

1.SEMESTER-PROJEKT

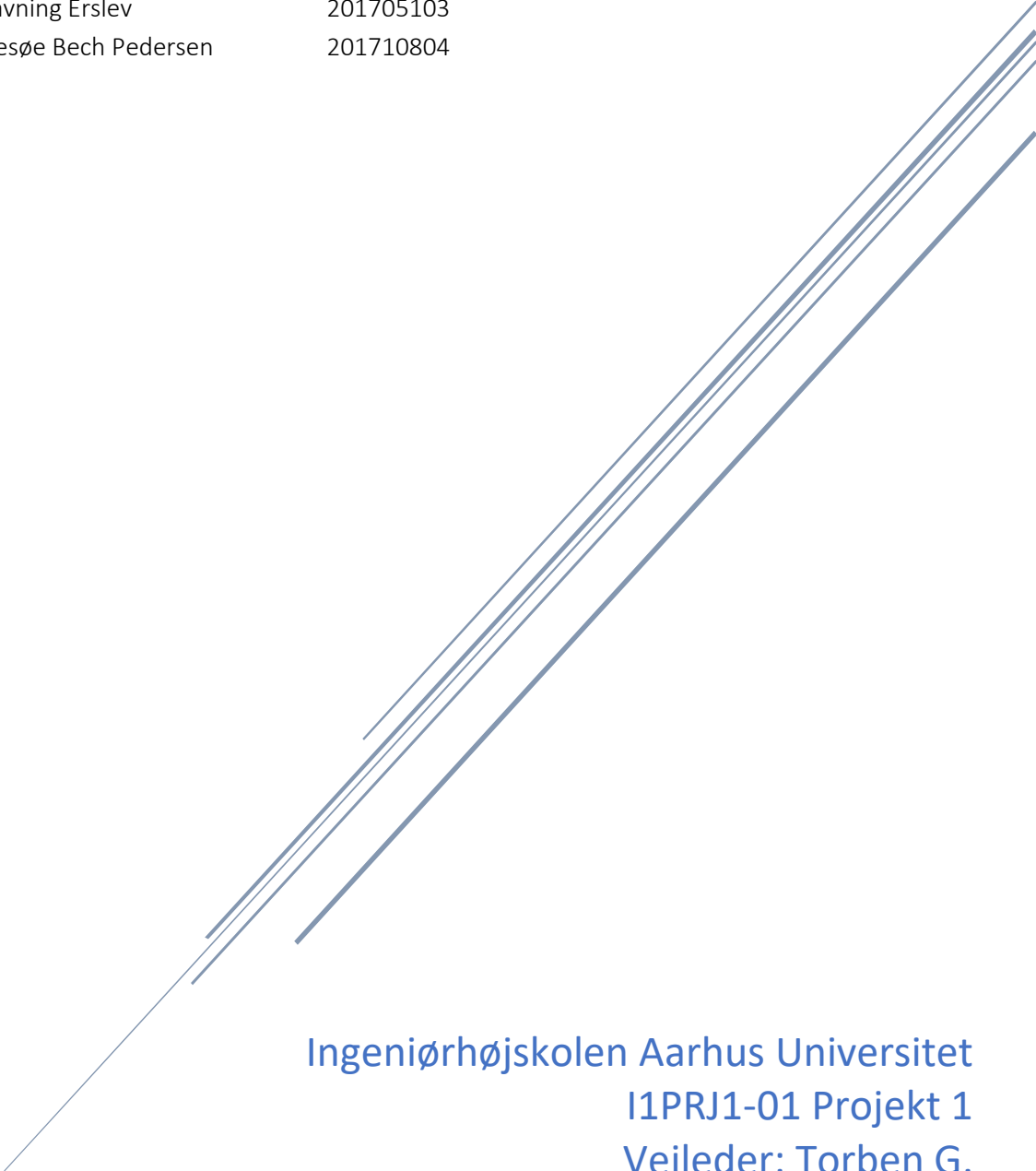
Gruppe Nr.: IKT1

Gruppemedlemmer:

Gustav Nørgaard Knudsen
Camilla Andreassen Holmstoel
Rasmus Møller Nielsen
Shynthavi Prithviraj
Simon Phi Dang
Andreas Stavning Erslev
Anders Kallesøe Bech Pedersen

Studie Nr.:

201807736
201900239
201909856
201807198
201705957
201705103
201710804



Ingeniørhøjskolen Aarhus Universitet
I1PRJ1-01 Projekt 1
Vejleder: Torben G.

16. januar 2020

Indholdsfortegnelse

Arbejdsfordeling	4
Indledning	4
Problemformulering.....	4
Kravspecifikation.....	9
Aktør-kontekst diagram	9
Funktionelle krav.....	9
Use case 1: "Kør banen"	10
Use case 2: "Afspil lyd"	10
Use case 3: "Styr forlys"	10
Use case 4: "Styr baglys"	11
Ikke-funktionelle krav	11
Hardware	13
Hardware arkitektur (Alle)	13
Hardware design	17
Motorhardware (Gustav)	17
Lyshardware (Simon og Shyn)	21
SOMO-II (Rasmus & Camilla).....	27
Software	29
Software arkitektur (Alle).....	29
Software design	30
Motor software (Gustav)	30
Lys software (Simon og Shyn)	33
Somo software (Rasmus & Camilla)	35
Sensor software (Andreas & Anders).....	38
Accepttest	42
Funktionelle krav.....	42
Ikke-funktionelle krav	43
Konklusion.....	46
Hardware konklusion	46
Software konklusion	46
Overordnet konklusion	46
Individuelle konklusioner	46
Rasmus	46
Gustav	47
Shynthavi.....	47

Simon	47
Anders	47
Andreas	48
Camilla.....	48
Billagsoversigt	49
Litteraturliste	49

Arbejdsfordeling

	Anders	Andreas	Camilla	Gustav	Simon	Shynthavi	Rasmus
Motor				✓			
Sensor	✓	✓					
Lys					✓	✓	
Lyd			✓				✓
Mødeleder	✓						
Referant					✓		
Planlægger			✓				

Tabel 1 - Arbejdsfordeling

Indledning

I dette 1. semester projekt er formålet at lave en bil, der kan køre gennem en opstillet bane med forskellige forhindringer, som bilen skal kunne reagere på. De 4 hovedkomponenter, som skal implementeres og som endegyldigt skal snakke sammen under gennemkørslen er lys, lyd, sensorer og motoren. Bilen opbygges med forskellige hardware-komponenter og software-kode, der samlet skal få bilen til at køre banen, hvor den skal reagere på refleksbrikker, afspille lyd, lyse på forskellige niveauer under kørslen alt i mens bilen skal kunne bakke og hastighedsstyres. Til gennemførslen af projektet er der lavet HW- og SW-arkitektur samt design af disse. Dette giver et overblik over den komplekse bil og, hvordan diverse komponenter er sammensat. Hertil er der lavet kravspecifikationer, der beskriver, hvilke krav der stilles til bilen samt en accepttest, hvor det testes om bilen endegyldigt lever op til de stillede krav. Endvidere er der et 5. komponent til bilen, nemlig individualiseringen af bilen i form af "flair". "Flair" for denne bil er i form af temaet, "Frozen", hvor bilen er udsmykket med tema-relevante frozen-komponenter, som en af hovedpersonerne fra filmen, Frozen, nemlig Elsa, samt lyde fra filmen.

Problemformulering

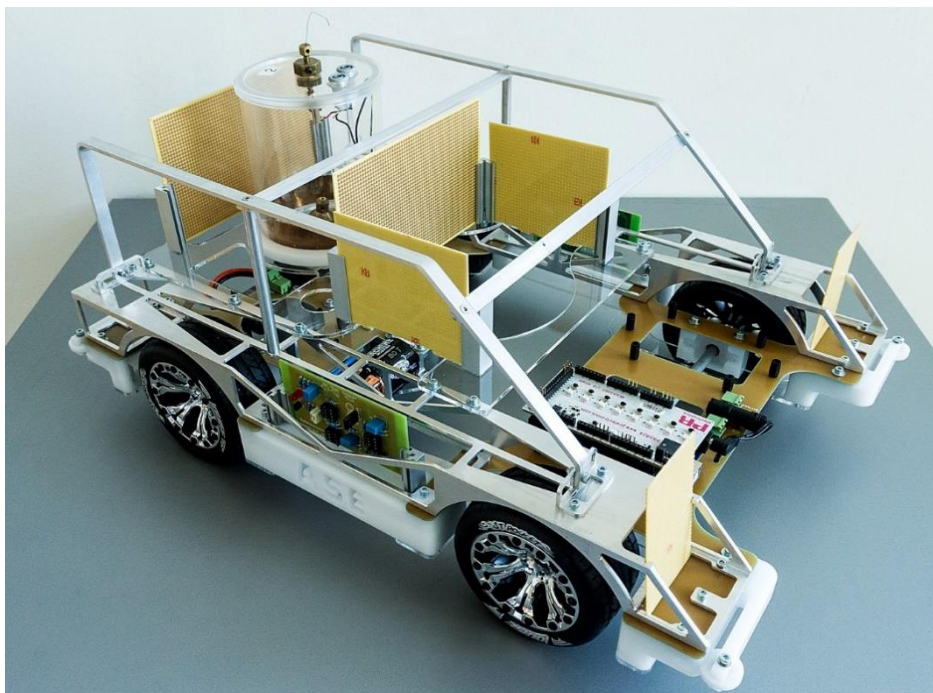
Det tværfaglige projekt PRJ1, der gennemføres på 1. semester for E-, IKT- og EE-studierne, drejer sig om udvikling af hardware og software til styring af en elektrisk bil.

Som en indledning til projektet gennemføres 4 øvelser i E-LAB. Disse kan betragtes som en forundersøgelse eller analyse af de teknikker, der tænkes anvendt under projektet.

Projektet skal udarbejdes efter retningslinjerne i dokumentet "Vejledning til gennemførelse af projekt 1".

Under projektets forskellige faser gives der enkelte forelæsninger, der uddyber indholdet af disse faser.

Et bilkarrosseri med påmonteret motor bliver udleveret til hver enkelt projektgruppe. Se figur 1.



Figur 1 Karrosseri med printplader, pendul og Arduino Mega2560

Den færdige bils formål er, at den skal kunne gennemkøre en konkurrencebane efter regler, der er nærmere beskrevet i bilag 1, "Konkurrencen".

Bilen skal udstyres med:

- Mikrocontroller.
- Elektronik til regulering af motor.
- Sensorer til detektering af refleksbrikker.
- For- og baglygter.
- Elektronik til styring af for- og baglygter.
- Eventuelt elektronik og højttaler til afspilning af lyd.

Nogle af de nødvendige hardwareenheder vil helt eller delvis blive udviklet i andre kurser på 1. semester.

Det drejer sig om:

- Print med kredsløb til detektering af refleksbrikker.
Dette print skal forbindes til bilens mikrocontroller.
Hvis der foretages modifikationer af printet for at tilpasse det projektet, skal disse modifikationer beskrives i projektdokumentationen.
- "Fejltæller": Print med kredsløb til tælling og visning af pulser fra et pendul.
Pendulet anvendes kun som et påmonteret måleudstyr under konkurrencen, og den er ikke en del af selve projektet.

Bilen skal forsynes med fremadrettet kørellys (hvidt) og baglys (rødt), samt bremselys (kraftigt rødt).
Se figur 2 og figur 3.

Bremselys og baglys udsendes fra samme kilde og skal kunne reguleres i to forskellige styrker.

Hvert enkelt for-, bag- eller bremselys kan realiseres som et antal lysdioder, der opbygges som et LED-sæt.

Hvert sæt kan opbygges som en kombination af serie- og/eller parallel-forbundne lysdioder.

I projektet anvendes microcontrolleren Mega2560 på et "Arduino Mega2560" kit. Denne anvendes til:

- Styring af bilens fart og retning.
- Styring af for- og baglygternes lysstyrke.
- Detektering af banens refleksbrikker.
- Afspilning af lyde og/eller melodier.

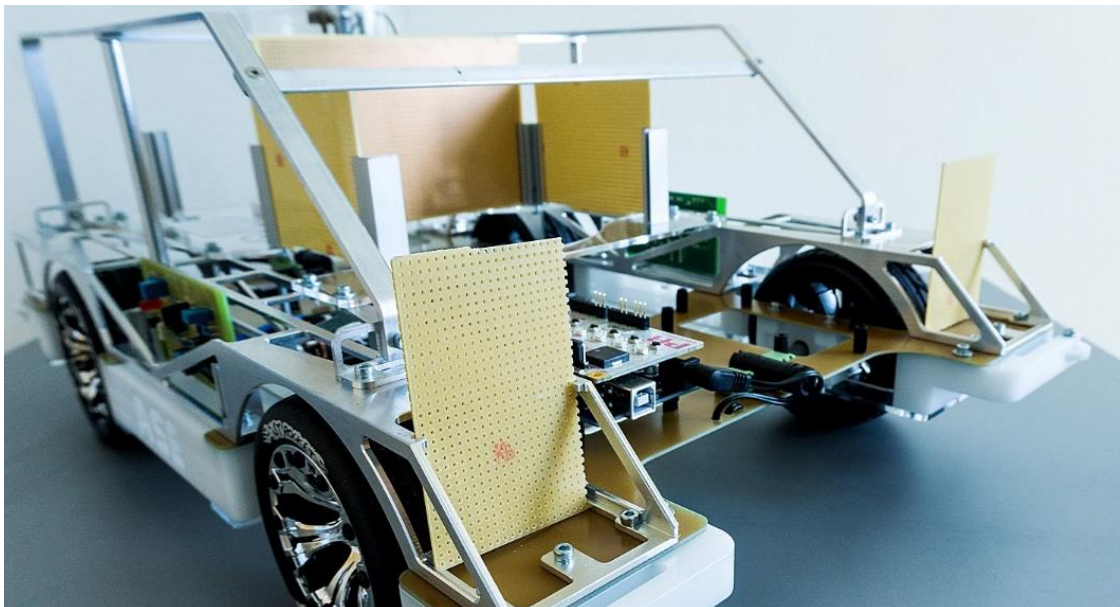
Bilen skal under konkurrencen køre på en bane med to baner, hvor banerne vil sikre, at bilen bliver på banen. Banerne er udstyret med 7 par refleksbrikker, der kan detekteres af optiske sensorer placeret på bilen.

Ved start skal bilen afspille en specifik startlyd eller startmelodi, og efter passage af mållinjen skal afspilles en specifik slutlyd eller slutmelodi. Ved passage af hver enkelt af refleksbrikkerne skal bilen afgive en specifik "refleksbriklyd".

På figur 4 er vist placering af et af printene til detektering af refleksbrikker (der skal være et på hver side).

Figur 5 viser en mulig placering af print med anden elektronik, som skal konstrueres for at kunne styre bilen.

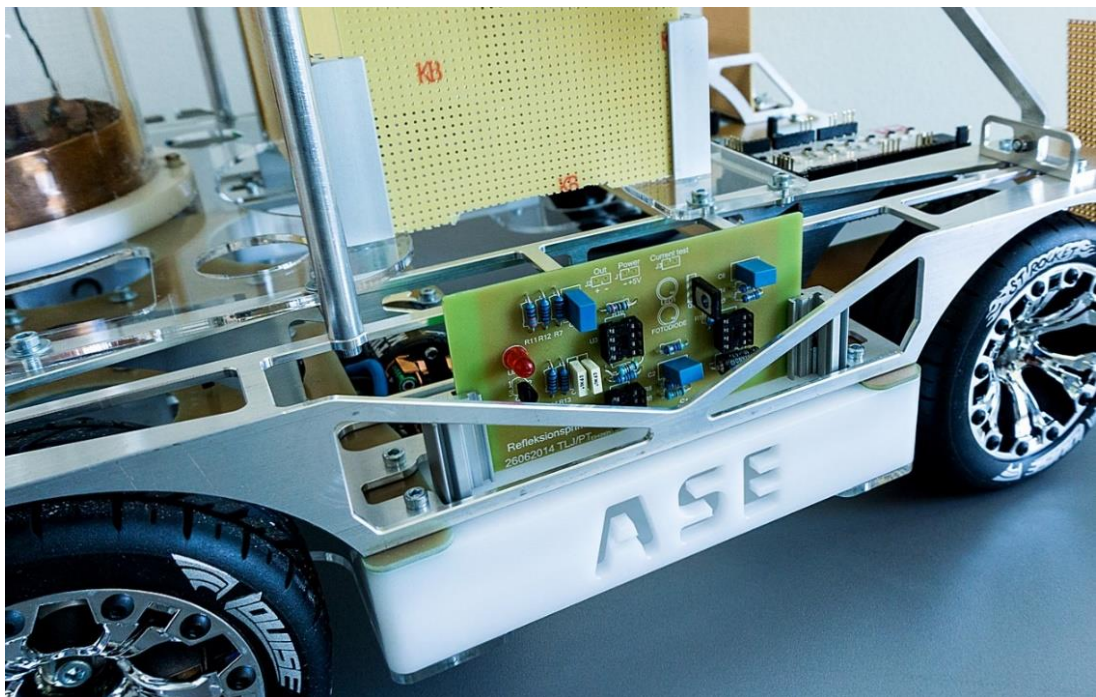
De præcise fysiske mål for bilkarrosseriet kan ses i bilag 1.



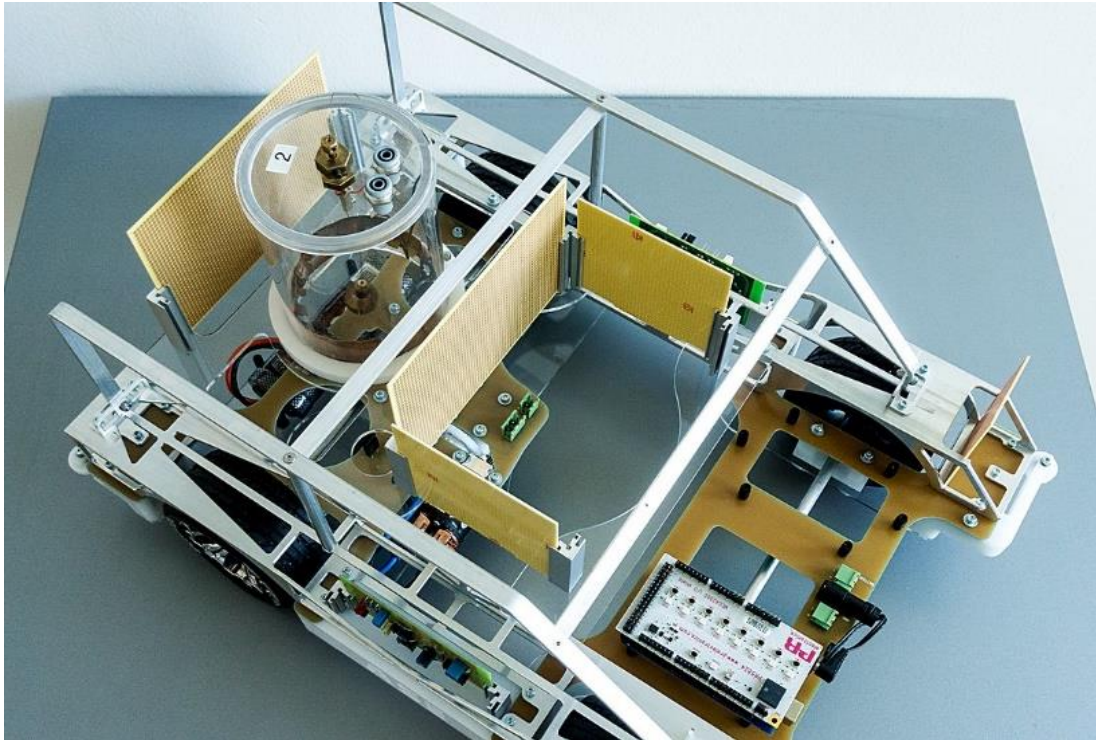
Figur 2 Placering af to print til forlys, højre og venstre side



Figur 3 Placering af print til bag- og bremselys



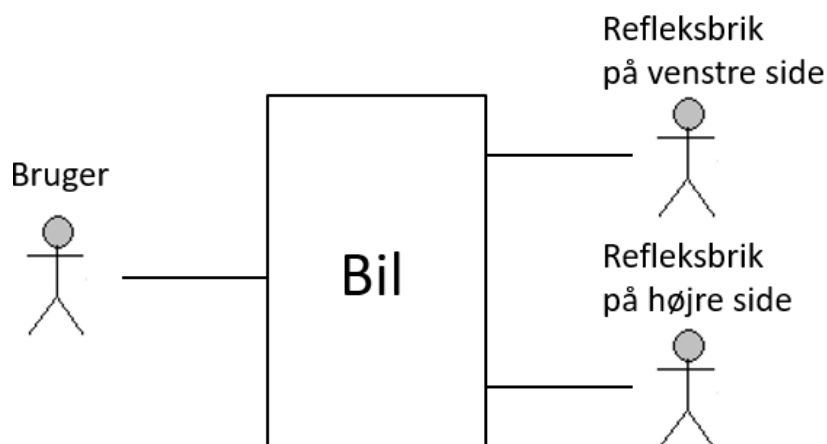
Figur 4 Placering af print til detektering af refleksbrikker



Figur 5 Print med øvrige elektronik placeret midt i bilen

Kravspecifikation

Aktør-kontekst diagram



Figur 6 Aktør-kontekst diagram

Det system, som skal udvikles, er selve den elektriske bil.

Aktøren "Bruger" kan få bilen til at starte et gennemløb af banen.

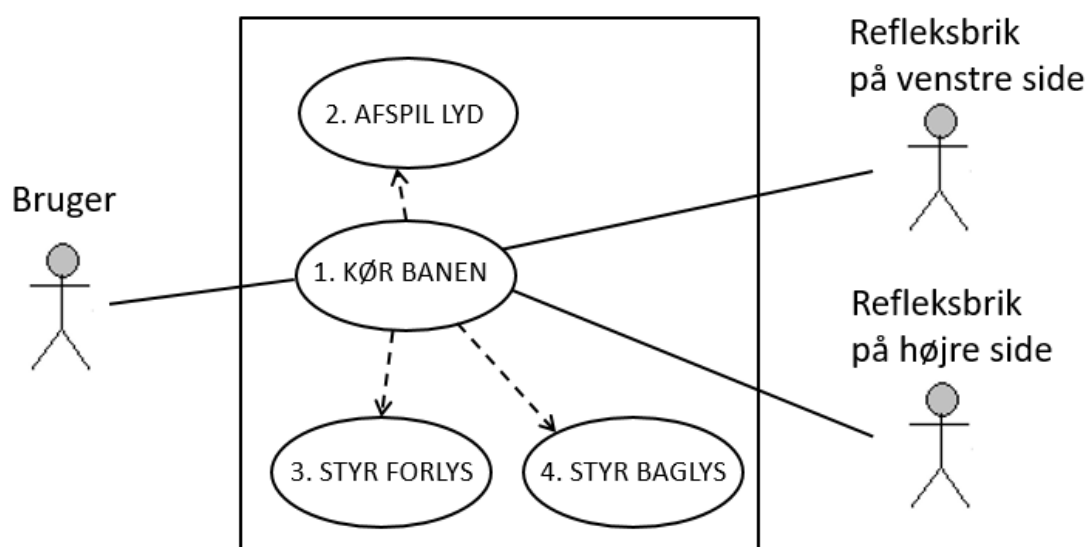
Brugeren er en primær aktør.

Aktøren "Refleksbrik på venstre side" kan detekteres af bilen, hver gang en sådan passeres.

Tilsvarende gælder for aktøren "Refleksbrik på højre side".

Refleksbrikkerne er sekundære aktører.

Funktionelle krav



Figur 7 Use case diagram

Use case 1: "Kør banen"

Mål

Denne use case beskriver, hvordan banen skal gennemkøres.

Initieres af: Bruger.

Normalt scenarie

1. Brugeren tænder for bilen og placerer den, så den kører forlæns ind på banen ved at passere startlinjen.
2. Når bilen har passeret refleksbrik nummer 6, bringes bilen til standsning, inden refleksbrik nummer 7 nås.
3. Bilen bakker, indtil refleksbrik nummer 5 er passeret, og bringes til standsning, inden refleksbrik nummer 4 nås.
4. Bilen kører forlæns, indtil refleksbrik nummer 7 nås.
5. Bilens bringes til standsning i målområdet, der er defineret som området mellem banens slut-ende og en meter efter banens slut-ende. Ved standsning skal hele bilen være placeret i målområdet.

Generelt gælder:

- Lyd afspilles som beskrevet i use case 2 "Afspil lyd".
- Forlyset styres som beskrevet i use case 3 "Styr forlys".
- Baglyset styres som beskrevet i use case 4 "Styr baglys".

Use case 2: "Afspil lyd"

Mål

Denne use case beskriver styring af en i systemet indbygget lyd giver.

Initieres af: Use case 1 "Kør banen".

Normalt scenarie

1. Når bilen tændes for at køre ind på banen, afspilles en specifik "startlyd" eller "startmelodi".
2. Hver gang en refleksbrik passeres, afspilles en specifik "refleksbriklyd".
3. Når refleksbrik nummer 7 passeres, afspilles en specifik "slutlyd" eller "slutmelodi".

Use case 3: "Styr forlys"

Mål

Denne use case beskriver styringen af bilens indbyggede forlys.

Forlyset kan være slukket eller tændt.

Initieres af: Use case 1 "Kør banen".

Normalt scenarie

Forlyset skal være tændt, når bilens motor påtrykkes en spænding, for at bringe den til at køre. Ellers skal forlyset være slukket.

Use case 4: "Styr baglys"

Mål

Denne use case beskriver styringen af bilens indbyggede baglys.

Baglyset kan være slukket, lyse med mellemstyrke ("almindeligt baglys") eller lyse med kraftig styrke ("bremselys").

Initieres af: Use case 1 "Kør banen".

Normalt scenarie

* Baglyset skal lyse med mellemstyrke ("almindeligt baglys"), når bilens motor påtrykkes en spænding, for at bringe den til at køre. Ellers skal baglyset være slukket.

* Baglyset skal lyse med kraftig styrke ("bremselys"), mens spændingens til bilens motor mindskes. Bremselyset skal herefter forblive tændt i 0,5 sekund +/- 0,1 sekund.

Use case 5: "Kørsel"

Mål

Denne use case beskriver styringen af bilen vha. Arduino mega2560 shield'et.

Når bilen tændes, går den i "tomgang" og venter på at brugeren påvirker en knap på Arduino shield'et. Under kørsel kan brugeren, ved påvirkning af én af to mulige knapper, enten "slukke" bilen eller sætte den i "tomgang". Hvis bilen "slukkes", skal Arduinoen reset'es før bilen kan køre igen. Sættes bilen i "tomgang", vil bilen ved gentaget påvirkning af knappen starte kørslen forfra. Køres banen færdigt vil bilen automatisk gå i "tomgang" efter endt kørsel.

Normalt scenarie

Når bilen tændes, går den i "tomgang" og afventer aktivering fra brugeren. Når bilen aktiveres, køres banen i henhold til use case 1. Efter endt kørsel går bilen igen i "tomgang".

Bilen "slukkes" ikke i et normalt scenarie.

Ikke-funktionelle krav

1. Generelle krav

- 1.1. Bilen skal styres, så den kan ændre sin hastighed under gennemkørsel af banen.
- 1.2. Bilen skal på en enkelt opladning af dennes batterier kunne gennemføre mindst 5 gennemkørsler af banen.
- 1.3. På bilens højre og venstre side skal placeres detektorer, der kan registrere en R80 refleksbrik i afstanden 2 cm til 25 cm. Se figur 5 og figur 6.
- 1.4. Bilen monteret med al udstyr må maksimalt veje 5 kg.
- 1.5. Bilens maksimale højde skal være 41 cm.

2. Bilens forlys

- 2.1. Implementeres med 2 hvide LED-sæt, der monteres med et sæt i henholdsvis højre og venstre side.
Se figur 5.
- 2.2. Når forlyset er tændt, skal hvert LED-sæt lyse svarende til én LED med middelstrømmen 50 mA
+/- 5 mA.

3. Bilens bag- og bremselys

- 3.1. Implementeres med 2 røde LED-sæt, der monteres med et sæt i henholdsvis højre og venstre side.

Se figur 4 og figur 6.

- 3.2. Ved "bremselys" skal hvert LED-sæt lyse svarende til én LED med middelstrømmen 50 mA +/- 5 mA.
- 3.3. Ved "almindeligt baglys" skal hvert LED-sæt lyse svarende til én LED med middelstrømmen 10 mA +/- 2 mA.

4. Bilens lyd

4.1. Startlyd / startmelodi

- 4.1.1. "001Cold.mp3" - 2 sekunder

4.2. Refleksbriklyd

- 4.2.1. 1. lyd - "002Scull.mp3" - 1 sekund
- 4.2.2. 2. lyd - "003Bones.mp3" - 1 sekund

4.3. Slutlyd / slutmelodi

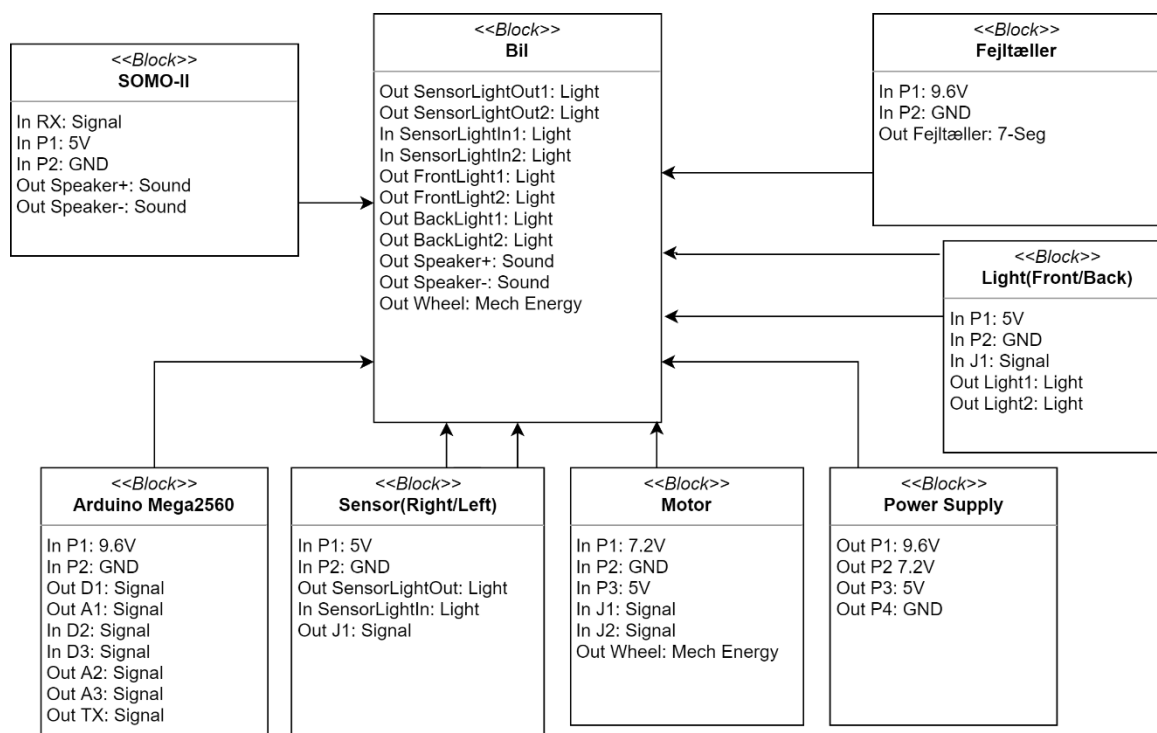
- 4.3.1. "004let-mp3" - 7 sekunder

Hardware

I følgende afsnit vil der blive beskrevet, hvordan den overordnede hardware arkitektur er udtænkt og derefter, hvordan den er udført til de forskellige moduler. Dette inkluderer, hvilke valg der er blevet gjort til valg af komponenter og begrundelsen for det.

Hardware arkitektur (Alle)

Figur 8 er en figur over bilens BDD (Block Definition Diagram). BDD'en er det samlede overblik over de hardwarekomponenter, der anvendes i bilen. Hver blok på diagrammet henviser til diverse hardwaregrupper, hvor der heri vises et overblik over diverse signaler i blokken og, hvordan disse signaler snakker sammen med hovedblokken "Bil".



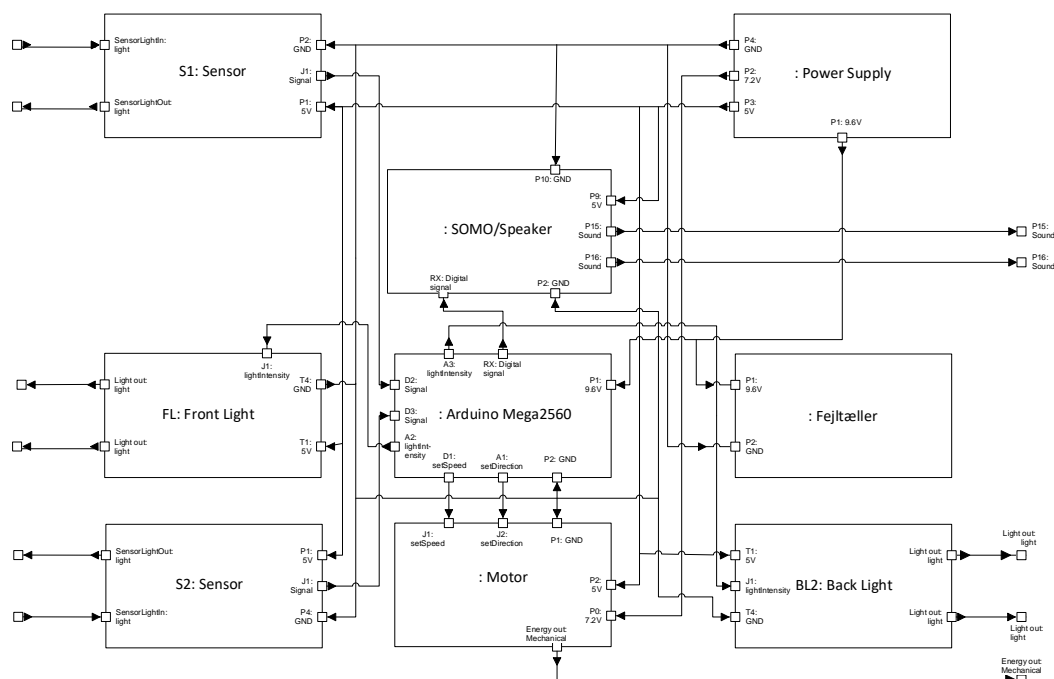
Figur 8 BDD

Tabel 2 er en beskrivelse af BDD'en, Figur 8. Her er et overordnet overblik over, hvad diverse signaler betyder. Her vises, hvilken blok, hvert signal tilhører samt dets funktion, navn, type osv.

Bloknavn	Funktions-beskrivelse	Signalnavn	Signaltype	Kommentar
Sensor	Detekterer lys	P1	5V	Strømforsyning til sensor
		P2	GND	Reference (Ground)
		J1	Signal	Sender signal videre om der er modtaget lys
		SensorlightOut	Light	Fysisk lys ud
		SensorlightIn	Light	Fysisk lys ind
Arduino	Processer for inputs	P1	9.6V	Strømforsyning til Arduino
		P2	GND	Reference (Ground)
		D1	Signal	Motor retning
		D2	Signal	Sensor
		D3	Signal	Sensor

		A2	Signal	LED
		A3	Signal	LED
		A1	Signal	Seriel kommunikation
		TX	Signal	Sender fra UART
Power	Forsyner Arduino Mega2560, sensorer, lys, motor og lyd	P1	9,6V	Strømforsyning
		P2	7,2V	Strømforsyning
		P3	5V	Strømforsyning
		P4	GND	Reference (Ground)
Lys	Afgiver lys	P1	5V	Strømforsyning
		P2	GND	Reference (Ground)
		J1	Signaltype	PWM-signal til LED
		Light1	Light	Forlys LEDsæt
		Light2	Light	Baglys LEDsæt
Motor	Styrer hjul	P1	7,2 V	Strømforsyning
		P2	GND	Reference (Ground)
		P3	5V	Strømforsyning
		J1	Signal	Retning
		J2	Signal	Hastighed
		Wheel	Mekanisk energi	Hjul
SOMO-II	Styrer lyd	P1	5V	Strømforsyning
		P2	GND	Reference (Ground)
		RX	Signal	Seriel kommunikation
		Speaker	Sound	Lyd
Fejltæller	Tæller fejl	P1	9.6V	Strømforsyning
		P2	GND	Reference (Ground)
		Fejltæller	7-Seg	Antal fejl

Tabel 2- Blokbeskrivelse af BDD



Figur 9 - IBD af bilen

Tabel 3, er et mere uddybende overblik over diverse signaler fundet på Figur 9. Tabellen viser, hvor de forskellige signaler stammer fra og hvilke enheder, der gør brug af signalet. Vi ser altså sammenhængen, mellem kilden (Port 1) og destinationen (Port 2) skrevet i BDD'en.

Signalnavn	Funktion	Område	Port 1 (source)	Port 2 (destination)	Kommentar
GND	Reference til spænding	0V	Power Supply, P4	Arduino, P2 Sensor, P2 Motor, P2 Light, P2 SOMO-II, P2 Fejltæller, P2	Ground
powerSupply	Strømforsyning	5V	Power Supply, P3	Sensor, P1 Motor, P3 Light, P1 SOMO-II, P1	
motorPower	Strømforsyning	7.2V	Power Supply, P2	Motor, P1	
arduinoPower	Strømforsyning	9.6V	Power Supply, P1	Arduino, P1 Fejltæller, P1	
sensorLightOut	Ultrarødt lys	624nm	SensorRight, SensorLightOut SensorLeft, SensorLightOut		Lys ud til refleksbrik
sensorLightIn	Ultrarødt lys	624nm		SensorRight, sensorLightIn SensorLeft, sensorLightIn	Lys ind fra refleksbrik
SensorRightSignal	Signal	0-5V	SensorRight, J1	Arduino, D2	Ved refleksion
SensorLeftSignal	Signal	0-5V	SensorLeft, J1	Arduino, D3	Ved Refleksion

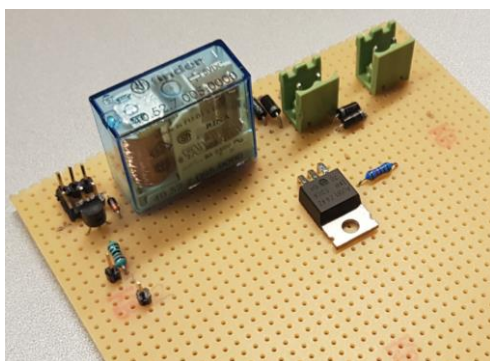
lightFront1 lightFront2	Lys	Hvidt lys	LightFront, Light1 LightFront, Light2		Forlygter
lightBack1 lightBack2	Lys	697nm	LightBack, Light1 LightBack, Light2		Baglygter
motorSpeed	Styrer hastigheden på motoren	0-5V	Arduino, A1	Motor, J2	PWM signal
motorDirection	Styrer retningen som motoren kører	0-5V	Arduino, D1	Motor, J1	Digital signal
lightFrontPWM	Styrer lysintensiteten for forlys	0-5V	Arduino, A2	LightIntensity, J1	Front light
lightBackPWM	Styrer lysintensiteten for baglys	0-5V	Arduino, A3	LightIntensity, J1	Back Light
Fejltæller	Tæller antal fejl	7-Seg	Fejltæller, 7-seg		7 segments display

Tabel 3 - Signalbeskrivelse af IBD-forbindelser

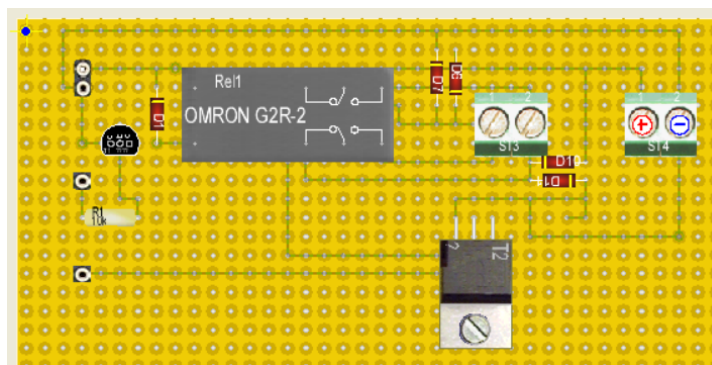
Motorhardware (Gustav)

[illegible]

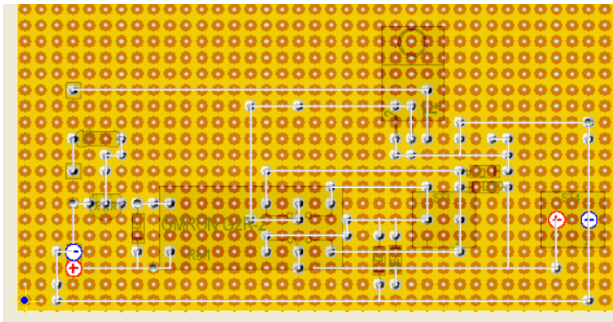
Figur 10 - design af modul for motor



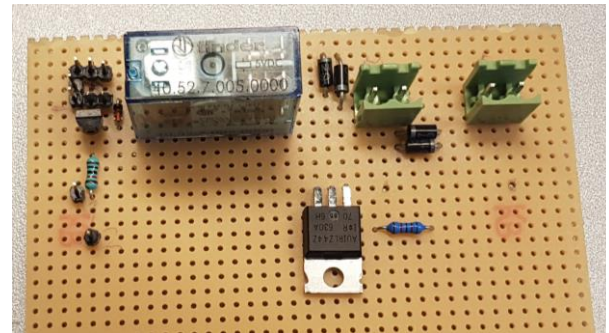
Figur 11. Motor modul



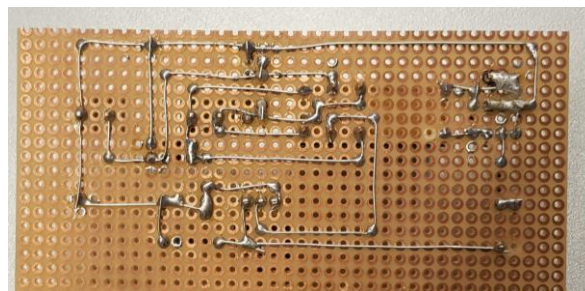
Figur 12. Motor modul front design



Figur 14. Motor modul bagside design



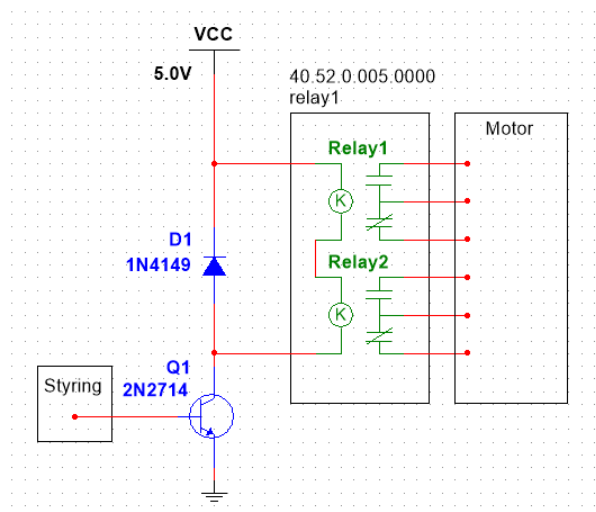
Figur 13. Motor modul front implementering



Figur 15. Motor modul bagside implementering

Relæ strøm

Motor retningen styres af et relæ af typen Finder 40.52.7.005.0000.(Automation, n.d.) Dette er et 0.5W DC, 5V relæ, med en indre modstand på 50Ω. Relæet er ideelt til projektet, da det hurtigt kan skifte mellem de forskellige stadier den har, klare den høje strøm, og tager ikke noget af spændingen fra motoren. På Figur 16 kan det tydeligt ses, hvordan 40.52 er blevet implementeret.



Figur 16. Motorstyring 1/4 - Relæ

Nominal voltage U_N	Coil code	Operating range		Resistance R	Rated coil consumption I at U_N
V		U_{min}	U_{max}	Ω	mA
5	7.005	3.7	7.5	50	100

Figur 17 - Karakteristik for 40.52.7.005.0000 - fra datablad (Automation, n.d.)

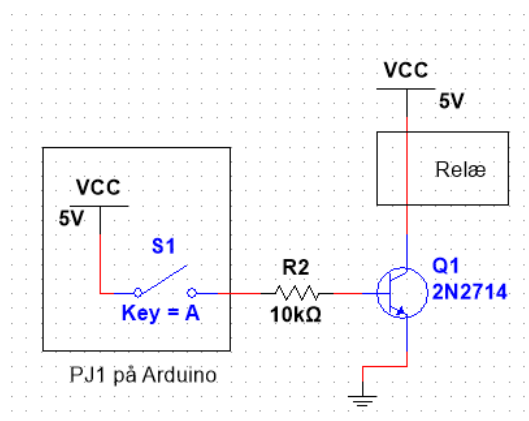
KVL	$V_{cc} + V_{relay1} + V_{Q1} = 0$
Normal spænding over Q1	$V_{Q1_CE(sat)_typ} = 250mV$
Forsynings spænding	$V_{cc} = -5V$
Spænding over relæ	$V_{relæ} = 5V - 250mV = 4.75V$
Strøm gennem relæ	$I_{relæ} = \frac{4.75V}{50\Omega} = 95mA$
Effekt igennem relæ	$W_{relæ} = 428mW$

Tabel 4. Ligninger til forståelse af relæ

Spændingsfaldet over relay1 er altså $5V - 0.25V = 4.75V$. Ifølge Figur 17 skal spændingen over et 40.52-relæ være mellem $3.7V - 7.5V$ og vi er derfor inden for den acceptable spænding. Dermed har 40.52 også et strømforbrug på cirka $I_{relæ} = \frac{4.75V}{50\Omega} = 95mA$

Der er tilføjet et push-back diode af typen "1N4148" over relæet. Dette er for at undgå støj og skader på udstyr fra spolen i relæet, ved pludselige skift af strømmen. Denne diode har $I_{FSM} = 500mA$, hvilket vil sige den kan klare op til 500mA ganske kort.

Relæ styring



Figur 18. Motorstyring 2/4 - Relæ styring

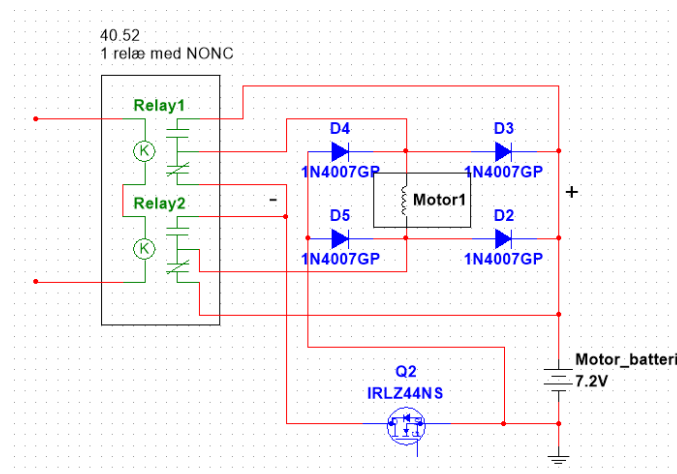
Styringen af strømmen til relæet foregår via en BJT547B-transistor (På Figur 18 bruges en 2N2714 da den har samme egenskaber). ("BJT547B transistor datasheet.pdf," n.d.)

Strøm på collector ben på Q1	$I_{relæ} = I_C = 95mA$
Forstærkningsfaktor for Q1	$\beta_{Q1} = 200 \sim 450$
Strøm i Q1 ved laveste forstærkningsfaktor	$I_{b_min} = \frac{95mA}{200} = 475\mu A$
Strøm i Q1 ved højeste forstærkningsfaktor	$I_{b_max} = \frac{95mA}{450} = 211\mu A$
Minimum formodstand til base på Q1	$R_{b_min} = \frac{5V}{440\mu A} = 10.526,3\Omega$
Maksimum formodstand til base på Q1	$R_{b_max} = \frac{5V}{211\mu A} = 23.696,6\Omega$

Tabel 5. Ligninger til forståelse af Q1

For at sikre der kommer den forventede strøm igennem relæet, implementeres Q1 med en formodstand lidt mindre end R_{b_min} som formodstand. Derfor er $R_b = 10k\Omega$ valgt til modulet.

Motor strøm

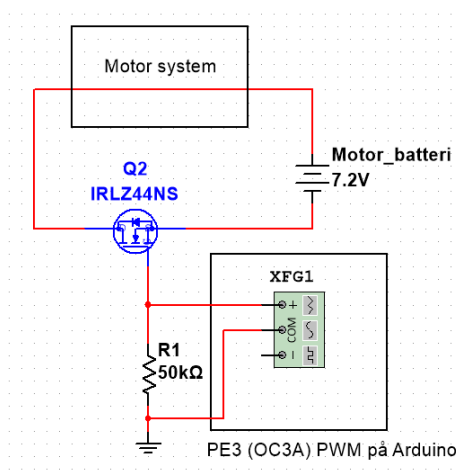


Figur 19. Motorstyring 3/4 - Motor strøm

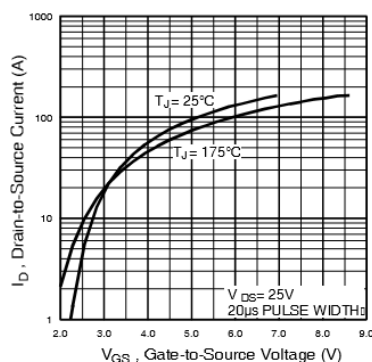
Alt efter om relæet er tændt eller slukket, kan strømmen igennem motoren styres i forskellige retninger.

Da pludselige ændringer i strømmen for en spole kan introducere høje spændinger i systemet, er der indsat push-back dioder af typen 1N4007 mod lede retningen. Dette betyder, at dioder ikke gør noget under normale omstændigheder, men ved skift i lede retning når motoren generer en høj/lav spænding af sig selv, kan denne spænding ledes tilbage til 7.2V/GND, alt efter hvad der er behov for. Disse dioder er valgt, da de har en høj tolerance på 1000V og peak strøm på 30A (gennemsnitlig 1A).

Motor styring



Figur 20. Motorstyring 4/4 - Motor styring



Figur 21. Typisk transfer karakteristisk for IRLZ44N (PrtMosFet, n.d.)

Tabt strøm gennem pull-down modstand for Q2	$I_{pull-down} = \frac{5V}{50k\Omega} = 0.1\mu A$
---	---

Tabel 6. Ligning til pull-down modstand for Q2

Strømmen gennem motoren styres med transistor Q2, en MOSFET af typen IRLZ44N.(PrtMosFet, n.d.) Dette er en N-channel general purpose transistor. Den er ideel til dette system da den kan klare de høje strømme der skal til for at køre motoren. Q2 styrer med et PWM-signal fra mikrocontrolleren (0V-5V). Implementeringen kan ses på Figur 20.

Ved 0V tillader transistoren ikke strømmen igennem, men ved 5V tillader den op til 100A igennem, som det kan ses på Figur 21. 1V~2V, hvilket er der hvor transistoren begynder at tænde.

Q2 har desuden en pull-down resistor på 50kΩ. Dette medfører et lille tab af strøm på cirka 0.1µA, når der går en spænding på Q2 gate.

Lyshardware (Simon og Shyn)

Bilen skal udstyres med forlys og baglys. Forlys og baglys skal tændes ved motorstart, og når bilen bremser eller bakker skal baglyset lyse kraftigere. Dette gøres ved hjælp af PWM-signaler, som sendes via Arduinoen. For at gøre det så realistisk som muligt får bilen monteret to forlygter og to baglygter på henholdsvis højre og venstre side. Dette gøres vha. LED-sæt, hvor hvert LED-sæt vil bestå af 4 individuelle lysdioder. To sæt med hvide LED'er, LTW-2S3D7(Only, n.d.), til forlyset og to sæt med røde LED'er, LH3330-PF(Pcr and Kit, 2012), til baglyset. Både forlys og baglys skal kunne ses

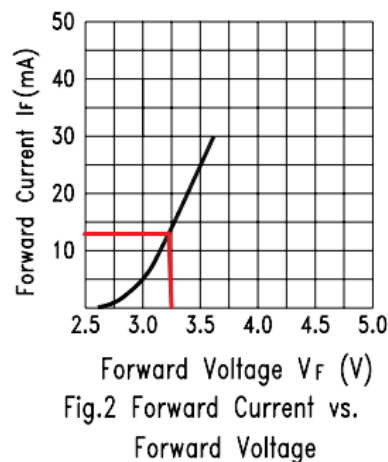
fra 2 meters afstand. Det kan i forvejen forventes, at de hvide dioder til forlygterne vil lyse kraftigere end de røde dioder, da de hvide dioder kræver en højere spænding i forhold til de røde dioder. Kravet for forlys og bremselys er, at middelstrømmen skal være på $50\text{mA} \pm 5\text{mA}$, mens baglysets middelstrøm skal være på $10\text{mA} \pm 2\text{mA}$. Heraf skal der derfor beregnes en formodstand til dioderne.

Forlys

Til forlyset anvendes LTW-2S3D7(Only, n.d.). Bilens forlygter består af 2 gange 4 dioder:

Bestemmelse af formodstand

Forlysets LED-sæts middelstrøm skal være på $50\text{mA} \pm 5\text{mA}$, hvilket betyder, at hver lysdiode skal have en middelstrøm på $50\text{mA}/4=12,5\text{mA}$. For at finde den rigtige modstand bestemmes først en værdi til spændingsfaldet over LED'en ved hjælp af databladet for LTW-2S3D7, som det kan ses på Figur 22:



Figur 22 Hvid led spændingsfald diagram

Ifølge databladet for den hvide diode, skal spændingsfaldet over LED'en være ca. 3,25V. Forsyningsspændingen er på 5V, og derfor bliver spændingsfaldet over modstanden:

$$\begin{aligned} V_{\text{forsyning}} &:= 5 \text{ V} \\ V_{\text{fald_LED}} &:= 3.25 \text{ V} \\ I_{\text{LED_SÆT}} &:= 50 \text{ mA} \end{aligned}$$

$$V_{\text{fald_modstand}} := 5 \text{ V} - 3.25 \text{ V} = 1.75 \text{ V}$$

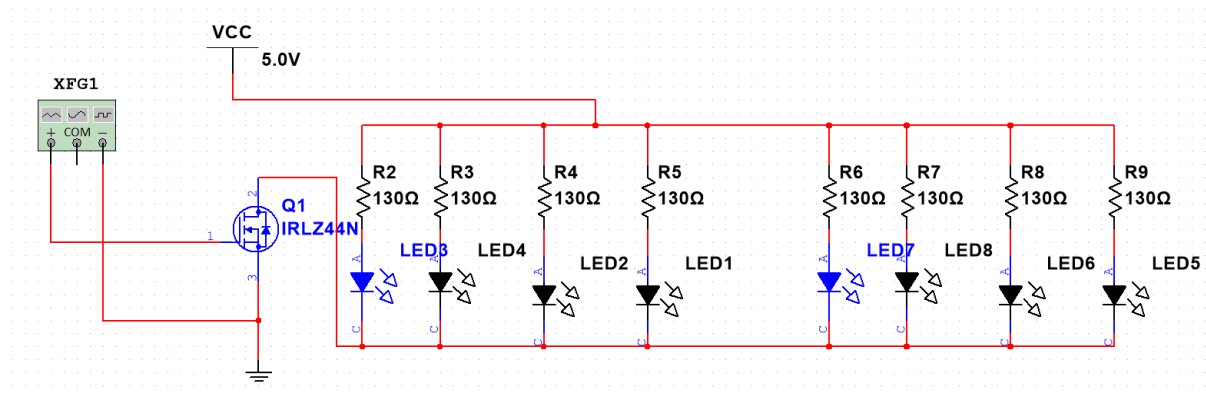
Formodstanden kan derfor bestemmes som følgende:

$$R_{\text{LED}} := \frac{V_{\text{fald_modstand}}}{\frac{I_{\text{LED_SÆT}}}{4}} = 140 \text{ } \Omega$$

Formodstanden til lysdioderne for forlys er beregnet til at være på $140 \text{ } \Omega$ til hver diode. Men da der ikke er en modstand på $140 \text{ } \Omega$ i værkstedet, vælger vi i stedet at anvende en formodstand på $130 \text{ } \Omega$. Årsagen til, at der bruges en modstand på $130 \text{ } \Omega$ er, at større strøm kan justeres ned til den ønskede værdi ved hjælp af software. Kredsløbet kan nu opbygges på multisim.

Diagram for opstilling

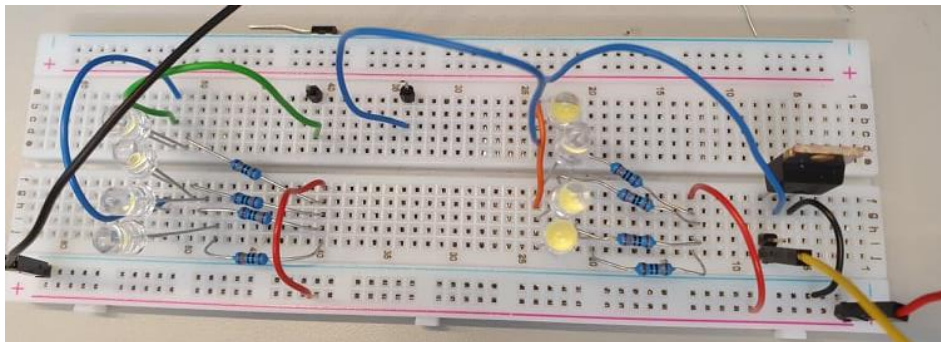
Idet at der skal sendes signaler til kredsløbet via Arduinoen, anvendes transistoren MOSFET IRLZ44n, som har tre ben: Gate, Drain og Source. Signalet sendes ind via Gate, Drain forbindes til LED'erne mens Source går ned til Ground. Kredsløbet for hele forlyset opbygges derfor således, at der anvendes en transistor der deler begge LED-sæt med formodstande. Opbygningen kan ses på Figur 23.



Figur 23 Diagram for opstillingen af LED sæt

Til opbygningen af kredsløbet på multisim for forlys benyttes 8 LED'er, 8 formodstande på 130 Ω og MOSFET IRLZ44n. Forsyningsspændingen på 5V kommer ned igennem modstandene og hen til LED'erne. LED'erne lyser først når transistoren får nok spændingsforskel mellem gate og source. For MOSFET'en har den en Gate Threshold voltage på min 1 V, og Arduino'en giver et firkantsignal mellem 0 og 5 V. Når threshold voltage opnås er der forbindelse mellem LED og ground. Middelstrømmen ser fornuftig ud, og kredsløbet kan nu testes på fumlebræt.

Opstilling på fumlebræt



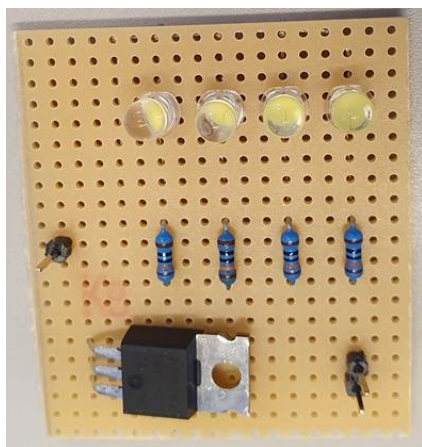
Figur 24 Opstilling af kredsløb på fumlebræt - Forlys

For at teste kredsløbet blev opstillingen sat op på fumlebrættet som kan ses på Figur 24. Der fås en middelstrøm på 105,4mA. Idet vi har at gøre med to LED-sæt passer dette med at ét sæt får en middelstrøm på $105,4\text{mA}/2 = 52,2\text{mA}$. Dette stemmer overens med kravet for forlys, og kredsløbet kan nu bygges på veroboard.

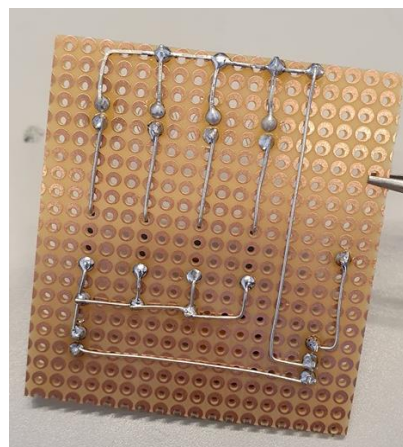
Opbygning på vero board

Bilens forlys består af to LED-sæt, et til højre og et til venstre side af bilen, derfor opbygges to vero boards. I stedet for at benytte to transistorer til forlys anvendes som nævnt én transistor, som deler signalet ud til begge vero boards. Dette gøres ved at VCC og signalledningen forbindes fra vero boardet, hvor transistoren sidder på, til vero-boardet uden transistor.

Side med transistor:



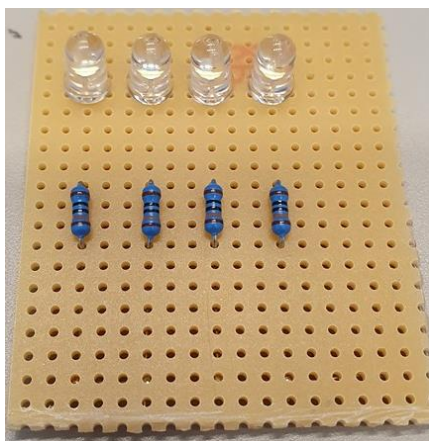
Figur 25 Forside af vero board forlys m. transistor



Figur 26 Bagside af vero board forlys m. transistor

På Figur 25 ses vero boardet med transistoren. Dette vil forbindes til vero boardet uden transistor, som ses nedenstående. Harwin-stikket i venstre side på Figur 25 er til signalet. De to ude til højre er til VCC (for oven) og ground (for neden).

Side uden transistor:



Figur 27 Forside af vero board forlys uden transistor



Figur 28 Bagside af vero board forlys uden transistor

På bagsiden af veroboardet Figur 28 ses de løse ender, som senere vil blive forbundet til vero boardet med transistoren. Den øverste løse ende forbindes til signalledningen, mens den nederste løse ende forbindes til VCC. Det konkluderes vha. test med testprogram, at forlyset fungerer som ønsket. Testprogrammet ses senere i softwaredelen under kapitlen "Lyssoftware".

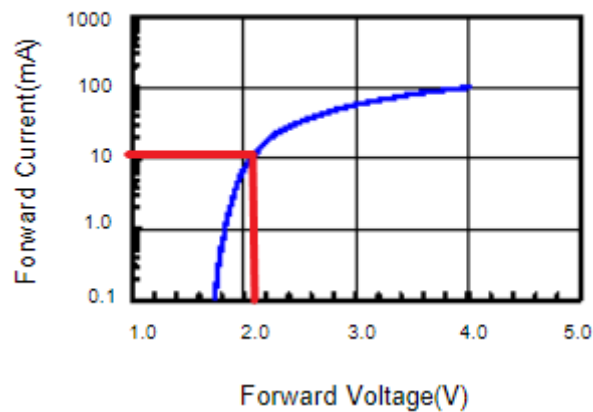
Baglys

Til baglyset anvendes LH3330-PF. Bilens baglygter består af 2 gange 4 dioder:

Bestemmer formodstand

Til baglyset skal vi have en middelstrøm på 50mA til bremselyset og 10mA til almindeligt baglys. Da bremselysets middelstrøm er størst, tager vi udgangspunkt i dette, da man senere kan sætte strømmen ned vha. PWM-signaler. Altså skal bremselysets LED-sæt have en middelstrøm på 50 mA ligesom forlyset, hvilket betyder, at hver lysdiode skal have en strøm på $50\text{mA}/4=12,5\text{mA}$. For at bestemme modstanden bestemmes værdien af spændingsfaldet over LED'en ved hjælp af databladet for LH3330-PF:

Fig.1 Forward current vs. Forward Voltage



Figur 29 Rød LED spændingsfald diagram

Ifølge databladet for den røde diode, skal spændingsfaldet over LED'en være ca. 2,1V. Forsyningsspændingen er på 5V, og derfor bliver spændingsfaldet over modstanden:

$$\begin{aligned} V_{\text{forsyning}} &:= 5 \text{ V} \\ V_{\text{fald_LED}} &:= 2.1 \text{ V} \\ I_{\text{LED_SÆT}} &:= 50 \text{ mA} \end{aligned}$$

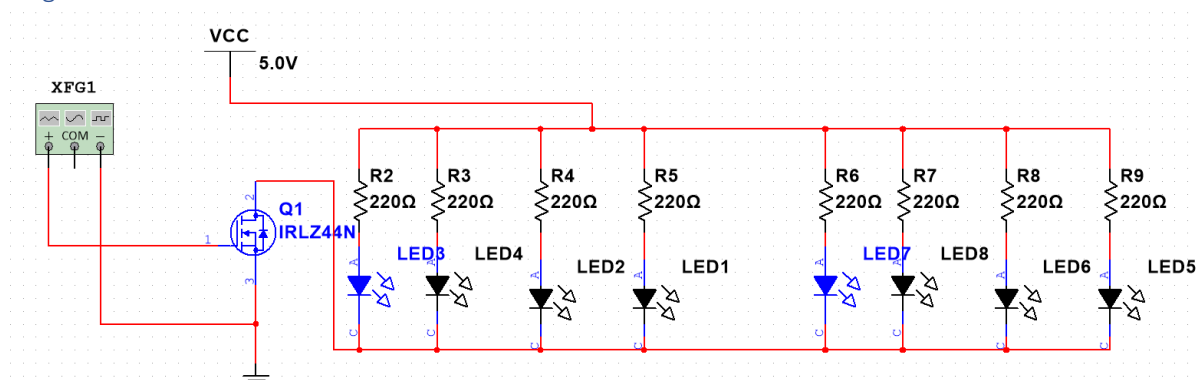
$$V_{\text{fald_modstand}} := 5 \text{ V} - 2.1 \text{ V} = 2.9 \text{ V}$$

Formodstand kan derfor bestemmes som følgende:

$$R_{\text{LED}} := \frac{V_{\text{fald_modstand}}}{\frac{I_{\text{LED_SÆT}}}{4}} = 232 \text{ } \Omega$$

Formodstanden til lysdioderne for baglys er beregnet til at være på 232 Ω , så der anvendes en modstand på 220 Ω . Igen bruges en mindre modstand, for at få nok strøm igennem. Kredsløbet tegnes nu i multisim, så der kan måles på om middelstrømmen er på 50mA.

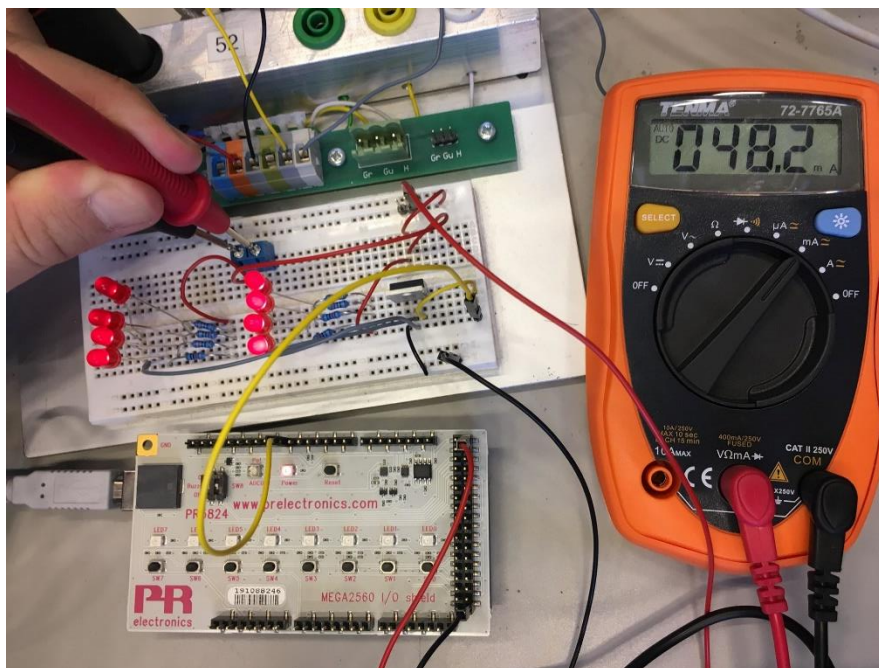
Diagram



Figur 30 Diagram i multisim

Til opbygning af baglys i multisim, er lavet ligesom diagrammet (Figur 23), men formodstandene er skiftet ud til 220 Ω . Middelstrømmen så fornuftigt ud, og kredsløbet opbygges på fumlebræt.

Opstilling på fumlebræt

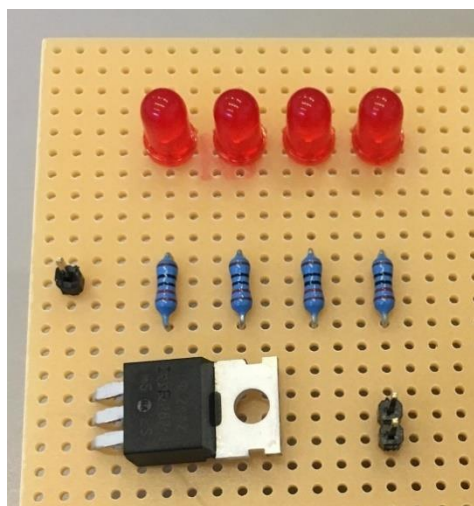


Figur 31 Opstilling af kredsløb på fumlebræt - Baglys

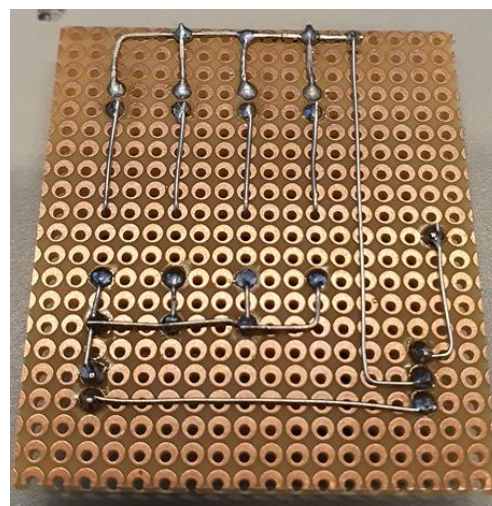
Kredsløbet testes nu på fumlebræt. Signalet fra den gule ledning kommer fra Arduino'en, som sendes ind i MOSFET'en så man kan få en PWM-signal til baglysene. Der fås en middelstrøm på 48,2mA ved bremselys og 9,6mA ved almindeligt baglys, så kredsløbet kan nu bygges på veroboard.

Opstilling på vero-board

Ligesom med forlysgælder det samme for baglys. De to LED-sæt til baglyset styres af en transistor. Bilen vil have to vero-boards på højre og venstre side, hvor vero-boardet uden transistor bliver forbundet med vero-boardet med transistor ved hjælp af signal- og VCC-ledning.



Figur 32 Forside af vero board



Figur 33 Bagside af vero board

Harwin-stikket til venstre på Figur 32 er til signalet. De to ude til højre er til VCC (for oven) og ground (for neden).

Generelt kan der siges, at bilen som sagt skal udstyres med forlys og baglys ved hjælp af hvide og røde lysdioder, hvoraf baglys også skal fungere som baklys og bremselys. Forlys og baglys tændes, når motoren startes. LED'erne skal have formodstande, som er blevet bestemt til at være $130\ \Omega$ for hvid LED og $220\ \Omega$ for rød LED. Disse er blevet bestemt ved at finde spændingsfaldet over LED'en i databladet for de to forskellige dioder.

SOMO-II (Rasmus & Camilla)

SOMO-II er en forkortelse for navnet Sound Module-II, det er et hardware-komponent anvendt til at afspille lyd gennem en forbundet højttaler. (Module, 2019) SOMO-II kan gennem et SD-kort afspille de pågældende MP3-filer på SD-kortet og fungerer som forbindelsen mellem lydfiler og højttaler.

For at forbinde SOMO-

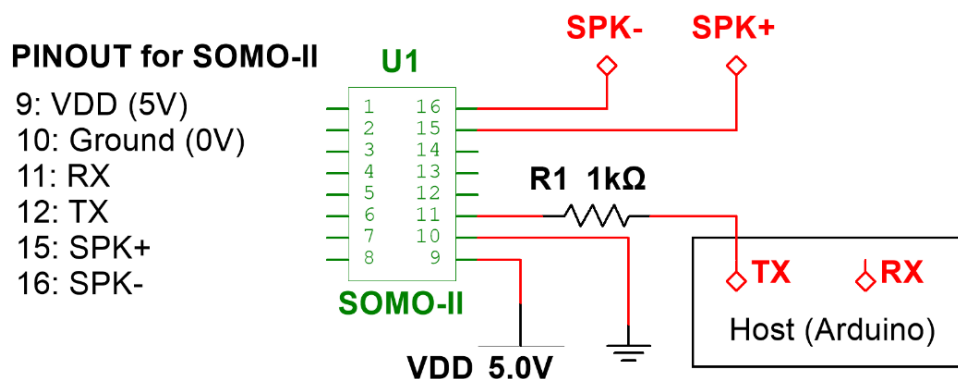
komponenten, anvendes opstillingen vist på Figur 35. Som det ses på Figur

35 forbindes SOMO-II til en host, i tilfældet for opstillingen til bilen, er denne host en Arduino.

Arduinoen fortæller gennem forbindelsen til SOMO-II ben 11 - receiver-benet, SOMO-II-komponenten, hvornår der er behov for afspilning af lyd. Hertil anvendes en $1k\Omega$ modstand mellem Arduinoens TX, transmitter-ben, og SOMO'ens RX for at undgå at SOMO'en brænder af¹. Ben 15 og 16 på SOMO'en anvendes til forbindelsen af højttaleren, på Figur 35 afbilledet som "SPK-" og "SPK+".



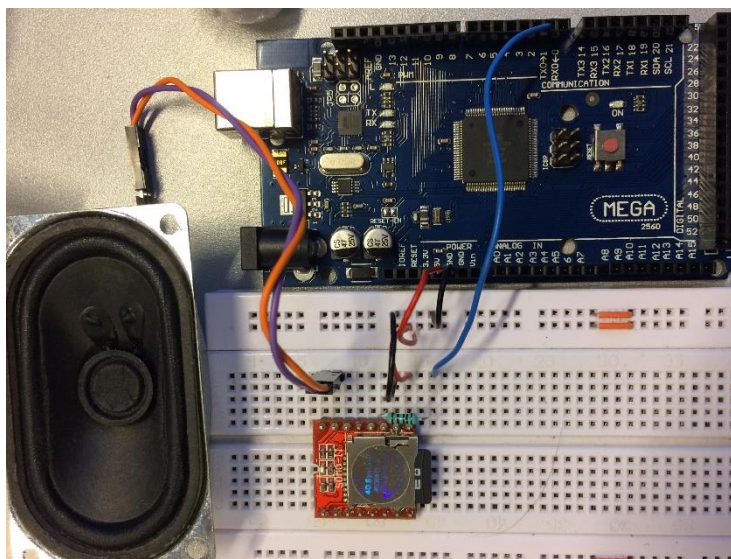
Figur 34 Pinout for SOMO-II



Figur 35 Design af SOMO-II

¹ Se "NOTE" i SOMO-II datablad afsnit 7.1 microSD-card

Til Figur 35 Design af SOMO-II er databladet for SOMO-II anvendt og her er PINOUT-diagrammet fundet. Ifølge dokumentationen ser den ud som på Figur 34. På Figur 36 kan man se en testopstilling af SOMO'en på et fumlebræt. Her kan man se en blå ledning, der løber fra TX på Arduinoen til RX på SOMO'en, hvor der forinden er en 1k Ohm modstand. Dette er den serielle forbindelse mellem Arduinoen og SOMO'en.



Figur 36 Testopstilling for SOMO-II

Sensor (Andreas & Anders)

For at bilen kan registrere refleksbrikkerne på banen er der behov for en lyssensor. Her anvendes to ens sensorkredsløb lavet under værkstedspraktik.

Sensoren, som ses på Figur 37, udsender et ultrarødt lys. Via refleksion vil en fotodiode absorbere det infrarøde lys, hvilket vil generere et signal på et Harwin-stik. Dette signal videreføres med ledninger til Arduinoen, hvor det bruges til at lave interrupts i programmet.



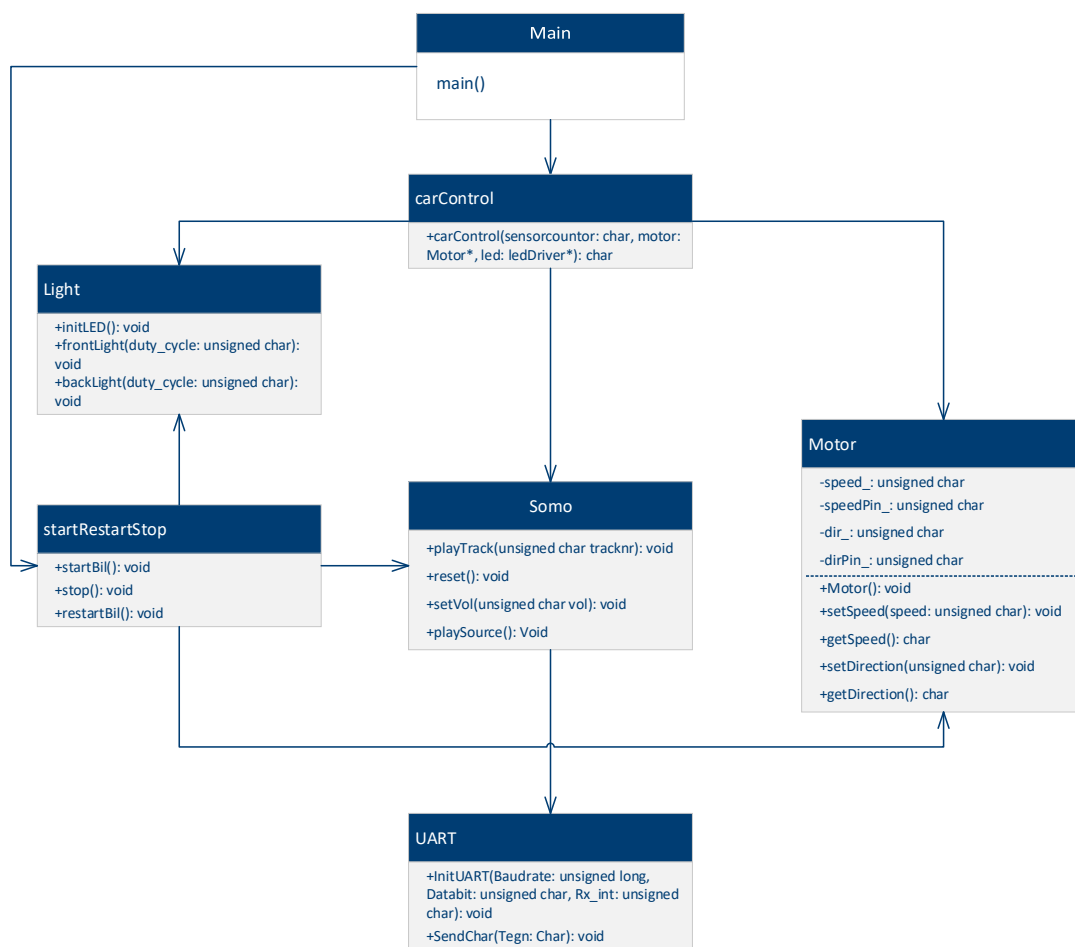
Figur 37 - sensorkredsløb

Software

Følgende afsnit beskriver den overordnede software arkitektur og herefter, hvordan de forskellige klasser og deres funktioner er opbygget. Herunder vises de forskellige testprogrammer for klasserne.

Software arkitektur (Alle)

På Figur 38 ses den samlede SW-arkitektur opstillet i et UML klasse-diagram. Principielt er det kun Motor og Led der er klasser, men for overskuelighedens skyld er den samme notationsform holdt i hele diagrammet. Pilene til og fra de moduler som ikke er klasser, viser derfor naturligvis ikke klasserelationer, men angiver på lignende vis, en relation mellem funktioner og hvor de anvendes.



Figur 38 - UML klasse-diagram for den samlede SW

Software design

Den overordnede software gennemløber, vha. signaler fra lyssensorerne, en række cases, som udfører funktioner der justerer bilens HW, til bestemte tidspunkter.

Vi har tre forskellige HW-specifikke funktioner/klasser (Motor, Led og SOMO), der råder over hver deres del af bilen.

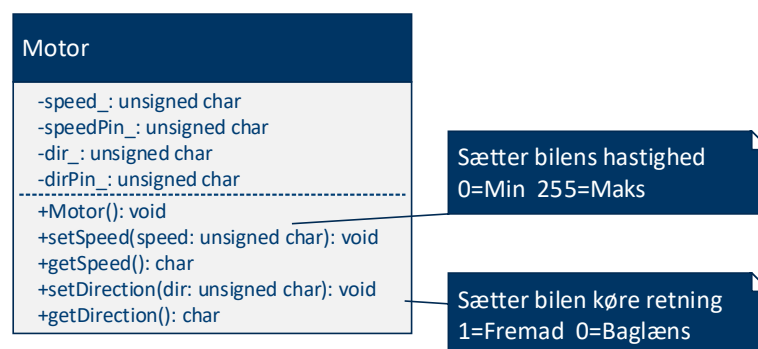
Yderligere består softwaren af funktionen carControl(), der sørger for at de rigtige justeringer sker de rigtige steder på banen.

Hovedalgoritmen, der befinder sig i main.cpp, bestemmer hvornår bilen skal skifte stadie og sender informationerne fra diverse funktioner/klasser videre til carControl. Algoritmen påbegynder selve funktionaliteten af alt software og er i stand til at resette og stoppe processen, alt via trykknapper på Arduino shield'et.

Motor software (Gustav)

Modulbeskrivelse af klassen "Motor"

Klassen Motor styrer retningen og hastigheden af motoren på bilen. Retningen styres af et digitalt signal til en transistor der styrer et relæ. Hastigheden styres af en transistor, som styres med en frekvens på 122Hz. Klassens motor kan gå ind og ændre dutycycle for dette signal, for at styre hvor ofte signalet er tændt. Klassens implementering kan overordnet ses på UML klassen på Figur 39.



Figur 39 UML notation for Motor

Ansvar: Kontrol af bilens motor, så den kan køre fremad eller baglæns med en variabel hastighed.

Motor()

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Constructor. Opsætning af porte PE3 og PJ1 til motor signaler, samt motor i fremad position og hastighed 0

void setSpeed (unsigned char)

Parametre: speed, et tal mellem 0-255 der fortæller motorens hastighed

Returværdi: ingen

Beskrivelse: Tjekker om speed_ er en acceptabel værdi (0-255). Hvis speed_ er acceptabel vil værdien blive sat, ellers vil speed_ blive 0. Desuden starter den et 8-bit

non-fast PWM-signal i non-inverting mode, på timer3 med en frekvens på cirka 100Hz
Fra Øvelse 4, findes det, at denne frekvens er optimal.

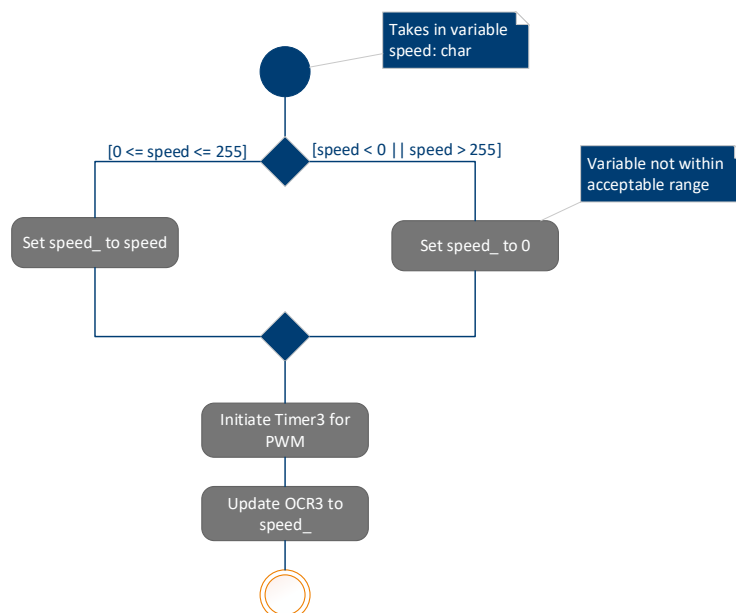
$$Prescaler = \frac{F_{CPU}}{2 * Frekvens * Top} = \frac{16.000.000Hz}{2 * 100Hz * 256} = 312.5 \approx 256$$

Ligning 1. Beregning af prescaler til Timer3

Prescaler for Timer3 kan beregnes som i Ligning 1. Det rundes op/ned til nærmeste mulige prescaler, hvilket giver en prescaler på 256. Derfor bliver den endelige frekvens 122Hz, som det kan ses på Ligning 2. Da timer3 i non-fast mode er en 8-bit timer, sættes speed_ direkte som OCR3A, hvilket kan bestemme dutycyclen.

$$Frekvens = \frac{F_{CPU}}{2 * Prescaler * Top} = \frac{16.000.000Hz}{2 * 256 * 256} = 122Hz$$

Ligning 2. Beregning af frekvens



Figur 40 Rutediagram for setDirection

unsigned char getSpeed ()

Parametre: Ingen

Returværdi: Unsigned char

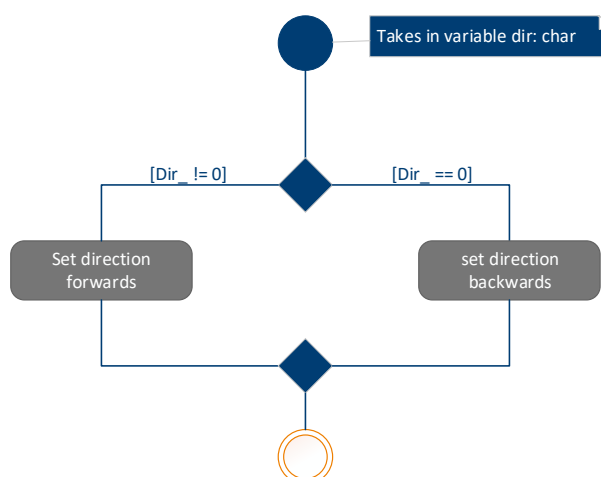
Beskrivelse: Returnerer værdien af speed_ (nuværende hastighed).

void setDirection (unsigned char dir)

Parametre: dir bestemmer hvilken vej motoren kører

Returværdi: Ingen

Beskrivelse: Ændre motormodulets strømretning. Hvis dir er 0, sættes dir_ til 0, er relæet slukket og motoren kører baglæns. Er dir ikke 0, sættes dir_ til 1 og motoren kører fremad.



Figur 41 Rutediagram for setDirection

Unsigned char getDirection ()

Parametre: Ingen.

Returværdi: Unsigned char

Beskrivelse: Returnerer værdien af dir_ (nuværende retning).

Test af Motor klasse

For at teste Motor class skrives et kort program der tester deklarering af et objekt myMotor i klassen Motor (Figur 42). Her efter testes start af motor, ændring af hastighed og stop af motoren. Derefter ændres motorens retningen og motoren startes og stoppes igen. Delays er indsat mellem hver funktion, så der er tid til at se effekten alle funktionerne.

```
int main(void)
{
    /* Replace with your application code */
    while (1)
    {
        Motor myMotor;          //Declaring class
        _delay_ms(1000);
        myMotor.setSpeed(150);
        _delay_ms(2000);
        myMotor.setSpeed(255);
        _delay_ms(2000);
        myMotor.setSpeed(0);
        _delay_ms(1000);
        myMotor.setDirection(1);
        myMotor.setSpeed(180);
        _delay_ms(2000);
        myMotor.setSpeed(0);
        while(1){}
    }
}
```

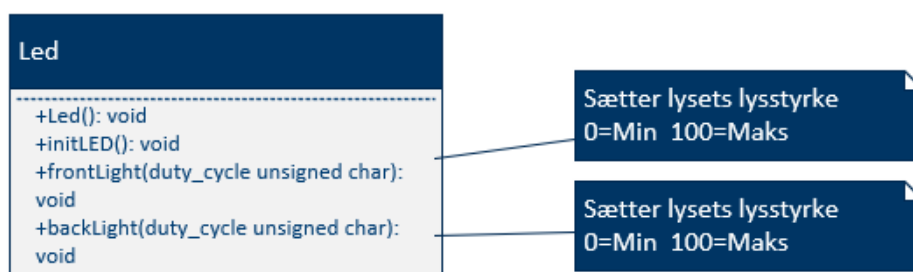
Figur 42 Testprogram for Motor klasse

Lys software (Simon og Shyn)

Softwaren skal sørge for at forlyset tændes, når motoren tænder, og slukkes, når motoren slukker. Softwaren for forlys er meget simpel idet, at den blot skal initieres når motor startes. Dette bliver udført ved en case som starter både motor og signal til forlysene og lignende slukker lys når motor slukkes. For almindeligt baglys gælder det samme, lyset skal tændes, når motoren tænder, og slukkes, når motoren slukker. Dog skal lysstyrken for baglys være kraftigere, når bilen bremses og bakker. Dette styres ved hjælp af PWM-signaler. Software for forlys og baglys styres begge ved hjælp af PWM-signaler, heraf ved hjælp af den indbyggede Timer 4 i Arduino'en, som initieres. Signal for både forlys og baglys sendes ud ved hjælp af port H. PWM-signalet ændres ved at indsætte en duty cycle, hvor softwaren omregner det til en brugbar værdi, der indsættes ind i OCR4n(Ligning 3). Ved brug af duty cycle, som er en justerbar værdi behøves kun en funktion for både baglys og bremselys.

Modulbeskrivelse af Klassen "Led"

Klassen Led styrer forlys og baglys. Klassen har constructor, initLed og funktioner til styring af PWM-signal der udsendes. Dette kan ses på Figur 43.



Figur 43 UML-notation for Led klassen

Ansvar: Led-driveren har ansvar for at håndtere signalet, som giver værdier til at sætte lysstyrken for baglys og sende signal til forlys.

Void Led();

Parametre: void

Returværdi: void

Beskrivelse: Constructor som sætter port H til outputs og indstiller timer 4 til PWM signal

Void initLED(void);

Parametre: void

Returværdi: void

Beskrivelse: Sætter porte (Port H) til outputs og timer (Timer 4, fast PWM mode 10 bit) til at lave PWM signal.

Lys skal have en stor nok frekvens så det ikke er synligt at det tænder og slukker, blev prescaler sat til at være 1 og frekvensen er således: $Frekvens = \frac{16 MHz}{1 \cdot (1+1023)} = 15625 Hz$.

Void frontLight(unsigned char);

Parametre: unsigned char duty_cycle (0-100)

Returværdi: void

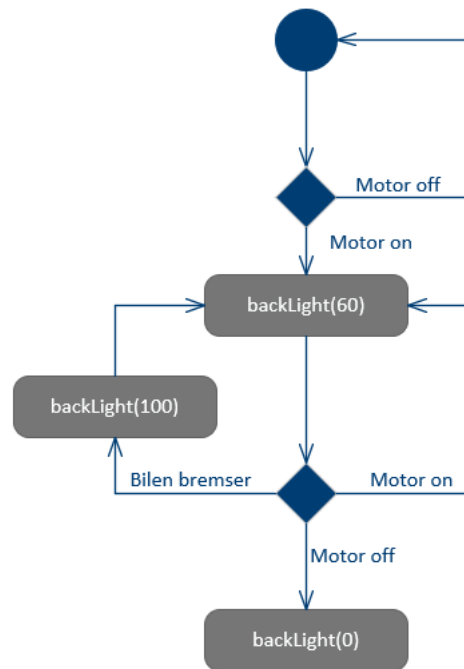
Beskrivelse: Sætter styrken på forlyset ud fra parametren duty cycle

Void backLight(unsigned char);

Parametre: unsigned char duty_cycle (0-100)

Returværdi: void

Beskrivelse: Sætter styrken på baglyset ud fra parametren duty cycle



Figur 44 Rutediagram for baglys

For at kunne sætte en duty cycle mellem 0-100 ind i koden, skal værdien omregnes om til en værdi som svarer til den rigtige OCR4n værdi som skal sættes ind. Man kan se omregningen ved Ligning 3 hvor 1023 er TOP værdien og duty cycle er værdien som indtastes.

$$OCR4n = \left(\frac{1023}{100} \right) \cdot (100 - dutycycle)$$

Ligning 3 Beregning af OCR4n ud fra duty cycle

Test af Led funktion

Ved hjælp af et testprogram bliver Led-driveren testet for at kunne justere PWM signalet. Der bliver defineret 2 forskellige lys klasser, og port H bliver initieres som port ud. Lysene bliver tændt, hvor der ændres i duty cycle. Test programmet kan ses på Figur 45.

```
int main()
{
    // Class definitions
    Led frontLight; // PH5
    Led backLight; // PH4

    // initiering
    frontLight.initLED();

    while (1)
    {
        frontLight.frontLight(100);
        backLight.backLight(100);
        _delay_ms(500);
        frontLight.frontLight(0);
        backLight.backLight(20);
    }
}
```

Figur 45 Test program for LED klasse

Somo software (Rasmus & Camilla)

Som beskrevet tidligere i afsnittet, ”

SOMO-II (Rasmus & Camilla)” bliver SOMO’en anvendt til at afspille lyd. For at afspille en fra lyd efter SOMO'en er det relevant at vide, hvordan man ”snakker” med dette hardware-komponent. Det gøres gennem software, hvor der tilsendes forskellige kommandoer i form af en besked sammensat af hex-værdier. Disse beskeder bliver sendt via UART (Universal Asynchronous Receiver/Transmitter), hvilket afsendes af Arduino’en.

Beskeden der tilsendes til SOMO'en, skal opstilles på en specifik måde, for at SOMO-II kan forstå den besked der sendes til den. Beskeden opstilles på følgende format:

\$S, CMD, Feedback, Para1, Para2, Checksum1, Checksum2, \$0

Navn	Indhold	Beskrivelse
\$S	Start karakteren, som er 0x7E	Alle kommandoer starter med denne værdi
CMD	Kommandokoden (opgivet i Databladet)	De forskellige kommandoer har hver et hex-tal, dette skrives her, så det specificeres, hvilken kommandoer SOMO'en skal igangsætte.
Feedback	Feedback kommando	Specificerer om den skal sendes feedback fra SOMO'en. 00 for ingen feedback og 01 for feedback.

Para1	Parameter #1	Første parameter for den specifikke kommandokode
Para2	Parameter #2	Anden parameter for den specifikke kommandokode
Checksum1	Checksum #1	Første byte af checksummen
Checksum2	Checksum #2	Anden byte af checksummen
\$0	Slut karakteren, som er 0xEF	Kode der fortæller at beskeden er slut.

Tabel 7 - Beskrivelse af kommando argumenter

Ovenstående Tabel 7 viser, hvad de forskellige dele af en kommando besked beskriver.

Checksummen som i Tabel 7 er delt op i 2 bytes er en udregning som beskriver den samlede størrelse af kommandoen, den bruges af SOMO'en til at tjekke om kommandoen er rigtigt udregnet og der ikke er opstået støj så den har fået en forkert kommando. Checksummen udregnes ved hjælp af følgende formel:

$$Checksum (2 bytes) = 0xFFFF - (CMD + Feedback + Para1 + Para2) + 1$$

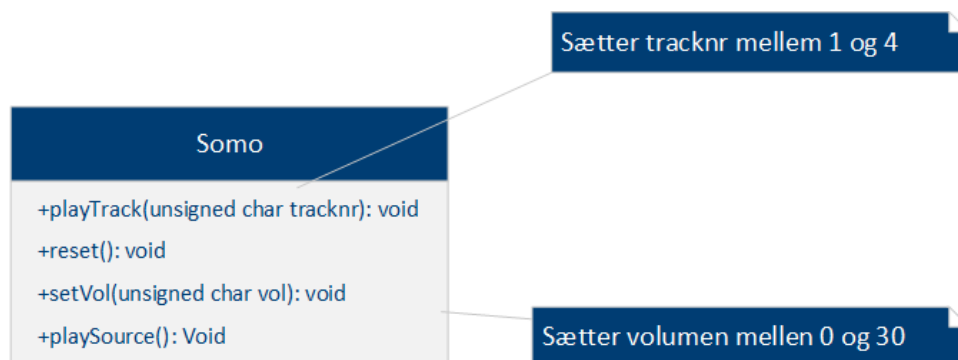
Ligning 4 Udregning af Checksum

Nedenstående tabel, Tabel 8, afbilder de forskellige kommandoer der anvendes under gennemkørslen af banen med bilen.

Command	Command code	Beskrivelse
Reset	7E 0C 00 00 00 FF F4 EF	Reset af SOMO-II, til "start-stadie".
Play Source	7E 09 00 00 02 FF F5 EF	Bestemmer, hvilken "source" der skal spilles fra, der kan spilles fra et USB-stik eller et SD-kort. Dette er kommandoen for afspilning fra SD-kort.
Specify Track 1	7E 03 00 00 01 FF FC EF	Specificerer hvilken sang fra source, her SD-kortet, der skal afspilles.
Specify Track 2	7E 03 00 00 02 FF FB EF	
Specify Track 3	7E 03 00 00 03 FF FA EF	
Specify Track 4	7E 03 00 00 04 FF F9 EF	
Volume #	7E 06 00 00 1E FF DC EF	Specificerer volume-niveauet, her for max, da det er den der anvendes.

Tabel 8 - Tabel over samlede kommandoer

SOMO'en skal kunne afspille 4 forskellige lyde, alt efter, hvilken "use-case" bilen er nået til på banen. SOMO'en skal afspille på max volumen, der er på 30 (1E i Hex). Herudover anvendes "reset"-kommandoen, da det er relevant at SOMO'en er sat til start-stadiet, hvor der ikke er registreret tidligere kommandoer fra tidligere brug, som kan komme i vejen for den nye kode der anvendes under gennemkørslen.



Figur 46 UML notation for SOMO modulet

Modulbeskrivelse af SOMO-funktioner

Følgende vil funktionerne i somo.h & somo.cpp blive beskrevet med parametre, returnværdi og en funktionsbeskrivelse. Ovenfor på Figur 46 kan man se prototyperne.

Ansvar:

SOMO-funktionerne har til ansvar at afspille valgte lyde når vi bl.a. starter, rammer refleks eller kommer i mål.

Void playSource();

Parametre: void

Returnværdi: void

Beskrivelse: Funktionen er en del af initieringen, hvor denne funktion vælger at der skal afspilles lyd fra SD-kortet

Void playTrack(unsigned char tracknr);

Parametre: Det nummer som SOMO'en skal afspille

Returnværdi: Void

Beskrivelse: Sender kommandoen, som vælger nummeret der skal afspilles fra SD-kortet

Void reset()

Parametre: void

Returnværdi: void

Beskrivelse: Sender reset kommandoen til SOMO'en

Void setVol(unsigned char vol)

Parametre: Volumen for afspilning

Returnværdi: void

Beskrivelse: Sender en kommando, der sætter volumen som SOMO'en skal afspille med

Test af SOMO

```
#include "somo.h"
#include "uart.h"
#include <utilities/delay>
#define F_CPU 16000000

int main(void){
    /* Replace with your application code*/
    initUART(9600,8,0); // 9600 baudrate, 8 bit, no interrupt
    reset(); // resets SOMO-II
    playSource(); // Plays from SD-card
    setVol(30); // Volume to max (0x1E)
    playTrack(1); // plays first track
    _delay_ms(5000); // 5 sec delay to hear the sound
    playTrack(2); // plays second track
    _delay_ms(5000);
    playTrack(3); // plays third track
    _delay_ms(5000);
    playTrack(4); // plays fourth track
    _delay_ms(5000);
    while(1){}
```

Figur 47 - Testprogram for SOMO-II

På ovenstående figur, Figur 47, kan man se et lille program, der tester SOMO'en og det bibliotek af funktioner som der er blevet skrevet til projektet. Programmets formål er at teste om der opstår forbindelse og om SOMO'en afspiller lyde. Disse funktion kald (playTrack()) er efterfulgt af et 5 sekunders forsinkelse, dette er til for at man ville kunne nå at høre det individuelle nummer og den ikke skifter med det samme.

Sensor software (Andreas & Anders)

Funktionerne start(), restart() og stop() viser når Mega2560 skifter stadige, og bestemmer om bilen skal starte, genstarte eller stoppe. Dette gøres og vises ved knapperne/LED'erne på Mega2560 shield'et. carControl styre hvad bilen skal gøre, ved den specifikke reflexbrik. ledCounter viser hvilken case vi er ved, altså hvad bilen er i gang med at udføre.

Modulbeskrivelse af startRestartStop

Ansvar:

Modulet har til ansvar at skifte/vise stadiet bilen befinder sig i. Her bruges brugerens input fra Mega2560 I/O shield'et fra MSYS, til at ændre dette. LEDCounter viser hvilken case vi er i.

Funktioner:

void startBil()

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Giver tegn til at bilen initialiserer om, hvornår den er klar til kørsel. Dette sker ved opstart af bilen, eller efter der er blevet lavet et reset af bilen. Dette ses ved, at LED'erne på Mega2560 shield'et vil blinke, i en specifik sekvens.

void restartBil(Motor* motor, ledDriver* led)

Parametre: Pointer til motor objekt, pointer til LED driveren

Returværdi: Ingen

Beskrivelse: Når statusBtn bliver større end 1, statusCounter bliver lig -1 eller quitBtn bliver lig -1, vil bilen blive resat. Dette ses, ved at LED'erne på Mega2560 shield'et, vil blinke i en specifik sekvens.

void stopBil()

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: Hvis quitBtn er lig -1, vil bilen, via LED'erne på Mega 2560 shield'et, give tegn på, at bilen er ved at stoppe, og bryde ud af main funktionen, og dermed stoppe hele koden. Efter stopBil() er blevet kørt, vil et restart af selve Mega2560 være nødvendigt.

void LEDCounter ()

Parametre: Ingen

Returværdi: Ingen

Beskrivelse: LEDCounter viser, ved hjælp af LED'erne på Mega2560 shield'et, hvilket case/stadie bilen er i.

Modulbeskrivelse af carControl()

Ansvar: Modulet bruges til at skifte mellem de cases bilen kan befinde sig i. Hertil bruges dette modul som hovedalgoritme, da det ændrer på de andre SW-moduler og derved styrer, hvordan bilen opfører sig.

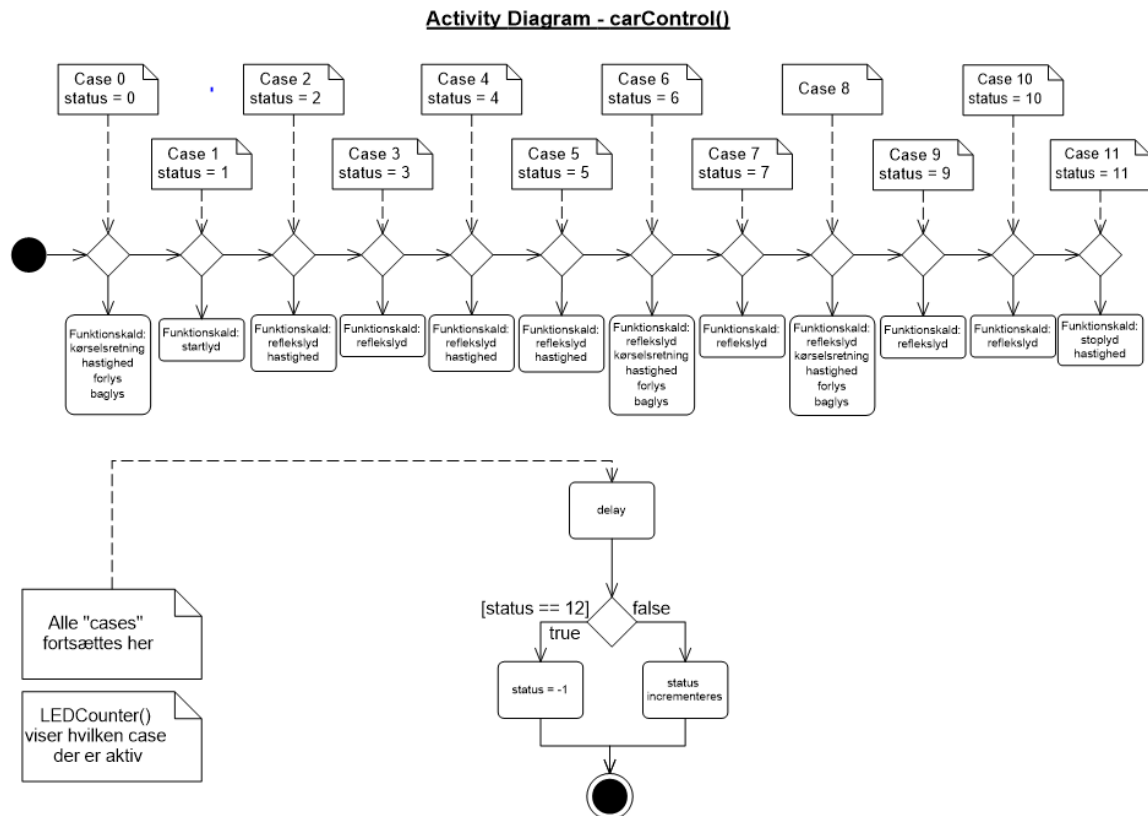
Funktion:

char carControl(char sensorCounter, Mortor* motor, ledDriver* led)

Parametre: sensorCounter bestemmer hvilken case der skal aktiveres. Motor giver adgang til objektet Motor. led giver adgang til objektet Led.

Returværdi: char sensorCounter.

Beskrivelse: carControl kaldes fra main(). Ved kald af carControl(), sendes variablen "sensorStatus" med. CarControl() udfører den "case" der matcher sensorStatus, altså det stadie bilen er i. For hvert kald af carControl(), stiger variablen "sensorCounter" med 1, indtil sidste "case" er udført, hvor sensorCounter ændres til -1 hvorved hovedalgoritmen restarter. Se diagrammet for carControl, som ses på Figur 48, for at få en visuel beskrivelse af carControl().



Figur 48 – et diagram over hvordan carControl() styre bilen ved en specifikke case

Main.cpp:

Del 1 – Initiering før main funktionen påbegyndes:

Først initieres alle globale variabler (btnStatus, sensorStatus, sensorCounter og quitBtn). Dernæst fremgår alle interrupt service rutiner (INT2_vect, INT3_vect, INT4_vect og INT5_vect).

Hvert interrupt ændrer værdien for en given global variabel.

Del 2 - Main funktionen påbegyndes:

Alle porte klargøres, hardwaren bliver initialiseret og alle interrupts aktiveres.

Del 3 - Hovedalgoritmen påbegyndes:

Påvirkning af interrupt INT2_vect (SW2), ændrer btnStatus (0 -> 1 -> 2 -> 0). Påvirkning af interrupt INT3_vect (SW3), sætter quitBtn til -1.

Tilstand af btnStatus og quitBtn

btnStatus = 0: tomgang

btnStatus = 1: start

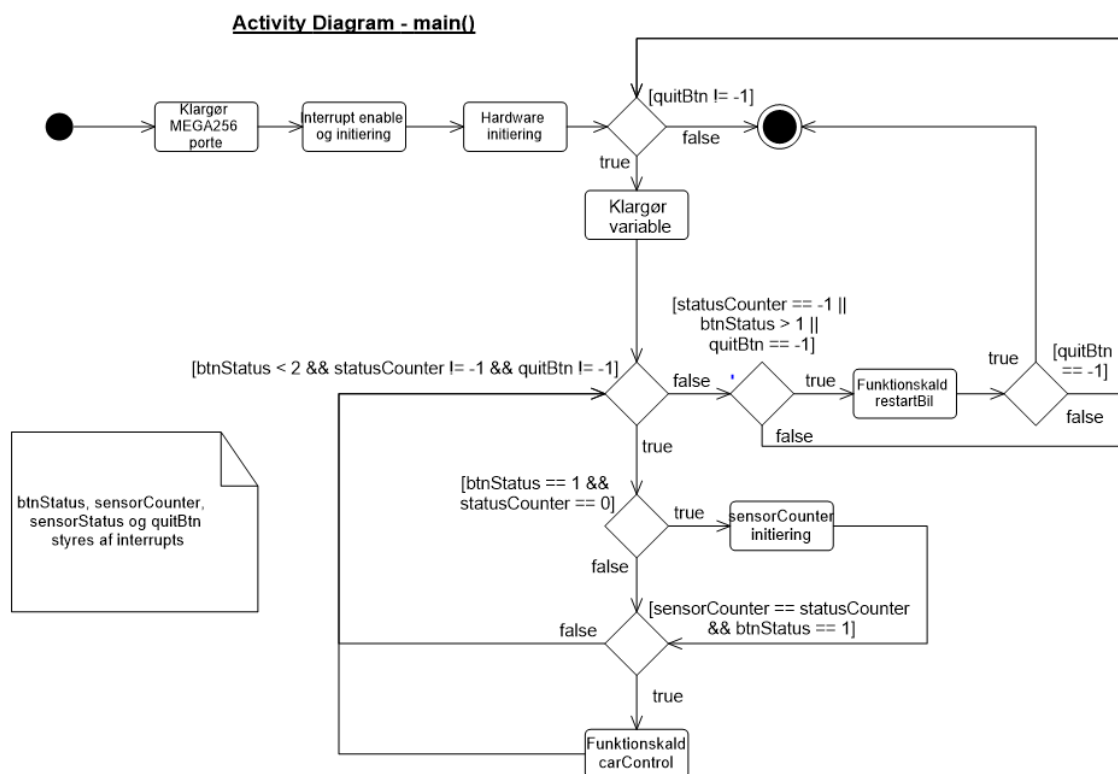
btnStatus > 2: restart

quitBtn = -1: programmet afsluttes

Beskrivelse

Ved brug af interrupts INT4_vect og INT5_vect (sensorer) sættes sensorStatus lig sensorCounter. Da sensorStatus sættes lig sensorCounter, vil sensorStatus aldrig blive højere end sensorCounter.

sensorCounter fortæller hvor mange gange funktionen "carControl" er blevet kørt og inkrementeres i funktionen carControl(). SensorStatus bestemmer altså hvilket stadie bilen er i. Når sensorStatus sættes lig sensorCounter påbegyndes næste stadie af i carControl(). Herefter, stiger sensorCounter, og der ventes på næste stadie af sensorStatus (hvilket sker ved næste reflexbrik). Da sensorStatus aldrig vil blive højere end sensorCounter, vil der aldrig misses et stadie, selvom der sker flere interrupts på en gang. Se Figur 49 for at få en visuel forståelse af main.cpp.



Figur 49 - diagram der giver et overblik over, hvordan hovedalgoritmen er sammensat

Test af main() og carControl()

main() er blevet testet via både trykknapperne og LED'erne på Arduino shield'et, sensor signaler og ved brug af carControl().

Først er main() testet, ved at se om en knap (brug af interrupt INT2_vect, knap SW2), er i stand til at påbegynde algoritmen. Det ses ved at aktivering af LED'er starter hovedalgoritmen, når btnStatus går fra 0 til 1. Herefter undersøges hvorvidt LED'erne slukkes, når btnStatus går fra 1 til 2. Sidst, ses det om der kan slukkes for alle signaler, og derved stoppe hovedalgoritmen, når quitBtn går til -1.

Herefter bliver sensor signalet simuleret, ved at bruge trykknapperne på Arduino shield'et (brug af interrupt INT3_vect, knap SW3). "Sensor signalerne" og carControl() blevet sat til, så der kan testes, om der er en reaktion, når sensoren modtager et signal. Her testes både carControl() og hovedalgoritmen. Ved registrering af et signal vil carControl() blive aktiveret, og LED'er på Arduino shield'et tændes. Her tæller den 1 LED op ved hvert signal, herved sikres det at alle cases bliver gennemløbet.

Sensorerne bliver nu aktiveret, så Mega2560 modtager signaler fra sensoren (interrupt INT4_vect og INT5_vect), når sensoren bliver aktiveret. Herefter er det samme processer, som ved de simulerede signaler.

Accepttest

Funktionelle krav

Use Case 1: "Kør banen"	Test	Forventet resultat	Resultat	Godkendt/ kommentar
Punkt 1 + punkt 2	Brugeren tænder for bilen og placerer den, så den kører forlæns ind på banen ved at passere startlinjen. Når bilen har passeret refleksbrik nummer 6, bringes bilen til standsning inden refleksbrik nummer 7 nås.	Visuel test: Bilen standser mellem refleksbrikkerne 6 og 7.	Stopper mellem 6 - 7 refleksbrikker	Godkendt
Punkt 3	Bilen bakker, indtil refleksbrik nummer 5 er passeret, og bringes til standsning inden refleksbrik nummer 4 nås.	Visuel test: Bilen bakker og standser mellem refleksbrikkerne 4 og 5.	Standser mellem 4 og 5	Godkendt
Punkt 4 + punkt 5	Bilen kører forlæns, indtil refleksbrik nummer 7 nås. Bilens bringes til standsning i målområdet, der er defineret som området mellem banens slut-ende og en meter efter banens slut-ende. Ved standsning skal hele bilen være placeret i målområdet.	Visuel test: Bilen standser indenfor målområdet	Standser ligeefter måleområde	Godkendt

Use Case 2: "Afspil lyd"	Test	Forventet resultat	Resultat
Punkt 1.	Use Case 1 "Kør banen" udføres.	Akustisk test: Når bilen tændes for at køre ind på banen: Der afspilles en	Godkendt

		specifik "startlyd" eller "startmelodi".	
Punkt 2.	Use Case 1 "Kør banen" udføres.	Akustisk test: Hver gang en refleksbrik passeres, afspilles en specifik "refleksbriklyd".	Godkendt
Punkt 3.	Use Case 1 "Kør banen" udføres.	Akustisk test: Når refleksbrik nummer 7 passeres, afspilles en specifik "slutlyd" eller "slutmelodi".	Godkendt

Use Case 5: "Kørsel"	Test	Forventet resultat	Resultat	Godkendt/kommentar
Punkt 1.	Use Case 1 "Kør banen" udføres. Under kørsel påvirkes SW2 på Arduino shield'et. Bilen placeres ved startlinjen og SW2 påvirkes igen.	Ved påvirkning af SW2 under kørsel stopper bilen. Når SW2 påvirkes igen kører bilen jf. Use Case 1.	Under kørsel og man trykker på SW2 stopper bilen og er klar til starte igen.	Godkendt
Punkt 2.	Use Case 1 "Kør banen" udføres. Under kørsel påvirkes SW3 på Arduino shield'et. Bilen placeres ved startlinjen og SW2 påvirkes. Herefter påvirkes reset-knappen og efterfølgende SW2 igen.	Ved påvirkning af SW3 under kørsel stopper bilen og slukker lys og lyd. Ved påvirkning af SW2 sker intet, men ved påvirkning af reset-knappen og herefter SW2 kører bilen jf. Use Case 1.	Under kørsel og man trykker på SW3 stopper lys, lyd og bilen og man kan ikke trykke på SW2 for at starte igen. Man skal trykke på reset knappen.	Godkendt
Punkt 3.	Use Case 1 "Kør banen" udføres. Efter endt kørsel placeres bilen ved startlinjen og SW2 påvirkes igen.	Når bilen efter første kørsel igen placeres ved startlinjen og SW2 påvirkes, kører bilen igen jf. Use Case 1.	SW2 gør det muligt at starte en ny prøvetur	Godkendt

Ikke-funktionelle krav

Krav nr.	Krav	Test	Forventet resultat	Resultat	Godkendt/kommentar

1.1.	Bilen skal styres, så den ændrer sin hastighed under gennemkørsel af banen.	Use case 1 "Kør banen" udføres.	Visuel test: Bilen sænker farten inden den når til bakken. Når bilen er på den anden side af bakken, øger bilen farten igen.	Bilen skifter hastighed adskillige gange under gennemkørslen, bl.a på vej ned ad bakken og for at 'bremse' når den skifter retning.	Godkendt
1.2.	Bilen skal på en enkelt opladning af dennes batterier kunne gennemføre mindst 5 gennemkørsler af banen.	Use case 1 "Kør banen" udføres fem gange.	Visuel test: Bilen gennemfører use case 1 fem gange før batteriet løber tør.	Efter 5 kørsler har den stadigvæk batteri.	Godkendt
1.3	På bilens højre og venstre side skal placeres detektorer, der kan registrere en R80 refleksbrik i afstanden 2 cm til 25 cm	Ved at benytte bilens refleksbriklyd, testes om bilen kan detektere en R80 refleksbrik i en afstand af 2cm og efterfølgende i en afstand af 25cm.	Akustisk test: Bilens højttaler afspiller den lyd der er tilknyttet denne detektor ved aktivering, når refleksbrikken er i en afstand mellem 2cm og 25cm af detektoren.	Ved både en test af refleksbrik på en afstand af ca. 2 cm og 25cm afspillede refleksbriklyden.	Godkendt
1.4.	Bilen monteret med al udstyr må maksimalt veje 5 kg	Bilen monteret med al udstyr vejes på en vægt med en målenøjagtighed bedre end 100 gram.	Vægten viser en vægt mindre end 5 kg.	4,95 kg	Godkendt
1.5.	Bilens maksimale højde skal være 41 cm.	Måling af bilens højde med målebånd	Bilen måler under 41 cm i højden.	33 cm høj	Godkendt

2.1.	Forlys implementeres med 2 hvide LED-sæt, der monteres med et sæt i henholdsvis højre og venstre side.	Bilens forlys kontrolleres visuelt før use case 1 udføres.	Visuel test: Bilen overholder kravet om påmonterede LED-sæt	Synligt lys kom ud fra både forlys og baglys fra ca. 1-2m afstand	Godkendt
2.2.	Når forlyset er tændt, skal hvert LED-sæt lyse svarende til én LED med middelstrøm men 50 mA +/- 5 mA.	Ved måling af LED sæt vha. Multimeter viser det en middelstrøm på 50 mA og +/-5 mA	Multimeter viser en værdi på 50 mA +/- 5 mA	Der er blevet målt på 2 LED-sæt, altså begge forlygter. I=105,3mA. Dette halveres, for at få strømmen for et LED-sæt. $I=52,65mA=$	Godkendt
3.1.	Bag- og bremselys implementeres med 2 røde LED-sæt, der monteres med et sæt i henholdsvis højre og venstre side.	Bilens bag- og bremselys kontrolleres visuelt før use case 1 udføres.	Visuel belysning kan ses fra 2 m afstand		Godkendt
3.2.	Ved "bremselys" skal hvert LED-sæt lyse svarende til én LED med middelstrøm men 50 mA +/- 5 mA.	Ved måling af LED sæt vha. Multimeter viser det en middelstrøm på 50 mA og +/-5 mA	Multimeter viser en værdi på 50 mA +/- 5 mA	Der er blevet målt på 2 LED-sæt . I= 97,5mA. Dette halveres, for at få strømmen for et LED-sæt. $I=48,75mA$	Godkendt
3.3.	Ved "almindeligt baglys" skal hvert LED-sæt lyse svarende til én LED med middelstrøm men	Ved måling af LED sæt vha. Multimeter viser det en middelstrøm på 10 mA og +/-1 mA	Multimeter viser en værdi på 10 mA +/- 2 mA	Der er blevet målt på 2 LED-sæt . I=23,6mA. Dette halveres, for at få strømmen for et LED-sæt. $I=11,8$	Godkendt

	10 mA +/- 2 mA.				
--	-----------------	--	--	--	--

Konklusion

Hardware konklusion

Hardware delen har klart været den del vi har haft størst problemer med, hvilket bekræfter fordommen om, at der ofte er langt fra teori til praksis. Da vi endelig havde loddet vores fejltæller modul færdig, kæmpede vi i lang tid med kortslutninger og fejlende dele, hvilket var meget frustrerende. Vi har haft problemer med at motormodulet kunne trække for meget strøm, hvilket brændte vores sikringer af. Da vi brugte en af vores lyssensor moduler som mellemlid for GND til baglys, introducerede vi så meget støj til sensoren, at den gav falske udslag. Alle disse komplikationer resulterede i mange timers frustration og fejlfinding.

Efter test af hardwaresystemerne, både ved individuel test og ved samlet modul test på bilen, er vores krav blevet opfyldt. Vi har lært meget om fejlfinding og løsning, og at intet er så nemt som diagrammerne umiddelbart siger.

Software konklusion

Softwaren er overordnet bygget op omkring de individuelle HW-moduler. Dette har vist sig meget nyttigt for debugging, strukturering og overskuelighed. På trods af grundige overvejelser omkring softwarens opbygning forud for implementeringen heraf, er det svært at lave fejlfri kode uden praktiske tests. Derfor har det været kritisk at have velfungerende hardware tidligt i forløbet, for at muliggøre praktiske tests, således softwaren kan tilpasses eventuelle uforudsete problemer.

Da softwaren i projektets forløb er skrevet delvist af forskellige personer, har det været højst nødvendigt, at have en effektiv fildelingsmekanisme af source-koden. Til udfærdigelse af projektet er benyttet GitHub, som dog har givet sin egen del af udfordringerne. Det har vist vigtigheden af, at have et system på plads fra starten, som alle involverede parter er indforståede med og kan anvende.

Overordnet konklusion

Bilens er opbygget af 4 hovedkomponenter; motor, lys, lyd og sensorer. Hver komponent i sig selv er et komplekst sammenspil mellem hardware og software, der skal snakke ordentlig sammen, før alle komponenterne endegyldigt kan kommunikere sammen under kørsel. For at skabe en velfungerende bil, opdeles diverse opgaver, som først løses individuelt i teams og derefter laves der en sammenfletning på bilen. De opgaver der er blevet tildelt, har bestået af en hardware og software del. Det har været essentielt, at de samme personer arbejder på både software- og hardwaredele, for at få den bedst mulige forståelse for løsningen af opgaven. Efter optimering af bilen, opfylder projektet alle de stillede krav.

Individuelle konklusioner

Rasmus

Efter projekt 1 har jeg fået en faglig forståelse for sammensætningen af de forskellige moduler på bilen og specifikt for mig, lydmodulet, der var mit primære område. For at opnå dette faglig udbytte var vi blevet inddelt i forudbestemte grupper baseret på den Insights test, vi tog i starten af semestret. Dette har betydet, at vi er blevet en gruppe med vidt forskellige personligheder og

baggrunde, som har formået at få et godt kammeratskab både socialt men også fagligt. Dette er sket på baggrund af sociale arrangementer og en god samlet indstilling til arbejdet på projektet.

Personligt syntes jeg, at projektet har været et godt eksempel på, hvordan man sammen med nye mennesker fokuseret arbejder sammen om et samlet projekt, men med klare fordelte roller og arbejdsopgaver

Gustav

Samarbejdet i gruppen er generelt gået meget godt. Vi har alle meget forskellige kompetencer og interesser, hvilket har givet os en bred vidensbank til projektets mange udfordringer. Det har været nemt at løse de konflikter vi har haft, ved at uddele opgaver og have fornuftige møder, hvor alle har kunne lufte sine meninger om projektet. Også socialt er det gået godt, da vi har forsøgt at lave aktiviteter uden for projektet såsom oplæsning til andre fag, madaften og tage på bar. Man har tydeligt kunne mærke vores brede fordeling af Insights-profiler, hvilket kun har bidraget til det brede og gode samarbejde i gruppen. Jeg har personligt fået rigtig meget ud af arbejdet, og føler min forståelse for teknologierne vi har arbejdet med, er blevet meget større.

Shynthavi

Projektet har som helhed medført, at vi i gruppen har kunne samarbejde og lære hinandens evner bedre at kende. Fagligt har projektet været med til, at de enkelte undergrupper har kunne arbejde dybdegående med ét emne for både software og hardware. Dette har givet en bedre forståelse af, hvordan software og hardware hænger sammen i et projekt, når et produkt bliver udviklet af en gruppe. Socialt har vi kunne fungere godt som gruppe, da vi har mødtes udenfor skolen for at hygge. Vejledningsmøderne har været hjælpsomme, da det har været med til at afklare alle de spørgsmål, der er dukket op undervejs. Alt i alt har dette projekt været meget lærerigt men samtidig også hyggeligt.

Simon

Gruppen har været god til at kommunikere, hvornår folk ikke kan møde op til forskellige møder. Til møderne var alle aktivt deltagende. Alle har haft forskellige præferencer i hvad de bedst kan lide at arbejde med og resulterede i god arbejdsfordeling. Der har været i gruppen en god blanding af sociale aktiviteter sammen som hjalp gruppen med at alle kan kommunikere med hinanden. Jeg har været tilfreds med arbejdsfordelingen gennem projektet, da alle fik lavet næsten lige meget arbejde i øvelserne og selve projektet. Da vores forskellige insights profiler dækkede et bredt område var der ikke store konflikter med fordelingen af arbejdet. Jeg har selv været god til at gå i gang med det praktiske når vi ved hvad for nogle opgaver, og har haft problemer med at komme i gang med arbejde når jeg ikke ved hvad jeg skal. Dette er typisk for en der blå i insightprofil som jeg er.

Anders

Projektet har været et godt supplement til den daglige undervisning, idet det har givet mulighed for at arbejde målrettet med de værktøjer og metoder man har lært gennem semesteret. Muligheden for til dels selv at forme opgaven og tilhørende løsninger, skaber en mere fokuseret og kreativ arbejdsproces som udspringer fra en reel interesse i projektet. Der er ofte opstået problemstillinger, som har krævet at man sætter sig grundigt ind i et emne eller en komponent, hvilket resulterer i en dybere forståelse end man havde opnået hvis det praktiske element ikke havde været til stede. Personligt har min største faglige læring vedrørt software og arbejde med mikrokontrolleren, da mine primære arbejdsopgaver har ligget her. HW har i mit projektforløb været sekundært men ikke ubetydelig.

Forløbet har også været en god læring i samarbejde og projektstyring. Eksempelvis er det tydeligt kommet til udtryk, når der har manglet organisering, hvilket indimellem har resulteret i ufokuseret

og usammenhængende arbejde, eller manglende overblik over projektets forløb. Generelt set har projektet dog forløbet jævnt, hvilket har vist sig ved at tidsplanen ikke er skredet. Det skyldes efter min mening en generelt god arbejdsmoral fra alle parter frem for god organisering.

Andreas

Projektet har lagt godt op til gruppe arbejde. Der er blevet lavet gruppe arbejde, både opdelt og samlet, hvilket har fungeret rigtig fint, da det har givet diverse kompetencer plads. Ved samlede møder, er der blevet lavet opdateringer og videre formidlinger, af hvad de individuelle grupper har lavet, både ved vejledning og løbende i gruppearbejdet. Projektet og grupperingerne samlede sig i udviklingen, i takt med at bilen blev samlet, og rapporten er blevet skrevet. Selve projektet har da udviklede sig meget organisk. Der har været hjælp kryds og tværs mellem grupperne, hvilket har været meget fint, da der har været plads til hjælp og ekstra øjne. Der har altså ikke været en, der ville dominere gruppen. De diverse "insight" profiler, har gjort at vi har fået en bred vifte af kompetencer. Dog har testen ikke været en dominerende faktor. Jeg har personligt kun lagt mærke til de forskellige profiler, når jeg er blevet mindet om det. Aktiviteter uden for projektet, har også fundet sted, hvilket har været friske pust og gjort gruppen mere samlet som en enhed.

Camilla

Dette projekt har været en god sammenfletning af de mange fag vi gennem 1. semester har haft. Hvert fag har været brugbart under udarbejdningen af dette projekt. Dette har hjulpet med at holde en god motivation for hvert fag, da man endegyldigt har haft mulighed for at anvende den viden man gennem semesteret har tillært sig. Derudover har projektet givet rig mulighed for et godt samarbejde og hjulpet til at få en god opstart da man gennem projektet har fået en gruppe man er tæt knyttet med gennem hele semesteret. i forhold til det faglige har jeg personligt fået et stort indblik i, hvordan software og hardware generelt kommunikerer samt. Specielt da jeg hovedsageligt har beskæftiget mig med lyd og herigennem har jeg lært meget om, hvordan en besked til et hardware komponent, gennem software, skal opbygges, for at det er forståeligt for komponenten. Alt i alt synes jeg projektet er gået rigtig godt, jeg har været heldig med min gruppe, som på baggrund af insight-profiler er sammensat, da vi hver er kommet med forskellige kompetencer og derved har kunnet bidrage med forskellige ting. Dette har skabt en rigtig god gruppedynamik, hvor alle har kunnet være med.

Billagsoversigt

1. Source kode
2. Litteratur
3. Mødereferater, Tidsplan
4. Visiofiler til BDD og IBD og diverse andre
5. Hardware diagrammer(Multisim filer)

Litteraturliste

Automation, T., n.d. Miniature PCB Relays.

BJT547B transistor datasheet.pdf, n.d.

Module, E.A., 2019. Embedded Audio-Sound Module 1–18.

Only, P.L., n.d. Lite-on Technology Corporation Lite-on Technology Corporation 1–8.

Pcr, A., Kit, L., 2012. Data Sheet Data Sheet 0–1.

PrtMosFet, I., n.d. InternationalRectifier: IRLZ44NPbF, FHEXFET Power MOSFET. Manual.