EV CALCULATIONS. URL: http://www.ev-propulsion.com/EV-calculations.html (sidst set 04.03.2018).

[18] Long Term Review: 10,000 Miles In My Mitsubishi iMiEV. URL: https://insideevs.com/long-term-review-10000-miles-in-my-imiev/ (sidst set 28.02.2018).

[19] Richard G. Lyons. Understanding Digital Signal Processing. Third Edition. Prentice Hall, 2011, s. 334.

[20] MOSCoW. URL: https://en.wikipedia.org/wiki/MoSCoW_method (sidst set 26.02.2018).

[21] Newton-Cotes formulas. URL: https://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas (sidst set 27.05.2018).

[22] Nordeuropas største energilaboratorium åbnet i Aarhus. URL: https://stiften.dk/aarhus/Nordeuropas-stoerste-energilaboratorium-aabnet-i-Aarhus/artikel/425632 (sidst set 28.02.2018).

[23] obd. URL: https://en.wikipedia.org/wiki/On-board_diagnostics (sidst set 27.05.2018).

[24] pyqt. URL: https://riverbankcomputing.com/software/pyqt/intro (sidst set 27.05.2018).

[25] qt. URL: https://en.wikipedia.org/wiki/Qt_(software) (sidst set 27.05.2018).

[26] qtdesigner. URL: http://doc.qt.io/qt-5/qtdesigner-manual.html (sidst set 27.05.2018).

[27] Raspberry Pi 7 inch Touchscreen datasheet. element14.

[28] raspbian. URL: https://en.wikipedia.org/wiki/Raspbian (sidst set 27.05.2018).

[29] Scrum (software development. URL: https://en.wikipedia.org/wiki/Scrum_(software_development) (sidst set 16.05.2018).

[30] Sharp. URL: http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a710k_e.pdf (sidst set 27.05.2018).

[31] Tore Skogberg. T-005 Analogteknik v. 2.0. URL: http://www.torean.dk/artikel/Analogteknik.pdf (sidst set 16.04.2018).

[32] MIT Electric Vehicle Team. A Guide to Understanding Battery Specifications. URL: http://web.mit.edu/evt/summary_battery_specifications.pdf (sidst set 04.03.2018).

[33] Tjek bilens lygter. URL: https://fdm.dk/alt-om-biler/bilen-hverdagen/vedligeholdelse/tjek-bilens-lygter (sidst set 27.05.2018).

[34] Trapezoidal rule. URL: https://en.wikipedia.org/wiki/Trapezoidal_rule (sidst set 27.05.2018).

[35] TYPICAL LITHIUM ION TECHNICAL DATA. URL: http://www.ibt-power.com/Battery_packs/Li_Ion/Lithium_ion_tech.html (sidst set 01.03.2018).

[36] Ultrasonic Ranging Module HC-SR04. elec freaks.

[37] Unknown. URL: https://www.raspberrypi.org/documentation/hardware/display/README.md (sidst set 27.05.2018).

[38] Unknown. 3-Terminal 3A Positive Adjustable Voltage Regulator. URL: http://pdf1.alldatasheet.com/datasheet-pdf/view/53583/FAIRCHILD/LM350.html (sidst set 27.05.2018).

[39] Unknown. Adjustable and fixed low drop positive voltage regulator. URL: http://www.st.com/content/ccc/resource/technical/document/datasheet/99/3b/7d/91/91/51/4b/be/CD00000544.pdf/files/CD00000544.pdf/jcr:content/translations/en.CD00000544.pdf (sidst set 27.05.2018).

[40] Unknown. Positive voltage regulator ICs. URL: https://www.mouser.dk/datasheet/2/389/l78-974043.pdf (sidst set 16.04.2018).