AARHUS
UNIVERSITET
INSTITUT FOR ELEKTRO- OG COMPUTERTEKNOLOGI

Vejledning til gennemførelse af semesterprojekt 1

Version 1.12
31. januar 2023
Torben Gregersen
tg@ece.au.dk

Indholdsfortegnelse

Indledning

Semesterprojekt 1 (PRJ1) tager sit udgangspunkt i en problemformulering, som indeholder en beskrivelse af en elektrisk bil, der i en konkurrence skal gennemføre en bane med forhindringer på kortest mulig tid.

Problemformuleringen for bilprojektet er beskrevet i dokumentet ”1. semester-projekt, projektoplæg” [1].

Bilprojektet skal opfattes som et lille udviklingsprojekt, som er fælles for E- og SW-studerende på diplomingeniøruddannelsens 1. semester på Institut for Elektro- og Computerteknologi, Aarhus Universitet.

Der skal i dette projekt anvendes stort set samme tid på hardwareudvikling, som der skal anvendes på softwareudvikling.

Projektet skal resultere i et prototype-produkt, som er selve den elektriske bil. Udover dette skal projektet resultere i en projektrapport, der dokumenterer udviklingen af dette prototype-produkt og de resultater der er opnået.

Denne notes mål er at beskrive arbejdsmetoden for hvordan et projektarbejde, f.eks. bilprojektet, kan organiseres og udføres i praksis. Denne arbejdsmetode benævnes i det følgende som en proces. Processen, der beskrives her, er tilpasset processen for projektarbejde, der anvendes på de efterfølgende semestre. Forskellen mellem processen for gennemførelse af *semesterprojekt 1* og processen for gennemførsel af senere semesterprojekter er blot, at kompleksiteten øges i løbet af diplomingeniøruddannelsen til og med bachelorprojektet.

Rammerne omkring projektarbejdet er:

1. Samarbejde i gruppen
2. Projektgennemførelse og udfærdigelse af projektdokumentation
3. Projektadministration
4. Den komplette projektdokumentation

Disse rammer udgør et godt fundament for projektarbejdet.

1. Samarbejde i gruppen

Gruppens samarbejde er af stor betydning for et godt projektarbejde. Det, der har betydning for en projektgruppes velbefindende, er præcist de samme ting, som har betydning i alle mulige andre sammenhænge, hvor et antal personer skal foretage sig noget seriøst sammen. I en projektgruppe har grupperelaterede glæder, bekymringer, konflikter og magtkampe deres rod i de samme sociale mekanismer, og gruppen skal beskæftige sig aktivt med sagen, hvis den skal blive velfungerende.

Noget af det gruppen skal tage højde for, er:

1.1. Personlige ressourcer

Vi har alle stærke og mindre stærke sider, og det er selvfølgelig hensigtsmæssigt, hvis de enkelte medlemmer i gruppen bidrager med det, som de er bedst til. Alle skal have et godt overblik over projektet. Det er typisk gældende i projektgrupperne, at lyst og evner til teoretisk arbejde, laboratoriearbejde, programmeringsarbejde, at holde orden i filer og papirer og at holde humøret højt, vil være ujævnt fordelt blandt gruppens medlemmer.
Tages der hensyn til dette når arbejdsopgaverne fordeles, øges gruppemedlemmernes tilfredshed og selvfølelse, og gruppens ressourcer udnyttes optimalt.

1.2 Personlige relationer

Det er selvfølgelig rarest at være sammen med folk, man kan lide, men man skal være indstillet på også at kunne arbejde sammen med folk, man ikke bryder sig om. Det vil naturligvis være tåbeligt at sammensætte en projektgruppe udelukkende med personer, der ikke kan fordrage hinanden. Men eftersom vi ikke altid selv kan vælge vores samarbejdspartnere, skal vi øve os i at kunne samarbejde med "kedelige" personer. Vi skal træne os i at ”bøje os mod hinanden”.

1.3 Gruppeledelse

Projektgruppen ledes af en projektleder, som har fokus på at projektudviklingen planlægges og gennemføres hensigtsmæssigt, dvs. så de aftalte mål for projektet kan opnås. Projektlederen er et af gruppens medlemmer, som ønsker at have denne rolle.
Rollerne som projektleder kan eventuelt gå på skift, f.eks. med et 2-ugers interval.

1.4 Gruppe / Team

En gruppe er ikke nødvendigvis det samme som et team. Dannelsen af et team ud fra en gruppe er betinget af, at alle gruppemedlemmer føler et fælles ansvar for projektet. En gruppe kan ikke beslutte sig for at blive til et team på et indledende gruppemøde, hvor projektet startes op. Det er noget, der langsomt vokser frem, hvis de rette betingelser er til stede. En af de nødvendige betingelser er, at alle gruppemedlemmer er med i det, der foregår. En målestok for et velfungerende team kan hænge sammen med hvor mange af gruppemedlemmerne, der deltager i gruppens øvrige sociale aktiviteter.

1.5 Aftaler

Indgåede aftaler skal overholdes. Det gælder ikke alene aftaler, der er bogførte i mødereferater, men også mere "løse" aftaler, for eksempel en mundtlig aftale mellem to gruppemedlemmer. Hvis man ikke kan stole på hinanden, går gruppen mere eller mindre i opløsning, og resultatet af projektarbejdet bliver mangelfuldt.

Erfaringer med vejledning af semesterprojekter viser, at der ofte opstår problemer med samarbejdet i gruppen. Det kan typisk være problemer som:

- Nogle gruppemedlemmer udebliver fra aftalte møder uden at melde afbud
- Det er ikke aftalt hvordan ledelsesstrukturen i gruppen skal være, så en eller flere påtager sig en selvbestaltet lederrolle og styrer ”enevældigt”
- Det er ikke aftalt hvor stort et problem skal være, før vejlederen kontaktes
- Gruppemedlemmerne har ikke afstemt ambitionsniveauer
- Der er ingen kontrol med, om samarbejdet fungerer for alle
- Det er ikke aftalt hvordan omgangstonen i gruppen skal være. Den må ikke være krænkende for nogen
- Der anvendes ikke mødeindkaldelser med dagsorden
- Ingen tager referat fra møderne
- Der er ikke aftalt konsekvenser for den enkelte, hvis samarbejdet negligeres eller saboteres

Det er et krav, at der nedfældes og underskrives en samarbejdsaftale mellem medlemmerne i gruppen.

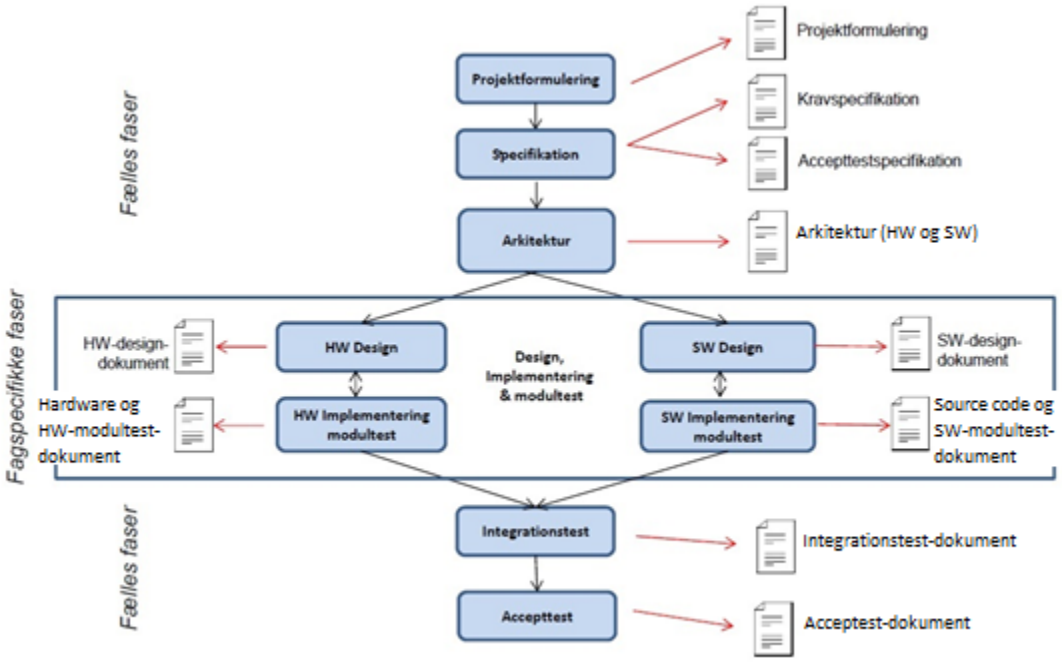
Det kan yderligere anbefales at nedfælde og underskrive en tilsvarende samarbejdsaftale mellem gruppen og vejlederen.

2. Projektgennemførelse og udfærdigelse af projektdokumentation

Arbejdsmæssigt udføres et projekt, som både indeholder udvikling af hardware (HW) og software (SW), i følgende faser:

- Problemformulerings-fase
- Specifikations-fase
- Arkitektur-fase
- Design-fase
- Implementerings- og modultest-fase
- Integrationstest-fase
- Accepttest-fase

Nedenstående figur beskriver et HW/SW-projekts typiske faser og dets output (artefakter):



Figur 1. HW/SW-projekts typiske faser og dets output (artefakter)

Undervejs i HW/SW-projektets faser, som er beskrevet ovenfor, opstår en række artefakter. Der opstår dels en række dokumenter som artefakter. Hensigten er også, at der skal opstå et samlet produkt som artefakt. Dette produkt kan opdeles i:

- et HW-produkt (en prototype eller et salgbart produkt)
- et SW-produkt (kildekode (source code), som oversættes til eksekverbar kode)

Artefakterne er derfor:

- Problemformulering (dokument)
- Kravspecifikation (dokument) – samtidigt: Accepttestspecifikation (dokument som skrives, men som ikke kan udfyldes med resultatet af accepttesten før produktet er færdigudviklet)
- HW/SW-arkitektur (dokument)

- HW-design (dokument)
- SW-design (dokument)
- Implementering af hvert enkelt HW-modul (veroboard, PCB etc.), som derefter gennemgår HW-modultest (som beskrives i et HW-modultest-dokument)
- Implementering af hvert enkelt SW-modul (source code -> eksekverbar kode), som derefter gennemgår SW-modultest (som beskrives i et SW-modultest-dokument)
- Integrationstest (som beskrives i et integrationstest-dokument)
- Accepttestspecifikation (dokument, som allerede er skrevet i kravspecifikations-fasen - de opnåede testresultater fra den gennemførte accepttest beskrives nu for hver enkelt test-case: der beskrives hvad der blev observeret og yderligere beskrives OK/ikke OK status for test-casen

I det følgende beskrives projektets faser og deres output/artefakter nærmere.

2.1 Problemformulerings-fase

(output/artefakt: problemformulering (dokument))

I problemformulerings-fasen beskrives overordnet, hvad projektet går ud på. I en "live-situation" svarer problemformuleringen til det, der kommer ud af de første henvendelser fra en potentiel kunde til et HW/SW-udviklingsfirma. Henvendelserne kan ske i form af mails, telefonsamtaler, møder etc., som beskriver et ønske fra kunden. Hvis udviklingsfirmaet vurderer, at det er muligt at løse opgaven, kan kravene efterfølgende beskrives nøje i et dokument: *problemformulering*.

I semesterprojekt 1 (PRJ1) er problemformuleringen på forhånd skrevet, dvs. "henvendelsen" til jer er allerede sket ved opstart af semesterprojekt 1.

2.2 Specifikations-fase

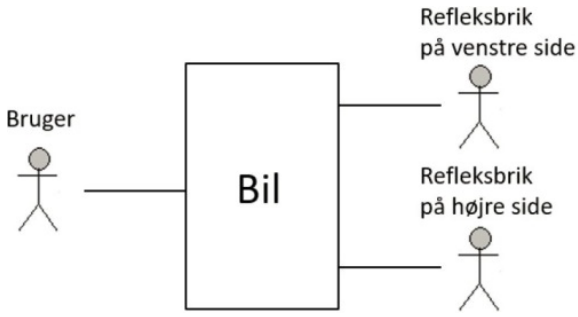
(output/artefakt: kravspecifikation (dokument))
(output/artefakt: accepttestspecifikation (dokument))

Hvis udviklingsfirmaet vurderer, at det er muligt at løse opgaven ud fra problemformuleringen (se ovenfor), kan kravene efterfølgende beskrives nøje i samarbejde med kunden.

I specifikations-fasen afklares alle forhold der vedrører rekvirenten af projektet, så udviklingsfirmaet og kunden er enige om, hvad der skal udvikles. Kravene skal alle være formuleret, så de er mulige at teste.

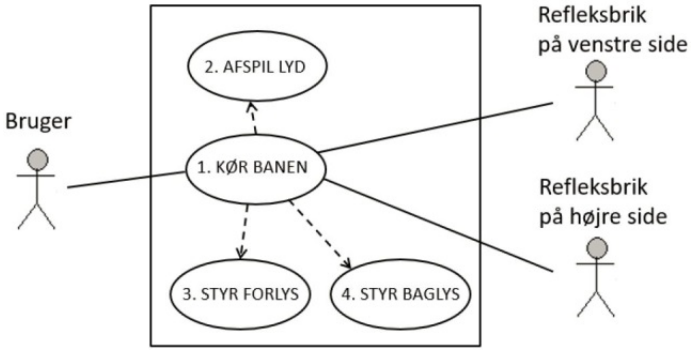
Der arbejdes ud fra et "top-down"-view, hvor hele systemet, både HW-mæssigt og SW-mæssigt, i første omgang beskrives som en "black box", der kan anvendes af en eller flere brugere (aktører). Aktører kan også være andre færdigudviklede systemer, som det system, der ønskes udviklet, skal kunne kommunikere med. Aktører kan både være primære og sekundære aktører. Primære aktører vises altid til venstre for systemkassen (her "Bil"). En primær aktør kan gribe direkte ind i systemets adfærd, f.eks. kan brugeren af bilen sætte bilen i gang. En sekundær aktør er nødvendig for at systemet kan fungere, men den har ikke en aktiv rolle i systemet. Sekundære aktører anbringes altid til højre for systemkassen" (her "Bil").

Præsentationen af aktører og system vises i et *Aktør-Kontekst Diagram* på figur 2:



Figur 2. Aktør-Kontekst Diagram for "Bil"

De enkelte aktørers roller beskrives efterfølgende for at uddybe figuren (se dokumentet "1. semester-projekt, projektoplæg" [1]). Herefter udarbejdes systemets *funktionelle krav* i samarbejde med kunden. Systemets samlede funktionalitet deles op i afgrænsede del-funktioner, "Use Cases". Denne opsplitning giver mulighed for at bevare overblikket i specifikations-fasen og i de følgende faser i udviklingsprocessen. For indledningsvist at skabe et overblik illustreres de "Use cases",
der anvendes i systemet, i en figur:



Figur 3. Use Case Diagram for "Bil"

Bemærk at "Use cases" navngives i bydeform (imperativ) for at opnå klarhed og præcision.

Funktionaliteten for hver enkelt "Use Case" beskrives herefter præcist i skemaer for at uddybe figuren (se dokumentet "1. semester-projekt, projektoplæg" [1]).
Nogle krav er ikke mulige at beskrive som funktionalitet (handling) i "Use Cases". Tekniske specifikationer som mål, vægt, systemets levetid, temperaturbestandighed etc. beskriver ikke funktionalitet. Derfor beskrives denne type krav som *ikke-funktionelle krav*. De ikke-funktionelle krav beskrives som regel umiddelbart efter beskrivelsen af de funktionelle krav.

Herefter beskrives HW/SW-systemets brugergrænseflade. Denne kan bedst forstås af læseren (kunden, efterfølgende HW/SW-udviklere), når de funktionelle krav er beskrevet forinden. Brugergrænsefladen defineres i samarbejde med kunden indtil der er opnået enighed. Herved er brugergrænsefladen "fastlåst", så systemets HW/SW-udviklere ikke "opfinder" funktionalitet, som kunden ikke har bestilt. Brugerfladen på bilen i 1. semesterprojektet er som regel yderst simpel, da den som regel kun består af en trykknap og et fejltæller-display. Dette kræver ikke en lang beskrivelse.

Udfordringen i specifikations-fasen er at specificere de aftalte krav i samarbejde med kunden, samtidigt med at kravene skal kunne forstås af det team af HW/SW-udviklere, som efterfølgende skal udvikle HW/SW-produktet.

Når alle krav er beskrevet præcist i kravspecifikationen, kan det nu beskrives, hvordan hele HW/SW-produktet skal testes. Dvs. der skal udføres en præcis beskrivelse af hvordan accepttesten af systemet skal udføres. Accepttest for hver enkelt "Use Case" beskrives med præcise testparametre, og det forventede resultat af testen beskrives præcist. Herved kan accepttesten af systemet gennemføres på en repeterbar måde, og evt. fundne fejl kan vises for den ansvarlige udvikler.
De ikke-funktionelle krav beskrives ligeledes med præcise test-parametre og præcise forventede resultater, så testen kan gentages (se dokument "1. semester-projekt, projektoplæg" [1]).
Når accepttesten skrives, kan man samtidigt verificere at kravene - såvel funktionelle som ikke funktionelle krav - er beskrevet tilstrækkeligt præcist. Alle krav skal være testbare!

2.3 Arkitektur-fase

(output/artefakt: HW/SW-arkitektur-dokument)

Denne fase har overordnet set til formål at gøre projektet overskueligt ved at analysere det, og herefter nedbryde det i mindre bestanddele – indtil der opnås overskuelighed.

I første omgang besluttes hvilke dele af projektet, der skal implementeres som HW, og hvilke dele der skal implementeres som SW.

Herefter skal der udføres en nedbrydning af hhv. HW-delen og SW-delen til mindre bestanddele – indtil der opnås overskuelighed. Bestanddelene navngives ofte på engelsk af hensyn til de værktøjer (simulatorer, compilere etc.), der skal anvendes til realiseringen af HW/SW-produktet.

HW nedbrydes i bestanddele, som kan kaldes blokke eller moduler.

SW nedbrydes ligeledes i bestanddele, som benævnes som moduler. Hvis der skal anvendes en *objektbaseret* SW-arkitektur, kaldes bestanddelene *moduler*. Hvis der derimod skal anvendes en *objektorienteret* SW-arkitektur, kaldes bestanddelene *klasser*. I bilprojektet, hvor sproget "C" oftest anvendes som programmeringssprog, kaldes bestanddelene derfor moduler, da "C" ikke er objektorienteret sprog. Anvendes et objektorienteret sprog (som f.eks. "C++") i bilprojektet, kaldes bestanddelene klasser.

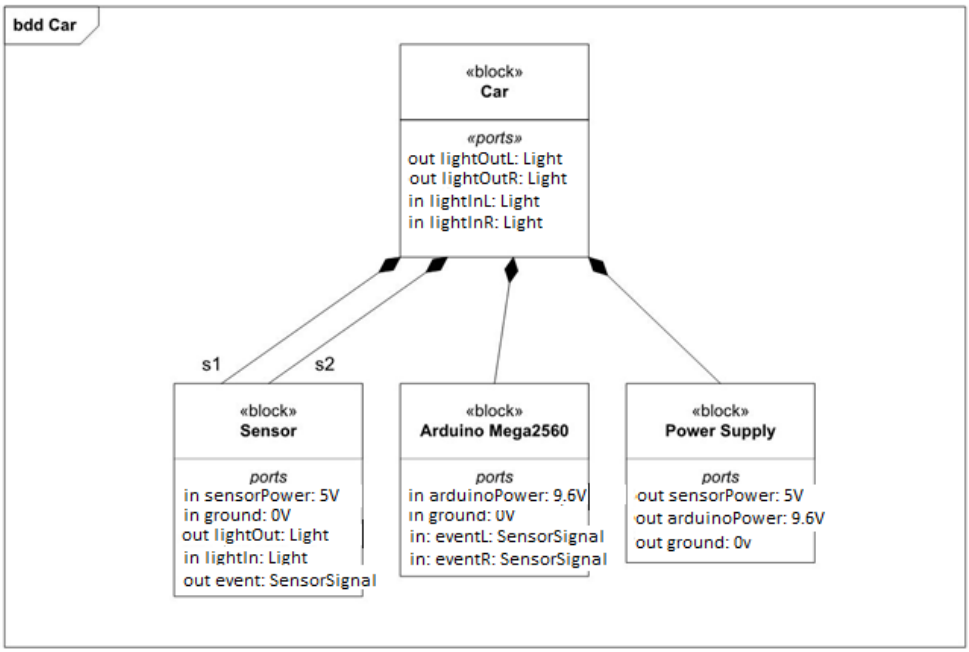
Det er muligt at udføre nedbrydning til overskuelighedsniveau uden i detaljer at vide, hvordan det enkelte modul/klasse skal udvikles. Der er kun brug for en beskrivelse af hvad modulet/klassen har ansvar for og en beskrivelse af samtlige væsentlige grænseflader mellem projektets forskellige SW- bestanddele. Dette giver frihed til senere i udviklingsforløbet at vælge mellem forskellige programmeringssprog til at udvikle de samme moduler/klasser.

Nedbrydningen til mindre bestanddele skal principielt være så detaljeret, at et mere detaljeret HW/SW-design af blokke/moduler/klasser kan overgives til underleverandører (andre HW/SW-udviklingsfirmaer, ansatte i forskellige HW/SW-afdelinger i eget firma, delgrupper af studerende i en projektgruppe etc.).

Overblikket over hhv. HW-arkitektur og objektbaseret/-orienteret SW-arkitektur kan skabes ved at udarbejde henholdsvis SysML-diagrammer til HW-arkitekturen og UML-diagrammer til SW-arkitekturen.

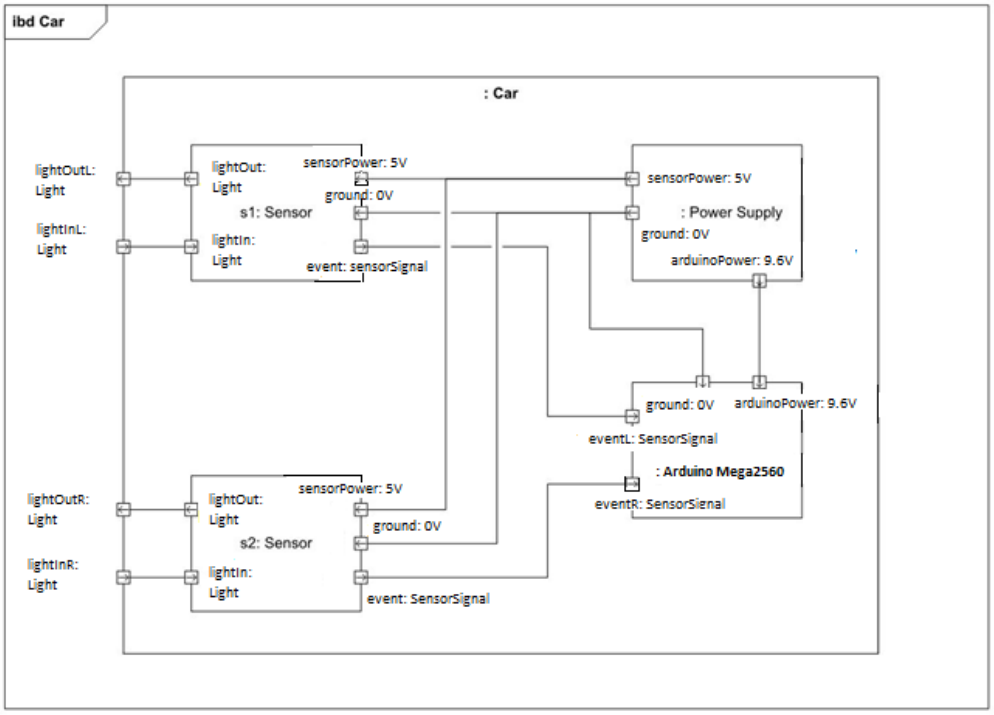
2.3.1 HW-Arkitektur

HW-arkitekturen kan udføres vha. SysML (System Modelling Language). HW-blokke kan i første omgang illustreres vha. et BDD (Block Definition Diagram), som nedbryder den samlede hardware i mindre bestanddele (blokke). Nedbrydningen kan fortsætte, indtil der opnås overskuelighed over den samlede hardware. Blokkene navngives og signalerne, der optræder på de enkelte porte i hver blok, navngives.



Figur 4. BDD for "Bil" (reduceret udgave)

I det BDD, der er vist ovenfor på figur 4, er det ikke muligt at vise, hvordan de enkelte HW-signaler forbindes mellem blokkene. Det er nødvendigt at vise disse forbindelser for at færdiggøre HW-arkitekturen. Forbindelserne kan illustreres vha. et IBD (Internal Block Diagram). Portene er forinden navngivet (f.eks. lightIn, event) i BDD. Det er kun selve HW-forbindelserne, der er tilføjet i IBD, sammenlignet med informationerne i BDD.



Figur 5. IBD for "Bil" (reduceret udgave)

IBD i figur 5 viser de signaler, der optræder mellem de enkelte blokke. Hvert signal starter og ender i en port. En port er den fysiske grænseflade på blokken. Det er vigtigt at hvert signal har entydige porte.

I stil med ”Use Case”-figuren i kravspecifikation kan BDD og IBD ikke alene forklare blokkenes og signalernes funktionalitet. Deres funktionalitet skal forklares præcist vha. forklarende tekst i tabeller.

Først beskrives blokken. Nedenstående tabel er et eksempel på en blokbeskrivelse. Der er mange muligheder for at skrive denne dokumentation med varierende detaljeringsgrad afhængigt af det aktuelle problem. Det kan være praktisk at have blokbeskrivelse og blokdiagram (BDD) samlet.

Blok	**Funktionalitet**
Sensor	Detekterer lys, som reflekteres fra en refleksbrik
Arduino Mega2560	Processerer input fra Sensor
Power Supply	Forsyner Sensor og Arduino Mega2560 med spænding

For at fuldende grænsefladebeskrivelsen skal signalerne behandles detaljeret.

Signalbeskrivelsen kan med fordel samles i en tabel for alle signaler, evt. sorteret alfabetisk. Denne tabel kan anvendes når grænsefladerne skal designes, testes etc.

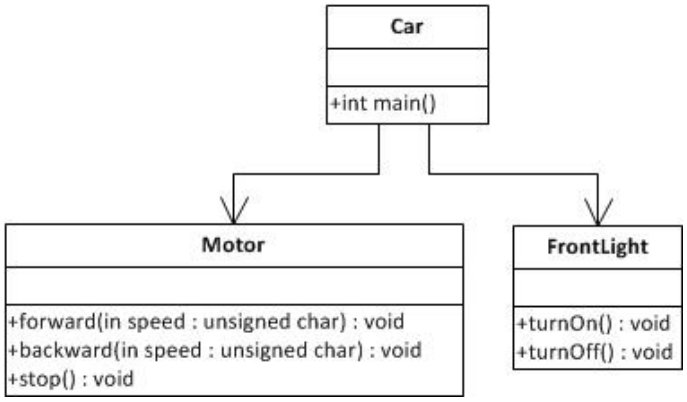
Signalnavn	**Funktion**	**Område**	**Port 1 (source)**	**Port 2 (destination)**	**Kommentar**
ground	Reference til analog spænding	0,0V	Power Supply, ground	Sensor1, ground Sensor2, ground Arduino Mega2560, ground	Stel
sensorPower	Forsyningsspænding	4.9-5.1V	Power Supply, sensorPower	Sensor1, sensorPower Sensor2, sensorPower	
arduinoPower	Forsyningsspænding	7-12V	Power Supply, arduinoPower	Arduino Mega2560, arduinoPower	
lightOut	Fysisk lys		lightOut		Lys ud til refleksbrik
lightIn	Fysisk lys			lightIn	Lys ind fra refleksbrik
event	Indikerer modtaget fysisk lys	0.0V-5.0V	eventL eventR	Arduino Mega2560, eventL (INT0 = Port D, pin0) Arduino Mega2560, eventR (INT1 = Port D, pin1)	Signaler fra sensorerne som aktiverer Arduino’ens Interrupt-inputs

Bemærk at terminalnavnene eventL og eventR er abstrakte og kortfattede, så der er plads dem i BDD’et og IBD’et. Terminalerne er beskrevet præcist i tabellen som (INT0 = Port D, pin0) og (INT1 = Port D, pin1). Herved er det beskrevet præcist hvordan HW-blokkene skal forbindes, og herved er det også beskrevet præcist hvordan systemets SW skal tilgå disse porte.

2.3.2 SW-Arkitektur

SW-arkitekturen kan udføres vha. UML (Unified Modeling Language). SW-arkitekturen fastlægges ved at finde de navneord, som optræder i kravspecifikationen. Disse analyseres for at beslutte om de kan være kandidater til at optræde som moduler/klasser i systemet. Følgende eksempel viser et objektbaseret design på en del af bilen, hvor modulets/klassens navn og navnet på hver enkelt funktion er angivet. Hvis der skal programmeres i et objektorienteret sprog som ”C++” kaldes en funktion for en metode. Modulernes/klassernes indbyrdes anvendelse er illustreret med pile. En pil rettet mod et modul /klasse indikerer, at der kaldes en eller flere funktioner/metoder i denne klasse.
I stil med SysML’s BDD og IBD kan illustrationen af moduldiagrammet/klassediagrammet ikke stå alene. De enkelte

moduler/klasser skal beskrives med en detaljeringsgrad, der gør det muligt for andre SW-udviklere at udføre et detaljeret design på modulerne/klasserne.



Figur 6. Moduldiagram/Klassediagram for "Bil" (reduceret udgave)

Hver modul/klasse, undtagen modulet/klassen som indeholder main(), repræsenterer en header- *og* en source-fil. Ovenstående eksempel på et statisk moduldiagram/klassediagram repræsenterer derfor 2 header-filer og 3 source-filer (den 3. source-fil indeholder main()). Header-filerne indeholder funktionernes/metodernes prototyper og source-filerne indeholder funktionernes/metodernes implementering (source code). Da SW-arkitekturen beskriver grænseflader mellem blokkene/klasserne er det kun *public* funktioner/metoder og deres parametre, der skal beskrives. Funktioner/metoder, der er *private*, dvs. interne i modulet/klassen, skal *ikke* beskrives i SW-arkitekturen.

Modulbeskrivelse/Klassebeskrivelse af Motor

Ansvar: Modulets/klassens ansvar er at kontrollere bilens motor, så den kan køre fremad eller baglæns med variabel hastighed. Modulets/klassens ansvar er også at standse bilens motor

Funktioner/Metoder:

`void forward ( unsigned char speed )`
Parametre: Den ønskede speed (0-100), som er den procentuelle angivelse af motorens maksimale ydelse
Returværdi: Ingen
Beskrivelse: Omsætter parameteren speed til en PWM duty factor, som styrer motoren. Herefter sættes motorens kørselsretning, så bilen kører fremad

`void backward ( unsigned char speed )`
Parametre: Den ønskede speed (0-100), som er den procentuelle angivelse af motorens maksimale ydelse
Returværdi: Ingen
Beskrivelse: Omsætter parameteren speed til en PWM duty factor, som styrer motoren
Herefter sættes motorens kørselsretning, så bilen bakker

`void stop( void )`
Parametre: Ingen
Returværdi: Ingen
Beskrivelse: Får motoren, og dermed bilen, til at standse

Beskrivelsen af hver enkelt funktion/metode skal være tilstrækkelig præcis til, at en anden person kan designe/implementere funktionen/metoden ud fra beskrivelsen. Meget simple funktioner/metoder beskrives eventuelt ikke.

Efter afsluttet HW/SW-arkitektur kan de enkelte HW-blokke/moduler og SW-moduler/klasser nu uddelegeres til udviklere, som kan udføre design i en så omfattende detaljeringsgrad, at de er klar til at blive implementeret.

2.4 HW/SW-Design-fase
(output/artefakt: HW/SW-design-dokumenter)

Formålet med HW/SW-design-fasen er at skabe et overblik, der er så detaljeret, at det er muligt at implementere produktets HW og SW. Designfasen er baseret på de beslutninger, der er taget i HW/SW-arkitektur-fasen.

2.4.1 HW-Design
På basis af HW-arkitektur-fasen, hvor grænsefladen til hver enkelt HW-blok er beskrevet, udføres nu et detaljeret design af hver enkelt HW-blok. Der udarbejdes et diagram (schematic), hvor der gøres rede for hver enkelt komponent.

Redegørelsen sker i form af præcise forklaringer til figuren med diagrammet, suppleret med de nødvendige beregninger af hver enkelt komponents værdi. Beregninger sker på basis af de valgte komponenters datablade og modulets interface-beskrivelse, som er fastlagt i arkitektur-fasen.

I det følgende vises et eksempel på design af et simpelt HW-modul:

LED fabrikat: LITEON
Part No. : LTW-2S3D7-012A

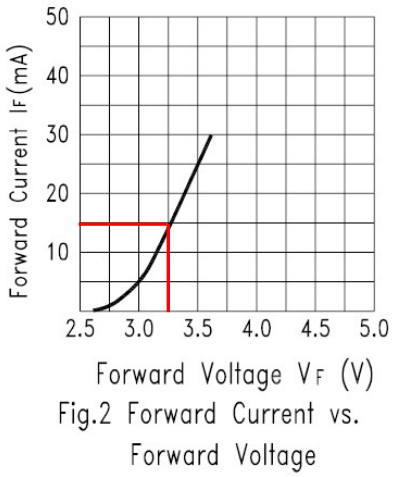
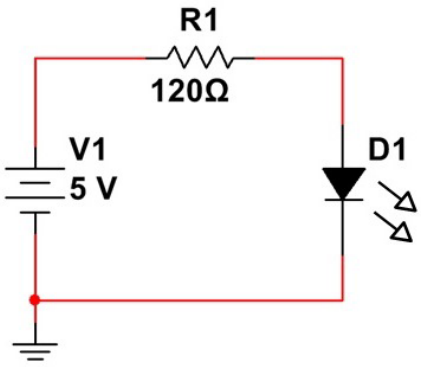


Fig.2 Forward Current vs. Forward Voltage

Strømmen gennem lysdioden D1 besluttes til at være 15 mA
Ifølge databladet giver dette et spændingsfald over D1 på 3,25V
Forsyningsspændingen V1 er 5V
Spændingsfaldet over R1 er da: VD1 = 5V – 3,25V = 1,75V
R1 kan derfor beregnes således: R1 = 1,75V / 15mA = 116,7 Ω ~ 120 Ω



Figur 7. Design af et simpelt HW-modul

Der udarbejdes styklister for samtlige komponenter, der skal anvendes. Hvis der skal anvendes PCB (Print Circuit Board) i implementeringsfasen, skal layout'et til dette PCB beskrives i den detaljerede HW-design fase.
Alt ovenstående HW-design beskrives i et HW-design dokument.

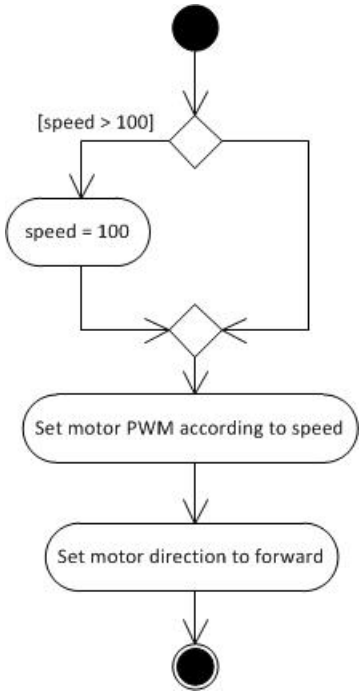
En stykliste for ovenstående diagram (schematic) kan se således ud:

Antal	Symbol på diagram	Komponent	Fabrikat/type
1	R1	Modstand	120 ohm, 1/4W
1	D1	LED	LITEON, Part No. : LTW-2S3D7-012A

2.4.2 SW-Design
På basis af SW-arkitektur-fasen, hvor grænsefladerne til hver enkelt SW-modul/klasse blev beskrevet, udføres nu et SW-design, der er så detaljeret, at alle væsentlige moduler/klasser, deres input/outputparametre og evt. deres interne algoritmer bliver veldefinerede. Designprocessen fortsættes, indtil der er klarhed om, hvordan den tilhørende source

code (implementering) kan skrives. Hvis de interne algoritmer er komplekse, kan det være nødvendigt at beskrive dem vha. af UML-aktivitetsdiagrammer (activity diagrams) og/eller vha. UML-tilstandsdiagrammer (state diagrams).
I bilprojektet er den algoritme, der håndterer optælling af passerede refleksbrikker, kombineret med frem- og tilbagekørsel, temmelig kompleks. Denne algoritme kan med fordel forklares vha. et UML-aktivitetsdiagram og/eller et UML-tilstandsdiagram.

UML-aktivitetsdiagram (activity diagram) for funktionen/metoden forward() i modulet/klassen Motor:



Figur 8. Design af en funktion/metode i et SW-modul/klasse vha. et UML-aktivitetsdiagram

I stil med det detaljerede HW-diagram skal figurerne understøttes af en præcis, teknisk forklaring ledsaget af evt. nødvendige beregninger.
Der er *ikke* vist et eksempel på anvendelse af UML-tilstandsdiagrammer (state diagrams) i dette dokument. Anvendelsen af disse gennemgås grundigt på 2. semester i forbindelse med 2. semesterprojektet.

Alt ovenstående SW-design beskrives i et SW-design dokument.

2.5 Implementerings- og modultest-fase

(Output/artefakt: HW: prototype/færdigt produkt (veroboard/PCB))
(Output/artefakt: SW: source code, som oversættes til en eksekverbar fil)
(Output/artefakt fra modultest: HW-modultest-dokument og SW-modultest-dokument)

2.5.1 HW-Implementering og modultest

På basis af HW-arkitekturen og det detaljerede HW-design bygges de enkelte HW-blokke/moduler, i første omgang som prototyper. Afhængigt af det miljø, som blokkene/modulerne skal fungere i, bygges de som ”fuglerede”, på veroboard eller på PCB. Miljøfaktorer, som afgør hvordan den designede HW skal implementeres, kan f.eks. være: mekanisk påvirkning (vibrationer), frekvensområde (høje frekvenser kræver korte, velovervejede forbindelser), EMC, kemisk påvirkning, temperaturpåvirkning etc.
Hver implementeret HW-blok/modul testes omhyggeligt i laboratoriet. Hvis det ikke fungerer korrekt, rettes fejlen. Medmindre der er tale om monteringsfejl eller løse forbindelser, er det typisk nødvendigt at ændre HW-modulets

design. Det kan yderligere evt. være nødvendigt at ”gå tilbage” og ændre arkitekturen. Er der rettet i HW-design og/eller HW-arkitektur, skal de tilhørende dokumenter opdateres.

2.5.2 SW-Implementering og modultest

På basis af SW-arkitekturen og det detaljerede SW-design skrives source code for hver enkelt SW-modul/klasse. Der udføres en modultest af hver metode i klassen på den HW, som klassen skal styre. F.eks. testes den SW-modul/klasse i bilprojektet, der styrer motoren på ”den ægte HW” (H-bro + selve motoren). Er dette HW-modul/klasse ikke færdigudviklet, kan SW-modulet/klassen testes på en simuleret HW (måleinstrumenter, lysdioder etc.), men sluttelig skal det eftervises, at SW-modulet/klassen kan styre HW-modulet som ønsket. Hvis SW-modulet/klassen ikke fungerer korrekt – og hvis HW-blokken med sikkerhed fungerer korrekt, rettes SW-fejlen. Det er typisk nødvendigt at ændre SW-modulets/klassens design. Det kan evt. yderligere være nødvendigt at ændre arkitekturen. Er der rettet i SW-design og/eller i SW-arkitektur, skal de tilhørende dokumenter opdateres.

2.6 Integrationstest-fase

(output/artefakt: Integrationstest-dokument)

Efter afsluttet implementering og modultest af HW/SW-modulerne samles disse moduler gradvist til et færdigt system i udviklingslaboratoriet. Når systemet er samlet fuldstændigt, og det ser ud til at fungere, er systemet klar til gennemførelse af accepttest.

Bilen, som anvendes til 1. semester-projektet, ryster og vibrerer under kørslen, så det kan betale sig at implementere den designede HW så solidt, at der ikke opstår løse forbindelser (og dermed funktionssvigt) under kørslen (konkurrencen).

Det kan også anbefales at lade alle stelforbindelser (ground) udgå fra et fælles stelpunkt på bilen for at undgå brumsløjfer (bilens motor udsender meget elektrisk støj). Det kan yderligere anbefales at sno stelledninger om signalledningerne for at undgå at signalerne forstyrres af elektrisk støj fra motoren. Den snoede stelafskærmning omkring signalledningerne forbindes kun til stel i den ene ende af forbindelsen. Korte forbindelser kan anbefales.

2.7 Accepttest-fase

(output/artefakt: accepttestspecifikation (dokument))

Det færdige system testes i henhold til accepttestspecifikationen, som blev udarbejdet i kravspecifikations-fasen. Systemet testes grundigt sammen med kunden (i PRJ1: vejlederen), resultatet af hver enkelt test-case beskrives og der markeres ”OK” eller ”Ikke OK” for hvert enkelt test-case i accepttesten. I *semesterprojekt 1* skrives en præcis konklusion på resultaterne fra accepttesten i slutningen af projekt-dokumentationen.

3. Projektadministration

Administrationen af et projektforløb skal sikre at projektgruppen:

- Har overblik over projektforløbet
- Ved hvem, der skal lave hvad, hvornår
- Ved hvor langt man er nået
- Ved hvor meget man mangler
- Kan reetablere udviklingsarbejdet i tilfælde af uheld

3.1 Samarbejdskontrakt

Som nævnt tidligere er det et krav at projektgruppen starter med at udarbejde en skriftlig samarbejdskontrakt. Denne kontrakt skrives af gruppen i fællesskab, så vidt muligt i fuld enighed. Aftalen præsenteres for vejlederen, hvorefter den underskrives af alle gruppemedlemmer. Samarbejdskontrakten kan evt. revideres, hvis dette skønnes nødvendigt, og hvis dette accepteres af alle medlemmerne. I referatet fra møderne vil det således fremgå, om samarbejdet fungerer i forhold til de aftaler, der blev indgået i starten af projektet.

Det er vigtigt, at det er hver enkelt gruppe, der laver sin egen samarbejdskontrakt (og ikke anvender en slags "standardkontrakt"), da gruppen derved får et ejerskab af aftalen, og denne bliver tilpasset kulturen i den konkrete gruppe. Oftest kan aftalen skrives på en enkelt A4-side. Som minimum bør aftalen indeholde følgende punkter:

- Hvor ofte holdes gruppemøder.
  Hvordan indkaldes til møderne (og af hvem)?
  Hvem udformer dagsorden?

- Med hvor kort varsel kan et medlem melde afbud til et møde?
  Hvad er konsekvensen, hvis et medlem udebliver fra et møde?
  Er der kun konsekvens ved gentagne udeblivelser?

- Hvem tager referat af møderne?
  Hvem udsender referatet?

- Hvordan ledes gruppen?
  Er der en eller flere faste projektledere, eller går projektlederrollen på skift?
  Hvilket ansvar og hvilke opgaver har projektlederen?

- Hvordan afgøres, om et problem har en karakter, så vejlederen bør informeres?

- Hvad er gruppens ambitionsniveau (vil vi blot netop bestå, eller går vi efter toppræstationen)?
  Har alle gruppens medlemmer samme ambitionsniveau (det er vel ikke forventeligt)? Hvis det ikke er tilfældet, noteres hver enkelt medlems ambitionsniveau.
  Hvordan tilfredsstilles de enkelte medlemmers ambitionsniveauer?

- Hvilken omgangstone er vi enige om at bruge i gruppen?

- Hvad er konsekvensen for et medlem, der ikke overholder samarbejdskontrakten?
  Skal der være en form for sanktion?
  I hvilke situationer vil vi inddrage vejlederen i for eksempel konfliktløsning

3.2 Tidsplan

Ved projektets opstart, når problemformulerings-fasen er overstået, skal projektgruppen hurtigst muligt udarbejde en tidsplan, f.eks. som et *Gantt chart,* efter nedenstående model:

Fase/uge	1	2	3	4	5	6	7	8	9
Kravspecifikation									
Accepttestspecifikation									

Figur 9. Eksempel på et Gantt chart til fastlæggelse af tidsplan

Det kan være nødvendigt at justere tidsplanen undervejs, og dermed arbejdebelastningen pr. tidsenhed, for at opnå det bedst mulige projektresultat.
Inddrag gerne PRJ1-vejlederen i udarbejdelsen af tidsplanen, da det kan være vanskeligt at estimere tidsforbruget for de enkelte faser i udviklingsprocessen.

3.3 Mødeindkaldelse
En vigtig del af projektadministrationen er at der indkaldes til møde med jævne, gerne helt periodiske mellemrum. Mødeindkaldelse skal foregå på en måde, så alle har mulighed for at se den. En skabelon for en indkaldelse til et møde, hvor vejlederen deltager, kan se således ud:

Indkaldelse til vejledermøde # nn
Dato:
Tid:
Sted:
Deltagere:
Dagsorden

1. Valg af mødeleder
2. Valg af referent
3. Godkendelse af referat fra forrige møde
4. Opfølgning på aktionspunkter fra forrige møde
5. ”Her kan indføjes ekstra punkter”
6. Gennemgang af tidsplan
7. Nye aktionspunkter til næste møde. Hvem gør hvad.
8. Tidspunkt for næste møde
9. Evt.

Et fast punkt på dagsordenen for hvert møde i gruppen bør være refleksion over samarbejdet i gruppen. Herved sikres, at alle har en mulighed for at få taget evt. ”luft af ballonen” eller måske rose øvrige gruppemedlemmer.

3.4 Referat
I den ovenfor viste indkaldelse til et vejledermøde er udpeget hvem der skal være referent. Referenten skriver hurtigst muligt et referat efter vejledermødet. Referatet udsendes til samtlige projektdeltagere, inkl. vejlederen.

En skabelon for et referat til et vejledermøde kan se således ud:

Referat fra vejledermøde # nn
Dato:
Tid:
Sted:
Fremmødte:
Udeblevet med afbud:

Udeblevet uden afbud:
Dagsorden

1. Valg af mødeleder
2. Valg af referent
3. Godkendelse af referat fra forrige møde
4. Opfølgning på aktionspunkter fra forrige møde
5. ”Her kan indføjes ekstra punkter”
6. Gennemgang af tidsplan
7. Nye aktionspunkter til næste møde. Hvem gør hvad.
8. Tidspunkt for næste møde
9. Evt.

ad 1)
ad 2)

4. Den komplette projektdokumentation

Projektet dokumenteres i en sammenhængende projektrapport i PDF-format med fortløbende sidenumre.
Filen skal indeholde følgende:

- Forside (inkl. titel, gruppenummer, studienummer og navn for hvert gruppemedlem, navn på institution, navn på vejleder, dato for aflevering, evt. illustration)
- Indholdsfortegnelse (skal angive overskrifter og sidenumre på samtlige afsnit)
- Problemformulering
- Kravspecifikation
- Arkitektur (HW/SW)
- HW-designdokument
- SW-designdokument
- HW-modultestdokument
- SW-modultestdokument
- Integrationstestdokument
- Accepttest inkl. resultater fra accepttesten
- Konklusion på projektet
- Individuel konklusion fra hver deltager i projektgruppen

Dokumentet "1. semesterprojekt, projektoplæg" [1] kan eventuelt anvendes som skabelon for projektrapporten.

Det skal tydeligt angives, hvem der har bidraget til de forskellige dele af projektet, herunder hvem der har skrevet de forskellige dele af projektrapporten.

Den individuelle konklusion, sammen med det, som den enkelte studerende har udviklet og beskrevet i projektrapporten, er grundlaget for at bestå *semesterprojekt 1.*

Omfanget af projektrapporten kan passende være 40 normalsider tekst. En normalside består af 2400 tegn med mellemrum, dvs. projektrapporten kan passende fylde 96.000 tegn, inkl. mellemrum.
Figurer, billeder og lignende tæller heller ikke med i omfanget.

Øvrige bilag som mødeindkaldelser, mødereferater, datablade, source code etc. pakkes sammen i én ZIP-fil.

Projektrapporten (PDF) og samtlige bilag (ZIP) uploades på:
www.eksamen.au.dk
Generel vejledning om digital eksamen findes på:
studerende.au.dk/studier/fagportaler/ece/eksamen/digital-eksamen

Referencer

[1] "1. semester-projekt, projektoplæg", Henning Hargaard, Institut for Elektro- og Computerteknologi, Aarhus Universitet