AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

Vejledning til gennemførelse af projekt 1

Torben Gregersen
tg@iha.dk
12.2.2015

Contents

Indledning

Projekt 1 tager sit udgangspunkt i en opgaveformulering, som indeholder en beskrivelse af en elektrisk bil, der i en konkurrence skal gennemføre en bane med forhindringer på kortest mulig tid.

Opgaveformuleringen for bilprojektet er beskrevet i dokumentet ”Pokaljagten”.

Bilprojektet skal opfattes som et lille udviklingsprojekt, som er fælles for E-, EP- og IKT-studerende på diplomingeniøruddannelsens 1. semester på Aarhus University School of Engineering (ASE).

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

Denne notes mål er at beskrive metoden for hvordan et projektarbejde, f.eks. bilprojektet, kan organiseres og udføres i praksis. Denne metode 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 *projekt 1* og processen for gennemførsel af senere projekter er blot, at detaljeringsgraden 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 præcis 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, sorger, 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 vore 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 rimeligt overblik over projektet. Det er dog altid gældende at lyst og evner til teoretisk arbejde, laboratoriearbejde, programmeringsarbejde, at holde orden i filer og papirer, 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 vore samarbejdspartnere, skal vi øve os i at kunne samarbejde med "kedelige" personer.

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.
Et andet gruppemedlem kan være procesleder. Procesleder en har fokus på at *processen* forløber optimalt, så de aftalte mål for projektet opnås.
Rollerne som hhv. projektleder og procesleder kan 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 vedledning 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 samme ambitionsniveau
- 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
- Ingen tager referat fra møderne
- Der anvendes ikke mødeindkaldelser med dagsorden
- 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.

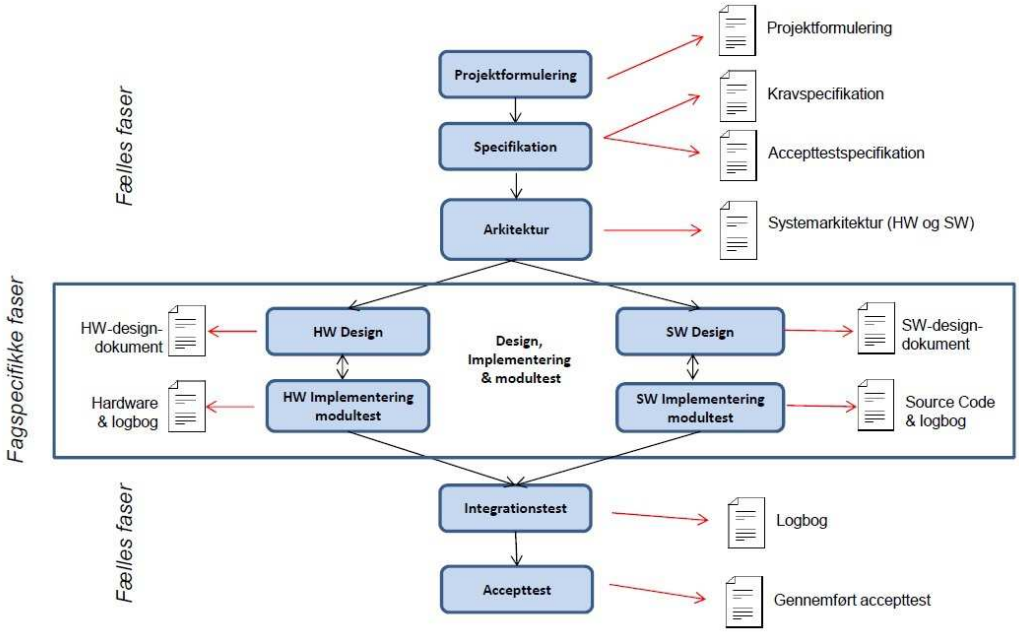
Det kan yderligere anbefales at nedfælde og underskrive en tilsvarende samarbejdsaftale mellem gruppen og vejlederen.

2. Projektgennemførelse og udfærdigelse af projektdokumentation

Arbejdsmæssigt udføres et projekt, som både indeholder udvikling af hardware og software, 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):



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)
- Systemarkitektur (dokument)
- HW-design (dokument)
  SW-design (dokument)
- Implementering af hvert enkelt HW-modul (veroboard, PCB etc.), som derefter gennemgår modultest (noter i logbog på 1. semester)
- Implementering af hvert enkelt SW-modul (source code -> eksekverbar kode), som derefter gennemgår modultest (noter i logbog på 1. semester)

- Integrationstest (noter i logbog på 1. semester)
- Accepttestspecifikation (dokument, som allerede er skrevet i kravspecifikations-fasen - nu indsættes de opnåede testresultater fra den gennemførte accepttest)

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

2.1 Problemformulerings-fase
(output/artefakt: problemformulering (dokument))

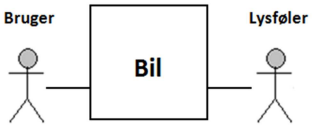
I problemformulerings-fasen beskrives overordnet hvad projektet går ud på. I en "live"-situation svarer problemformuleringen til det, der kommer ud af de første henvendelser fra en potentiel kunde til et HW/SW-udviklingsfirma. Henvendelserne kan ske i form af mails, telefonsamtaler, møder etc., som beskriver et ønske fra kunden. Hvis udviklingsfirmaet vurderer at det er muligt at løse opgaven kan kravene efterfølgende beskrives nøje i et dokument: *problemformulering*.
I *projekt 1* er problemformuleringen på forhånd stillet fra undervisernes side, dvs. "henvendelsen" til jer er allerede sket ved opstart af *projekt 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.

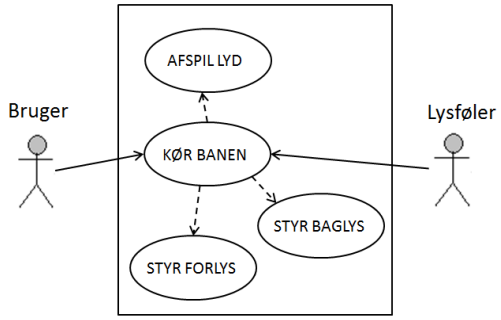
I specifikations-fasen afklares alle forhold, der vedrører rekvirenten af projektet, så udviklingsfirmaet og kunden er enige om hvad der skal udvikles. Kravene skal alle være formuleret, så de er mulige at teste.

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. Præsentationen af aktører og system vises i et *Aktør-kontekst diagram*:



De enkelte aktørers roller beskrives efterfølgende for at uddybe figuren (se dokumentet "Pokaljagten").

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:



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 ”Pokaljagten”).
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.

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 nøjagtig samme måde senere, 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 ”Pokaljagten”).
Når accepttesten skrives kan man samtidigt verificere at kravene, såvel funktionelle som ikke funktionelle krav, er beskrevet tilstrækkeligt tydeligt. Alle krav skal være testbare!

2.3 Arkitektur-fase
(output/artefakt: systemarkitektur (dokument))

Denne fase har overodnet 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.

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

SW nedbrydes ligeledes i bestanddele, som kan kaldes blokke, moduler eller pakker. Hvis der skal anvendes en objektbaseret eller objektorienteret SW-arkitektur benævnes blokkene/modulerne som pakker, der hver indeholder et antal klasser, der har en logisk sammenhæng i pakken. I meget små systemer, f.eks. i bilprojektet, er pakkerne identiske med klasser, dvs. der fokuseres kun på klasser i systemarkitekturen i projekt 1.

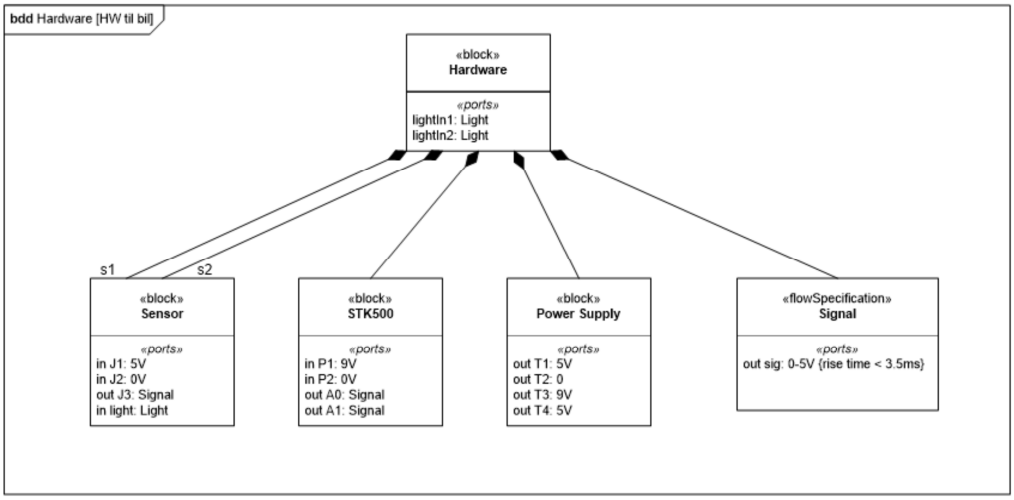
Det er muligt at udføre nedbrydningen uden i detaljer at vide hvordan den enkelte blok/modul/pakke/klasse skal udvikles. Der er kun brug for funktions-beskrivelse og en beskrivelse af samtlige væsentlige grænseflader mellem projektets forskellige bestanddele. Dette giver frihed til på et senere tidspunkt at vælge en ny teknologi til at udvikle de samme blokke/moduler/pakker/klasser.

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

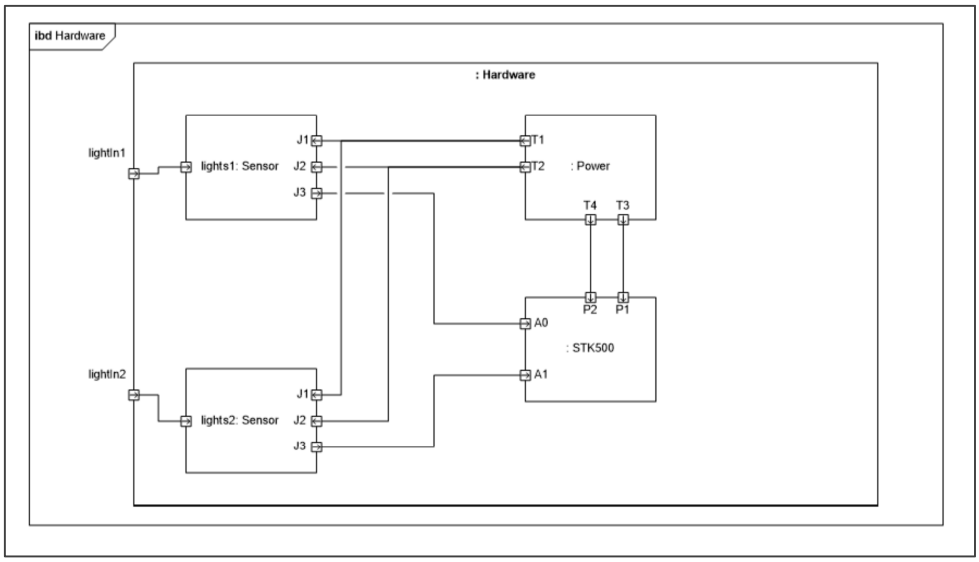
Overblikket over hhv. HW-arkitektur og objekt-baseret SW-arkitektur kan skabes ved at udarbejde hhv. SysML-diagrammer til HW-arkitekturen og UML-diagrammer til SW-arkitekturen.

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. Del-blokkene navngives og signalerne, der optræder på de enkelte porte i hver blok, navngives.



I det BDD, der blev vist ovenfor, kan man ikke vise hvordan de enkelte HW-signaler forbindes mellem blokkene. Det er nødvendigt at vise disse forbindelser. Forbindelserne kan illustreres vha. et IBD (Internal Block Diagram). Portene er forinden navngivet (f.eks. J1, T1) i BDD. Det er kun forbindelserne, der er tilføjet i IBD.



IBD 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 kan 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 detaljegrad afhængigt af det aktuelle problem. Det kan være praktisk at have blokbeskrivelse og blokdiagram (BDD) samlet.

Blok-navn	Funktionsbeskrivelse	Signaler	Kommentar
Sensor	Detekterer lys	5V	Strømforsyning

		0V	Reference
		Signal	Udgangssignal
		Light	Lys
STK500	Processerer input fra Sensor	9V	Strømforsyning
		0V	Reference
		Signal	Lys
		Signal	Lys
Power	Forsyner STK500 og Sensor med spænding	5V	Strømforsyning
		0V	Reference
		9V	Strømforsyning
		5V	Strømforsyning

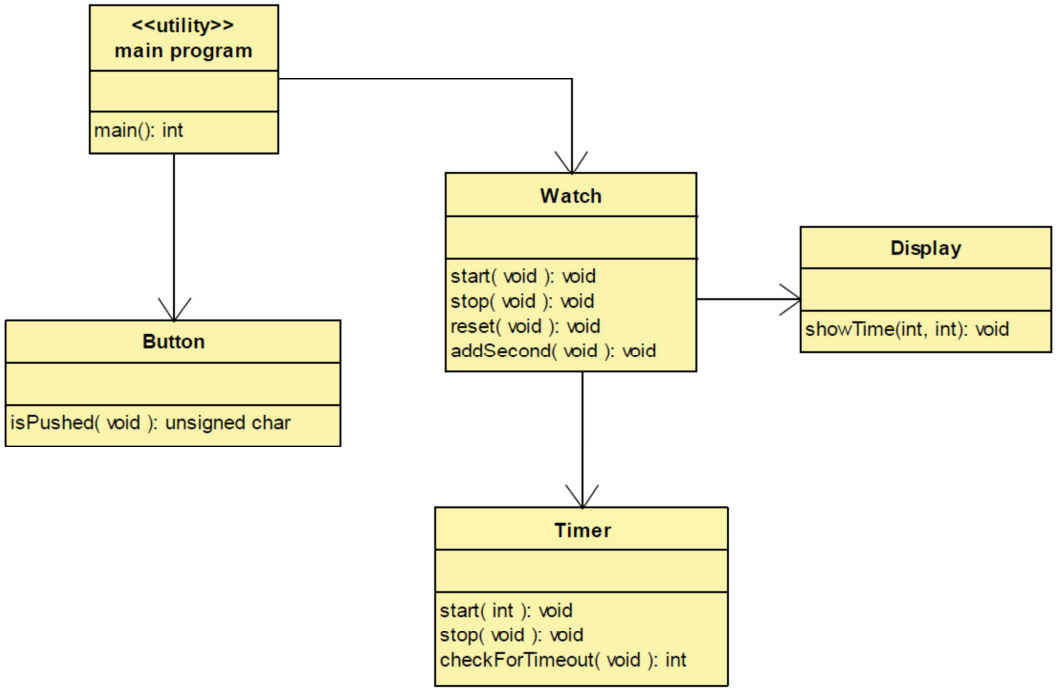
For at fuldende grænsefladebeskrivelsen skal signalerne behandles detaljeret.

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

Signal-navn	Funktion	Område	Port 1	Port 2	Kommentar
0V	Reference til analoge spændinger		Power, T1 Power, T1 Power, T3	Sensor1, J1 Sensor2, J1 STK500, P1	Stel
5V	Forsyningsspænding	4,9-5,1V	Power, T4	STK500, P2	
9V	Forsyningsspænding	8,9-9,1V	Power, T2 Power, T2	Sensor1, J2 Sensor2, J2	
Light	Fysisk lys				
Signal	Indikerer registrering af fysisk lys	0-5V	Sensor1, J3 Sensor2, J3	STK500, A0 STK500, A1	Rise time < 3.5ms

SW-arkitekturen fastlægges ved at finde de navneord, som optræder i kravspecifikationen. Disse analyseres for at beslutte om disse navneord kan være kandidater til at optræde som klasser i systemet. Følgende eksempel viser et objektbaseret design for et stopur, hvor klassens navn og navnet på hver enkelt metode (funktion, operation) er angivet. Klassernes indbyrdes anvendelse er illustreret med pile. En pil rettet mod en klasse indikerer at der anvendes en eller flere metoder i denne klasse.

I stil med SysML's BDD og IBD kan illustrationen af klassediagrammet ikke stå alene. De enkelte klasser skal beskrives med en detaljeringsgrad, der gør det muligt for andre SW-udviklere at udføre et detaljeret design på klasserne.



Hver klasse (undtagen <<utility>> klassen) repræsenterer en header- *og* en source-fil. Ovenstående eksempel repræsenterer altså 4 header-filer og 5 source-filer (den 5'te er den source-fil, der indeholder main()).

Header-filerne indeholder funktionernes (metodernes/operationernes) prototyper og source-filerne indeholder funktionernes implementering (source code).

Klassebeskrivelser

Timer

Ansvar: Klassens ansvar er at styre tiden et millisekund ad gangen

Metoder:
void start(int milliseconds)
Parametre: Den ønskede tid i millisekunder
Returværdi: Ingen
Beskrivelse: Funktionen skal starte optællingen af millisekunder
void stop(void)
Parametre: Ingen
Returværdi: Ingen
Beskrivelse: Funktionen skal stoppe optællingen af millisekunder
int checkForTimeout(void)
Parametre: Ingen
Returværdi: True (1) hvis den ønskede tid er gået – false (0) hvis den ønskede tid ikke er gået
Beskrivelse: Funktionen skal tjekke om den ønskede tid er gået

Ur
Ansvar: Bla bla bla
Metoder:
void start(void)
Parametre: ingen
Osv. osv.

Beskrivelsen af hver enkelt funktion skal være så præcis, at der *ingen* tvivl er om, hvad den skal gøre, således at en anden person kan designe/implementere den direkte ud fra beskrivelsen. Meget simple funktioner beskrives eventuelt ikke.

Efter afsluttet HW/SW-systemarkitektur kan de enkelte HW-blokke/moduler og SW-pakker/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)

På basis af 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 systemarkitektur-fasen.

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

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

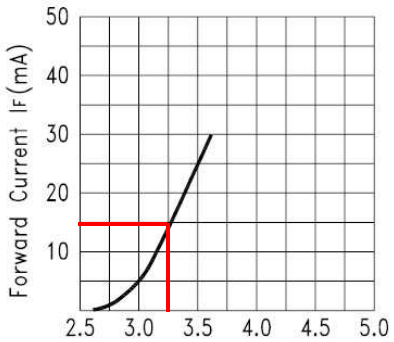
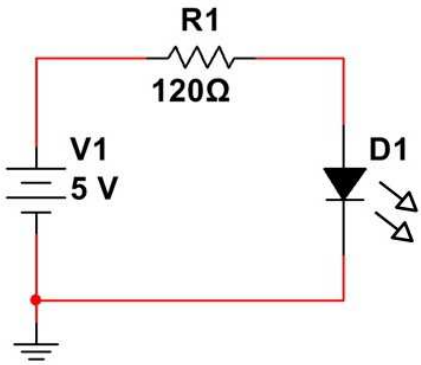


Fig.2 Forward Current vs. Forward Voltage

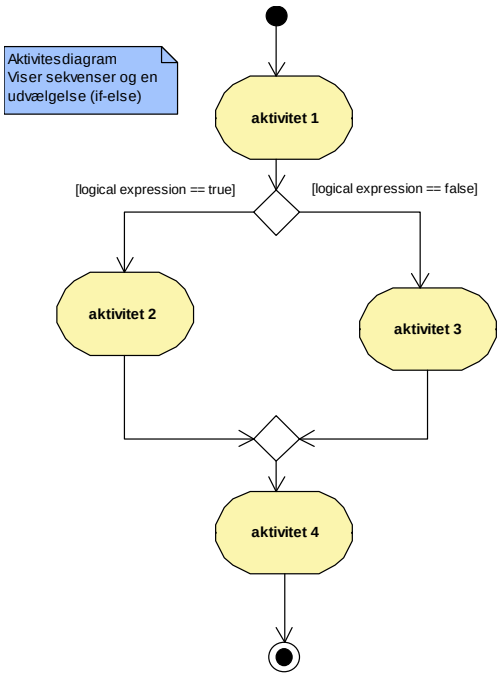
Strømmen gennem lysdioden D1 besluttes til at være 15 mA
Ifølge databladet giver dette et spændingsfald over D1 på 3,25V
Forsyningsspændingen V1 er 5V
Spændingsfaldet over R1 er da: VD1 = 5V – 3,25V = 1,75V
R1 kan derfor beregnes således: R1 = 1,75V / 15mA = 116,7 Ω ~ <u>120 Ω</u>



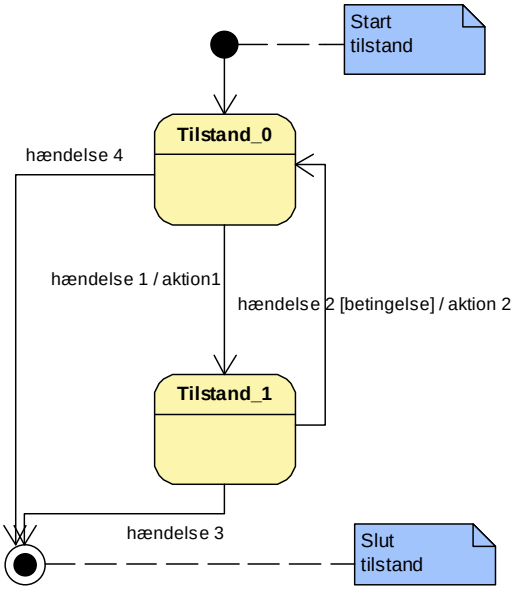
Der udarbejdes styklister for samtlige komponenter, der skal anvendes. Hvis der skal anvendes PCB (Print Circuit Board) i implementeringsfasen skal layout'et til dette PCB beskrives i den detaljerede HW-design fase.
Alt ovenstående HW-design beskrives i et HW-design dokument.

På basis af arkitektur-fasen, hvor grænsefladerne til hver enkelt SW- blok/modul/pakke/klasse blev beskrevet, udføres nu et SW-design, der er så detaljeret at alle væsentlige 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). Alternativt kan pseudokode anvendes i stedet for UML-aktivitetsdiagrammer.
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 eller vha. pseudokode.

UML-Aktivitetsdiagram:



UML-tilstandsdiagram:



I stil med det detaljerede HW-diagram understøttes figurerne af en præcis, teknisk forklaring, ledsaget af de nødvendige beregninger.
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: logbog på 1. semester)

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 etc.
Hver implementeret HW-blok/modul testes omhyggeligt i laboratoriet. Hvis dette ikke fungerer korrekt rettes fejlen. Med mindre der er tale om monteringsfejl eller løse forbindelser, er det typisk nødvendigt at ændre HW-modulets design. Det kan yderligere evt. være nødvendigt at ændre arkitekturen. Er der rettet i HW-design og/eller HW-arkitektur skal de tilhørende dokumenter opdateres.

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

2.6 Integrationstest-fase

(output/artefakt: logbog på 1. semester)

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.

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, og der markeres "OK" eller "Ikke OK" for hvert enkelt punkt i accepttesten.
I *projekt 1* skrives en præcis konklusion på resultaterne fra accepttesten i slutningen af projektdokumentation.

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 retablere udviklingsarbejdet i tilfælde af uheld

3.1 Samarbejdskontrakt

Som nævnt tidligere er det er en fordel hvis 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 ledere, eller går lederrollen på skift?
  Hvilket ansvar og hvilke opgaver har lederen / lederne?

- 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 12 tallet)?
  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 enig om at bruge i gruppen?

- Hvad er konsekvensen for et medlem, der ikke overholder samarbejdskontrakten?
  Skal der være en form for "straf"?
  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									

Det kan være nødvendigt at justere tidsplanen undervejs, og dermed arbejdebelastningen pr. tidsenhed, for at opnå det bedst mulige projektresultat.

3.3 Mødeindkaldelse
En vigtig del af projektadministrationen er at der indkaldes til møde med jævne, gerne 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 efter vejledermødet et referat, som udsendes til samtlige projektdeltagere, incl. 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)

3.5 Logbog

Vigtige hændelser af teknisk og ikke teknisk art, som opstår under projektforløbet, skrives i en logbog. Det er især vigtigt at notere hændelser i implementeringsfasen i forbindelse med udvikling, modultest, integrationstest og fejlfinding af hhv. HW og SW. Tidspunktet for hver enkelt hændelse angives i logbogen.

4 Den komplette projektdokumentation

Projektet dokumenteres dels på papirform, dels på et digitalt medie (upload af komprimeret fil (.zip eller .rar) til projektgruppens Campusnet-side).

De vigtigste dele af projektets dokumentation afleveres på papirform. Denne form muliggør at underviser og censor kan skrive kommentarer i dokumentationen. Disse kommentarer bruges i forbindelse med censur af projektet.

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

For *projekt 1* gælder at projektdokumentationen afsluttes med at skrive:

- En projektkonklusion – som beskrives af hele gruppen
- En individuel konklusion – som skrives af hver enkelt studerende

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

I de efterfølgende semestre skrives resultatbehandling og konklusion i en separat *projektrapport*, hvor der i alt skal afleveres 2 dokumenter: en *projektrapport* og en *projektdokumentation*.

I *projekt 1* afleveres kun et, samlet dokument.

Projektdokumentationen består af følgende:

- Forside
- Problemformulering
- Indholdsfortegnelse
- Kravspecifikation
- Systemarkitektur (HWog SW)
- HW-designdokument
- SW-designdokument
- Accepttest med testresultater
- Projektkonklusion, skrevet af gruppen
- Individuel konklusion, skrevet af hver enkelt studerende i gruppen

4.1 Forside

Projektdokumentationens forside skal indeholde følgende informationer:

- Projektets titel og evt. undertitel
- Gruppenummer
- Projektdeltagernes studienumre, navne og underskrifter
- Navn på institution
- Dato for aflevering
- Navn på vejleder
- Evt. illustration

4.2 Indholdsfortegnelse

Skal angive overskrifter og sidenumre på de forskellige afsnit.

4.3 Opgaveformulering

Her indsættes den udleverede opgaveformulering.

4.4 Kravspecifikation

Her indsættes den fuldstændige kravspecifikation.

4.5 Systemarkitektur (HW og SW)
Her beskrives systemets overordnede arkitektur.

4.5 Design (HW/SW)
Her beskrives det detaljerede design for hhv. HW og SW.

4.6 Accepttest
Her beskrives den gennemførte accepttest. Testresultater af de enkelte dele af testen (OK/ ikke OK) angives præcist.

4.7 Bilag (anbringes kun på et digitalt medie i komprimeret form (.zip eller .rar))
Her indsættes alle relevante bilag til projektet.

- Ovennævnte papir-dokumenter på elekronisk form (Word-filer, PDF-filer …)
- Datablade
- Litteraturliste
- Kode (”source code”)
- Brugermanual
- Samarbejdsaftale
- Tidsplan
- Mødeindkaldelser
- Mødereferater
- Logbog
- Komponentliste
- Printudlæg
- Fotos, film etc.
- Slides fra præsentation (.ppt, .pdf)