Pokaljagten

## semester projekt

## Gruppe 7

|  |  |  |
| --- | --- | --- |
| Navn | Studienummer | Underskrift |
| Mads Gjerrild Søgaard | 201270777 |  |
| David-Samuel Buhauer | 201270749 |  |
| Mark Wilhardt Maansson | 201270994 |  |
| Kristoffer Sebastian Breuer | 201270746 |  |
| Kim Lykke Johansen | 201271092 |  |
| Apinayan Mohanathas | 11837 |  |
| Mathias Schlütter | 201270835 |  |
| Martin Østergaard | 201271031 |  |



# Resume

*Skrevet af Mark W. Maansson*

I dette projekt har vi haft til opgave at programmere og lave ekstra hardware til en bil. Vi har været 8 mand fra AU Ingeniørhøjskolen om projekt. Vi har haft omkring et helt semester til at få kendskab og skabe erfaringer nok, fra 5 forskellige fag og til slut at ende ud med at samle det hele i dette projekt, med hjælp fra en vejleder. Vi har kæmpet hård. I projektet her kommer vi ind på bla. programmet motor styring ved hjælp af timer og PWM signal, programmering af lyd som er forskellige lyde fra Mario. Vi kommer også ind på programmering af lys, både for- og baglys, og nogle lys effekter, plus refleks detektorer, med tilhørende Hardware. Det hele skal så ende ud i at vores færdige projekt skal ud og dyste i den store pokaljagt.

# Abstract

*Skrevet af Mathias Schlütter*

In this project our task has been to code and create extra hardware to a car. We have been 8 guys from Aarhus university school of engineering on this project. We have had approximately 1 semester to acquire knowledge and gain enough experience from 5 separate classes, and at the end pull them all together into this project, with the help of a guidance counselor. We have fought hard. In this project we will among other things, include the code for engine control with the help of timers and PWM signals, coding of sound which is multiple different sounds from the Mario theme. We also use coding of light both front and rear and some light effects, and reflection detectors. All the following code will have following hardware produced and tested. At the end it will all culminate in a great “trophy” hunt, which is a race between the cars made by all the project groups.

Indhold

[Resume 1](#_Toc346225890)

[Abstract 1](#_Toc346225891)

[Indledning 3](#_Toc346225892)

[Systembeskrivelse 5](#_Toc346225893)

[Kravspecifikationer 6](#_Toc346225894)

[Strukturering/design 8](#_Toc346225895)

[Hardware 8](#_Toc346225896)

[Blok beskrivelse 9](#_Toc346225897)

[Signaldiagram 11](#_Toc346225898)

[Signal beskrivelse 12](#_Toc346225899)

[Software 14](#_Toc346225900)

[Klassediagram 14](#_Toc346225901)

[Klassebeskrivelse 15](#_Toc346225902)

[Implementering 19](#_Toc346225903)

[Hardware 19](#_Toc346225904)

[Lys 19](#_Toc346225905)

[H-bro og motorregulering 22](#_Toc346225906)

[Tæller 25](#_Toc346225907)

[Knight rider funktionen: 29](#_Toc346225908)

[Refleksions detektor 30](#_Toc346225909)

[Software 30](#_Toc346225910)

[Lys 30](#_Toc346225911)

[Lyd 34](#_Toc346225912)

[Motor 39](#_Toc346225913)

[Refleksions detektor 42](#_Toc346225914)

[Bluetooth 45](#_Toc346225915)

[Main.C 46](#_Toc346225916)

[Knight rider funktionen: 48](#_Toc346225917)

[Accepttest 50](#_Toc346225918)

[Accepttest specifikationer 50](#_Toc346225919)

[Konklusion 55](#_Toc346225920)

[Fælles konklusion 55](#_Toc346225921)

[Delkonklusioner 56](#_Toc346225922)

[Litteraturliste 59](#_Toc346225923)

# Indledning

Pokaljagten er en dyst, hvor en elektrisk bil skal gennemføre en bane med forhindringer på kortest mulig tid. Bilen er udstyret med en accelerationssensor (pendul), der kan registrere accelerationer over en fast grænse. Overskridelse af denne grænse udløser strafpoint. Forhindringerne er udformet som ramper bilen skal passere. Det vanskelige ved gennemkørslen er at tilpasse bilens acceleration til banens forskellige dele. Det er kun bilens fart der kan reguleres. Banen har bander der sørger for bilens retning, og banderne er forsynet med refleksbrikker til positionsbestemmelse.

Følgende regler gælder for gennemkørslen af banen:



Figur 1 Køre banen

Point gives ud fra forskellige kriterier:

* Point er lig med gennemførselstiden i sekunder.
* Der gives 10 bonus point ved udnytte et pitstop ved refleksbrik # 6.
* Der gives yderligere 20 bonuspoint hvis bilen kan bakke fra refleksbrik #6 til #5.
* Alarmtællerens visning tillægges som strafpoint.
* Bilen diskvalificeres hvis den kører længere end 1,5 m efter mål.

Banen er udstyret med en kontakt ved banens start og en lysbarriere ved banens mål.

Gennemførselstiden måles fra lysbarrieren ved start til lysbarrieren ved slut.

Efter mål **skal** bilen holde stille i målområdet (max 1,5 m efter lysbarrieren ved slut). Hvis bilen kører længere end målområdet, diskvalificeres kørslen.

Det samlede pointtal udregnes ved:

Laveste antal point vinder.

Pokaljagten er knyttet til det tværfaglige projekt PRJA, der gennemføres på 1. semester. I dette projekt fremstilles hardware og software til bilen. Dette er specificeret i afsnittet kravspecifikation. Bilen skal udstyres med:

* For- og baglygter
* Mikrocontroller
* Elektronik til regulering af motor
* Sensorer til detektion af refleksbrikker
* Tæller der registrerer hvor mange gange accelerationen er for stor

De øvrige kurser på 1. semester bidrager hver i sær med bl.a. alarmtæller, SW til regulering af PWM-signal og detektorer til refleksbrikkerne.

# Systembeskrivelse

*Skrevet af Mathias Schlütter*

Vores bil er udstyret med 2 STK500-kit, et første styrer vores bluetooth enhed, samt en del af lyden. Det andet STK500-kit er programmeret til at kontrollerer motoren, lyset, refleksionsdetektoren samt den resterende lyd.

Vi har udstyret bilen med hvidt kørelys, bagudrettet kørelys i rødt. Der er til lyset programmeret styring af bremselys (kraftigt rødt). Som tillæg er der foran på bilen monteret ”Night Rider” lys i rødt (bølge pulserende lys).

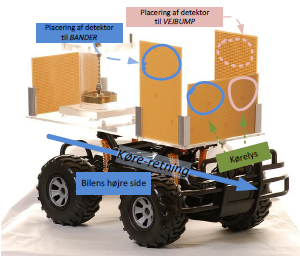
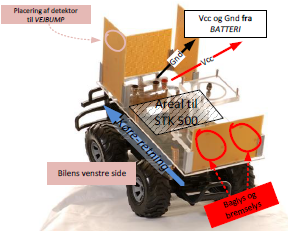
Bilen er udstyret med en refleksionsdetektor, som er programmeret til at opfange de refleksbrikker som er opstillet på banen, de bliver endvidere brugt til navigation at bilen. Når refleksdetektoren opfanger en brik af gives en coin lyd. Lyden spilles fra 2 separate højtalere, der er monteret på vores STK500-kit, og vil give lyd alt efter hvor på banen bilen befinder sig, samt start og afslutnings melodier.

Bilens motor er styret af vores ene STK500-kit, og der er blevet lavet en H-bro til motoren, så det er muligt at kunne køre både frem og tilbage. Motorens ”kraft” kommer fra 1x 9,6V og 1x7,2V batterier, disse giver også strøm til STK500-kittene.

På bilen er et preinstalleret pendul som skal til fejltælling, til dette er der lavet en fejltæller, som øger mængden af fejl hver gang pendulet berører kanten af klokken som denne befinder sig i.

# Kravspecifikationer

1. Generelle krav
   1. Middelværdien af forsyningsspændingen til bilens motor skal styres via pulsbreddemodulation (PWM) fra 0 til maksimal batterispænding i 256 trin.
   2. På bilens højre og venstre side skal placeres detektorer, der kan registrere en R80 refleksbrik i en afstand (2 cm til 25 cm) +/- 1 cm. Se Figur 2 og Figur 3.
   3. Print til detektorer skal have en bredde på 80 mm +/- 1 mm. Se Figur 2 og Figur 3.
   4. Bilen skal kunne afgive forskellige lydsignaler ved start, stop og detektion af refleksbrik – samt afgive lydsignal når bilen bakker.
   5. Bilen skal kunne udføre bak sekvensen for bonus point.
   6. Bilen skal kunne betjenes via radio fjernbetjening, rækkevidde minimum 25 m i frit rum(på nye batterier(hvis batteridrevet model)).
2. Bilens lys
   1. Fremadrettet kørelys
      1. Implementeres med 8 hvide LED, der monteres 4 i højre og 4 i venstre side, på bilens front. Se Figur 2 – LTW-2S3D7 eller kompatible.
      2. Kørelyset må kun være tændt når bilens motor påtrykkes en middelspænding > 0Volt.
      3. Når det hvide lys er tændt skal middelstrømmen gennem hver LED være 50 mA +/- 1 mA.
   2. Bagudrettet køre- og bremselys
      1. Implementeres med 8 røde LED, der monteres 4 i højre og 4 i venstre side, på bilens bagende. Se Figur 3 – LH3300 eller kompatible.
      2. Når middelspændingen til biles motor mindskes, skal middelstrømmen gennem hver bagudrettet LED være 50 mA +/- 1 mA i en periode på 2,5 s +/- 0,5 s.
      3. Når middelspændingen til biles motor øges eller er konstant, skal middelstrømmen gennem hver bagudrettet LED være 10 mA +/- 1 mA. Krav 2.2.2. har højere prioritet end krav 2.2.3.
   3. Print til fremadrettet og bagudrettet kørelys skal have en bredde på 160 mm +/- 1 mm.
   4. Belysning af undervogn: På undervognen af bilen skal der sidde belysning, som er nedadrettet. Denne belysning skal være grøn.
   5. Knight rider belysning skal monteres lige under forlygterne. Det skal bestå af 8 grønne LED på række.
   6. Når Knight rider er tænkt skal middelstrømmen gennem hver LED være 50mA +/- 1 mA.
   7. Knight rider må først starte når bilen køre og slukke når bilen slukker
3. Bilens lyde
   1. Lyd ved start: Lyden ved start skal være genkendelig version af: Start lyden Super Mario
   2. Lyd ved stop: Lyden ved stop skal være genkendelig version af: Level complete fra Super Mario
   3. Lyd ved dektektion: Lyden ved detektion skal være en genkendelig version af: Coin lyd fra Super Mario
   4. Lyd når bilen bakker: Lyden når bilen bakker, skal være genkendelig version af: en lastbil som bakker
4. Bilens kørsel
   1. Fremadrettet kørsel
      1. Bilen skal kunne accelerere hurtigst muligt, i en jævn acceleration, så der registreres så få fejl som mulig, på fejltælleren med pendullet på.
   2. Bagudrettet kørsel
      1. Bilen skal være i stand til at stoppe, bakke og køre frem ad igen.
   3. Bilen skal være i stand til at stoppe inden for 1,5 m målområde, i en jævn opbremsning der sikre et minimum af straf point.
   4. Belysning af undervogn: På undervognen af bilen skal der sidde belysning, som er nedadrettet. Denne belysning skal være grøn.
5. Ydereligere krav
   1. Fjernbetjening
      1. Med fjernbetjeningen skal bilen kunne startes, således at programmet der driver bilen op startes.
      2. Med samme fjernbetjening skal brugeren være i stand til at nulstille, tælleren der tæller pendul fejl ved ujævn kørsel.
   2. Pendul til fejltælling.
      1. På bilen skal der monteres et pendul, som kan detekterer ujævn kørsels.
      2. Pendulet skal være forbundet til et digitalt display, som tæller hvor mange udslag der kommer under kørsel, minimum 5 udslag/sekund + ½ udslag/sekund.
      3. Pendultælleren skal have en reset-knap, som nulstiller det digitale display.(Udføres evt. som element på fjernbetjeningen).
      4. Reset funktionen må ikke være tilgængelig under kørsel.



Figur 2 - Placering af kørelys og detektorer

Figur 3 - Placering af dektektorer, baglys/bremselys og STK 500 kit

# C:\Users\Ejer\Desktop\Unavngivet.pngStrukturering/design

## Hardware

Figur 4 - Blok diagram af hardware

### 

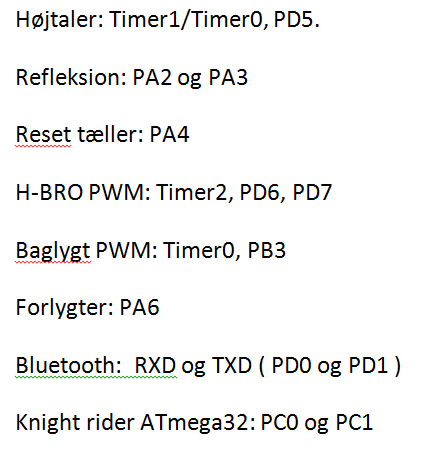
### Blok beskrivelse

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Blok navn | Funktions beskrivelse | Signaler | | Kommentar | |
| Batteri 7,2 V | Spændingskilde til motor | V\_batteri\_motor | | Spændingskilde til motor | |
| Motor | Drivkraft af bilen | V\_batteri\_motor | | Spændingskilde til motor | |
| PWM\_motor | | Indgangssignal til styrring af motor | |
| H-bro | Vending af strømmens retning. Formål: at gøre bilen i stand til at bakke | 0 V\_mega 32 | | Reference fra mega 32 | |
| V\_strømretning | | Strømretningssignal fra mega 32 | |
| PWM\_motor | | Ind/udgangssignal til styrring af motor | |
| 5 V\_ analog | | Strømforsyning | |
| 0 V\_analog | | Reference | |
| Batteri 9,6 V | Spændingskilde til system(for uden motor) | 9,6 V\_analog | | Spændingskilde til spændingsregulator og undervognsbelysning | |
| Spændingsregulator | Regulerer spændingen til analoge moduler | 9,6 V\_analog | | Spændingskilde | |
| 5 V\_ analog | | Udgangsspænding til H-bro | |
| 5 V\_ analog | | Udgangsspænding til refleksions detektor | |
| 5 V\_ analog | | Udgangsspænding til pendul | |
| 5 V\_ analog | | Udgangsspænding til tæller | |
| 5 V\_ analog | | Udgangsspændings til STK 500 | |
| 5 V\_ analog | | Udgangsspænding til forlygter | |
| 5 V\_ analog | | Udgangsspænding til baglygter | |
| 0 V\_analog | | Reference | |
| STK 500 / Mega 32 | Mega 32: µProcessor til software behandling, styring af funktioner.  STK 500: Kommunikations print mellem Mega 32 og anden hardware | PWM\_motor | | Udgangssignal PWM til styrring af motor | |
| PWM\_baglygt | | Udgangssignal PWM til styring af baglygter | |
| Refleksions detektor | | Indgangssignal fra refleksions detektor | |
| Lyd\_signal | | Udgangssignal til forskellige lydsignaler | |
| V\_reset | | Udgangssignal til reset af tæller | |
| Bluetooth\_reset | | Indgangssignal fra bluetooth COM-port til reset af tæller | |
| Bluetooth\_start | | Indgangssignal fra bluetooth COM-port til start af køresekvens | |
| V\_strømretning | | Udgangssignal til kontrol af strømmens retning igennem H-broen | |
| 5 V\_ analog | | Spændingskilde | |
| 0 V\_analog | | Reference | |
| Højtaler | Afspilning af lydsekvenser | Lyd\_signal | | Indgangssignal | |
| Lyd | | Eksternt udgangssignal | |
| Lys | Hovedblok til bilens lys |  | |  | |
| Front Lys | Fremadrettet kørerlys | 50 mA\_kørsel | | Strømforsyning | |
| 0 V\_analog | | Reference | |
| Undervogns belysning | Gejl | 9,6 V\_analog | | Spændingskilde | |
| 0 V analog | | Reference | |
| Bag lys | Bagudrettet kørerlys  Bremse lygter | 10 mA | | Strømforsyning v/ kørsel | |
| 50 mA\_brems | | Strømforsyning v/ brems | |
| 0 V\_analog | | Reference | |
| Knightrider lys | Gejl | Mega32 ved 5V | | Strømforsyning | |
| 0 V\_analog | | Reference | |
| 50 mA\_blink | | Strømforsyning | |
| Refleksionsdetektor | At detektere refleksbrikker og sende denne information til Mega 32 | 5 V\_ analog | | Strømforsyning | |
| 0 V\_analog | | Reference | |
| Detektion | | Detektions indgangssignal | |
| Refleksions detektor | | Udgangssignal | |
| Pendul | Registrere ujævn acceleration og sende information til tæller | 5 V\_ analog | | Strømforsyning | |
| 0 V\_analog | | Reference | |
| V\_registrering | | Udgangssignal | |
| Tæller | Sørge for at der hverken bliver talt for mange eller for få fejl, samt at vise dette på et display | 5 V\_ analog | | Strømforsyning | |
| 0 V\_analog | | Reference | |
| V\_registrering | | Indgangssignal | |
| Display | | Eksterngrænseflade, udgangssignal | |
| Fjernbetjening STK | Fjernbetjening af bilens system | Stikkontakt | | Strømforsyning/Reference | |
| Bluetooth | | Udgangssignal | |
| Bluetooth\_start | | Udgangssignal | |
| Bluetooth\_reset | | Udgangssignal | |
| Start knap | At starte bilens køresekvens | Bluetooth \_start | | Udgangssignal til bluetooth | |
| Reset knap | At resette tælleren | Bluetooth \_reset | | Udgangssignal til bluetooth | |
| Bluetooth | Kommunikations vej fra fjernbetjening til bilen | Fjernbetjening | Bil\_STK | Fjernbetjening | Bil\_STK |
| Bluetooth\_start | Bluetooth \_start | Udgangssignal | Indgangssignal |
| Bluetooth\_reset | Bluetooth \_reset | Udgangssignal | Indgangssignal |

### Signaldiagram

*Skrevet af David Buhauer*





Figur 5 - Signaldiagram for systemets hardwareblokke

### Signal beskrivelse

*Skrevet af Kim Lykke Johansen*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Navn | Funktion | Område | Terminal 1 | Terminal 2 | Terminal 3 |
| 0 V\_analog | Reference til analoge moduler |  | Stel  H-bro  Spændingsregulator  STK 500 / Mega 32  Front lys  Bag lys  Undervogns belysning  Refleksionsdetektor  Pendul  Tæller | Intern strømforsyning |  |
| 0 V\_mega 32 | Reference til H-bro fra Mega 32 |  | Stel | Intern strømforsyning |  |
| 10 mA | Strøm gennem baglygter ved kørsel | 10 mA +/- 1 mA | PWM\_baglygt  Timer 0:  Pin D2 og Pin B3 | Intern strømforsyning |  |
| 50 mA\_brems | Strøm gennem baglygter ved opbremsning | 50 mA +/- 1 mA | PWM\_baglygt  Timer 0:  Pin D2 og Pin B3 | Intern strømforsyning |  |
| 50 mA\_kørsel | Strøm gennem forlygter | 50 mA +/- 1 mA | Intern strømforsyning |  |  |
| 50 mA\_blink | Strøm gennem Knightrider lys | 50 mA +/- 1 mA | Intern strømforsyning |  |  |
| 5 V\_ analog | Spændingskilde til analoge moduler | 4,7-5,2 V | 5 V  H-bro  Refleksions detektor  Pendul  Tæller  STK 500  Forlygter  Baglygter | Intern strømforsyning |  |
| 9,6 V\_analog | Spændingskilde til spændingsregulator 🡪 analoge moduler | 5-9,6V | 9,6 V  Spændingsregulator | Intern strømforsyning |  |
| Bluetooth | Kommunikation mellem fjernbetjening og bilen | 2,45 GHz | COM port 2 på STK 500 der fungerer som fjernbetjening | COM port 1 på STK 500 der styre bilen |  |
| Bluetooth\_reset | Signal fra fjernbetjening til mega 32 på bilen om at reset tæller | Aktiv lav ≈ 0 V  1,8<VTG<6,0V  Bluetooth: | Swith 0 (SW0) på STK 500 fjernbetjening  Pin: A0 | COM port 2 på STK 500 der fungerer som fjernbetjening | COM port 1 på STK 500 der styre bilen |
| Bluetooth\_start | Start signal fra fjernbetjening til mega 32 | Aktiv lav ≈ 0 V  1,8<VTG<6,0V  Bluetooth: | Swith 1 (SW1) på STK 500 fjernbetjening  Pin: A1 | COM port 2 på STK 500 der fungerer som fjernbetjening | COM port 1 på STK 500 der styre bilen |
| Detektion | Detektion af refleksbrik | 400-1100nm |  | ?? |  |
| Display | Visualisering af tæller | 2,0-2,5V  30 mA DC | Tæller | Ekstern-grænseflade |  |
| Lyd | Audio afspilning af lydsekvenser | 100-250Hz | Mega 32  Pin: A0 og A1 | Ekstern-grænseflade |  |
| Lyd\_signal | Digitalt signal til højtaler, hvilke toner der skal afspilles | 0-5 V | Mega 32  Pin: A0 og A1 | Højtaler |  |
| PWM\_baglygt | Digital styring af lysstyrke på baglygter | 0-5 V | Mega 32  Pin: D2 og B3 | Baglygter  (2 stk. LED - LH3300) |  |
| PWM\_motor | Digital PWM styring af motor | 0-5 V | Mega 32  Timer 1:  Pin: D3, D4 og D5 | H-bro 🡪 Motor |  |
| Refleksions detektor | Signal til mega 32 om detektion af refleksbrik | 4,5-5V | Refleksions detektor | Mega 32  Pin: A2 og A3 |  |
| Stikkontakt | Strømforsyning til fjernbetjening | 230 V\_AC 🡪 10-15 V\_DC  500mA | Omformer | STK 500 fjernbetjening |  |
| V\_batteri\_  motor | Spændingskilde til motor | 5-7,2 V  3-10 A | 7,2 V  Motor | Intern spændingskilde |  |
| V\_registrering | Signal fra pendul til tæller om ujævn kørsel, skal trigger display count | 4,5-6V | Pendul | Tæller |  |
| V\_reset | Puls til reset af tæller | 4,2 – 5 V | Mega 32 | Tæller |  |
| V\_strømretning | At toggle strømmens ret igennem H-bro | 4,2 – 5 V | Mega 32 | H-bro |  |

## Software

### Klassediagram

*Skrevet af David Buhauer og Kristoffer Sebastian Breuer*

 Figur 6 – Klassediagram

### Klassebeskrivelse

*Skrevet af Mathias Schlütter og David Buhauer*

**Bluetooth**

**Ansvar**: Klassens ansvar er at starte kørselssekvensen samt nulstille fejltæller.

**Metode** : void StartSignal()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen starter bilen, nulstiller fejltæller og starter alt funktionelitet.

**Refleksion**

**Ansvar:** Give signal(lyd) ved opfattelse af refleksionsbrik, samt aktivere bakkemekanisme(i motor).

**Metode**: void ReflectionCount()

Parametre: Ingen.

Returværdi: Ingen

Beskrivelse: Funktionen skal detektere en R80 refleksbrik og tælle antallet ved at gemme det i en global tællervariabel af typen int kaldt antalRefleksbrikker.

**Lyd**

**Ansvar**: Afspille lyd ved start af program, samt afspille lyd ved Reflections funktion = true, afspille lyd ved bakkemekanisme, samt ved afslutning af program.

**Metode:** void StartMario()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen skal afspille en genkendelig version af melodien ”Super-Mario” hovedtemaet, når programmet starter, og bilen kører.

void StopLevelComplete()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen skal afspille en genkendelig version af melodien ”Level Complete” fra Super-Mario, når programmet slutter, og bilen er nået i mål.

void DetekCoin()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen skal afspille en genkendelig version af melodien ”Coin” fra Super-Mario, når der under bilens kørsel detekteres R80 refleksbrikker.

void ReverseBeebBeeb()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen skal afspille en genkendelig lyd fra en lastbil der bakker, når strømmens retning vendes, og bilen bakker.

**Lys**

**Ansvar:** Styring af frontlys og baglys samt forstærke baglyset ved opbremsning. All lys skal også slukkes ved afslutning af bane.

**Metode** : void AllLightsOn()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Tænder frontlyset og tænder bremselyset med 10% Duty Cycle

void AllLightsOff()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Slukker for frontlyset og slukker for bremselyset

void BrakeLightOn()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Forøger intensiteten af bremselyset(PWM 100%).

void BrakeLightOff()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Mindsker intensiteten af bremselyset(PWM 10%).

void InitFrontLEDPort()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Initierer porten for frontlyset.

void InitBackLEDPort()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Initierer porten for bremselyset og Timer0 registeret til PWM

void BrakeLightsNormal()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Tænder for bremselyset uden PWM styring.

void BrakeLightsNormal()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Tænder for bremselyset uden PWM styring.

void KnightRiderStart()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen starter et lysloop der minder om lyset fra Knight Rider temaet.

void KnightRiderSlut ()

Parametre: Ingen.

Returværdi: Ingen.

Beskrivelse: Funktionen slutter lysloopet, når bilen slukker.

**Motor**

**Ansvar:** Skal styre motoren med et PWM signal, hvor der kan ændres på farten og få bilen til at køre frem, bakke samt bremse.

**Metode**: void DriveForward()

Parametre: Ingen.

Returværdi: Ingen

Beskrivelse: Funktionen skal få bilen til at accelerere med et PWM signalet og give bilen en fart.

void Reverse()

Parametre: Ingen.

Returværdi: Ingen

Beskrivelse: Funktionen skal vende strømretningen vha. en H-BRO, så bilen kan bakke.

void Brake()

Parametre: Ingen.

Returværdi: Ingen

Beskrivelse: Funktionen skal ændre på PWM signalet sådan så bilen motor bremser.

# Implementering

## Hardware

### Lys

##### Forlys

*Skrevet af Mads Gjerrild Søgaard*

Forlyset er et kredsløb med en spændings regulator til at regulere strømmen fra batteriet(9,6V) til 5,0V, da det er vigtigt for at få en stabil strøm igennem systemet, fordi at en stabil strøm vil få hver LED til at lyse med den samme konstante effekt. Kredsløbet er bygget op af en NPN transistor(BD139) som bruges som en switch, fordi lyset først skal lyse når bilen startes, NPN transistoren virker som switch pga. der skal gå en lille strøm igennem basen før der går strøm fra emitter til collector, så de 8 hvide LED tænder først når basen får strøm. Der er 8 modstande af hver 120 Ohm som går ud til hver sin hvide LED.

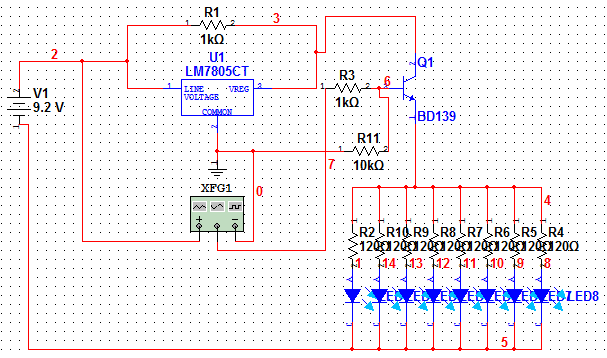
Der er målt strømmen igennem hver LED til 49,7 mA, som er inden for det acceptable 50 mA +/-1 mA, ud over dette er NPN transistoren brugt som switch og det er programmeret så det kun lyser når bilens motor påføres > 0 Volt. Det er testet visuelt ved, at når der ikke er strøm til bilens motor, er lyset slukket. Når bilens motor så påføres mere end 0 Volt, lyser bilens forlys.

Modstanden R10, som er modstanden før NPN transistoren, til at begrænse strømmen der kommer ind via basen er udregnet således:

Modstanden R10, er vigtig da vi ikke vil have en ændring i lysstyrken på forlyset og modstanden begrænser strømmen fra stk500 til at være minimal, og får NPN transistoren til at virke som en switch.

Der er brugt en pulldown modstand på 10k ohm, for at få forlyset til at lyse med en konstant intensitet, hvis pulldown modstanden ikke er på der, får det forlyset til at flimre.

I simulationen nedenunder har jeg brugt en funktions generator med et firkant signal, på 100Hz, til at få LED’erne til at lyse, fordi så kan det ses at når signalet er 0, så går der ikke strøm fra collector til emitter fordi basen er 0.



Figur 7 - Simulation af forlys

##### Baglys

*Skrevet af Mads Gjerrild Søgaard*

Ved vores baglys bruger vi en spændings regulator som regulerer strømmen fra vores batteri(9,6V) til 5,0V, for at få en stabil strøm igennem hele kredsløbet. Fra spændings regulatorens indgang(plus) og udgang er der en modstand på 1k Ohm. Udgangen går ud i en NPN transistors base og i NPN transistorens collector har vi et PWM signal genereret af vores mega32, i NPN transistorens emitter er der udgang til 8 modstande af 27 Ohm, de går hver op til en rød LED og derefter til stel.

PWM signalet får baglyset til at også fungere som bremselys, idet bilen bremser, vil de lyse kraftigere, pga. NPN transistoren får et signal fra vores mega32, som fortæller den til at sende mere strøm igennem og får derved vores 8 stk. LED til at lyse kraftigere.

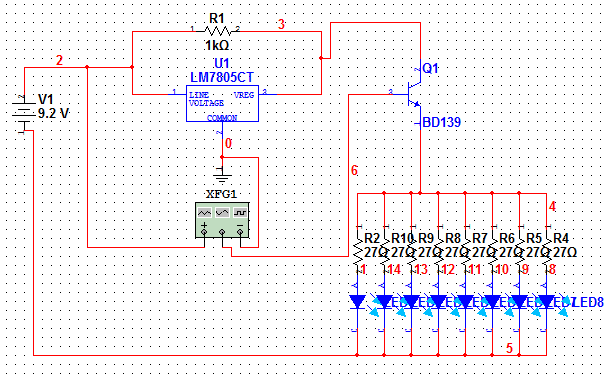
Grunden til at der er brugt 27 Ohms modstande i dette kredsløb frem for de 120 ohms modstande som er blevet brugt til forlyset, er fordi at PWM signalet fra vores mega32 er så svagt, at LED’erne næsten ikke lyste ved større modstande.

Middelstrømmen gennem hver LED, når middelspændingen til bilens motor mindskes, er på 49,8 mA som er inden for de 50 mA +/- 1 mA i en periode af 2,5 s +/- 0,5 s.

Når bilens middelspænding øges, falder middelstrømmen til hver LED til 10 mA +/- 1 mA.

For at få PWM styringen til at virke korrekt spiller det en stor rolle at stk500 og kredsløbet har en fælles GND, for at få signalet til at gå igennem.

Bremselyset kunne også have været løst ved at bruge forskellige modstande, ud over de 8 modstande af 27 Ohm, ville der være blevet brugt 8 mindre modstand, og via et simpelt signal fra vores mega32 når bilen bremser, få systemet til at bruge de mindre modstande, det vil altså virke som en switch og få lyset til at lyse kraftigere når bilen bremser.



Figur 8 - simulation af baglys med PWM signal

##### Delkonklusion for belysning på bilen:

*Skrevet af Mads Gjerrild Søgaard*

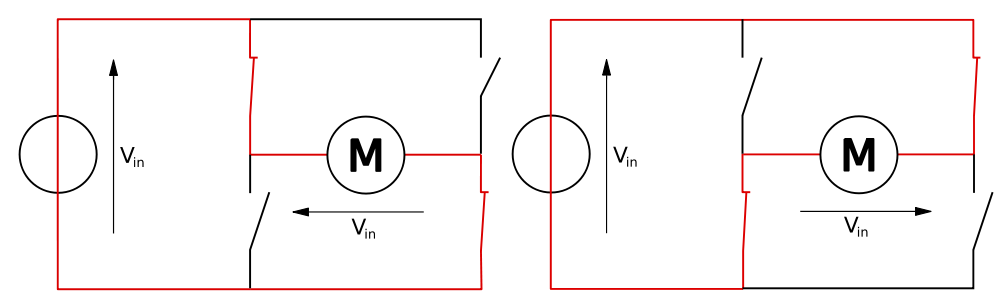
For bilens baglys var det vigtig at få testet i et tidligt stadie, da modstandende foran hver LED skulle være meget lavere end forventet pga. PWM signalet fra stk500 var meget lav, derfor havde det stor betydning for os at vi testet det hele igennem da vi startede med modstande på 120 Ohm og efter gennemtest fandt vi ud af, at ved 27 ohm ramte vi den strøm på lige omkring 50 mA. Beslutningen med at lave baglyset via PWM styring, var fordi det virkede som den mest effektive løsning, da PWM styring var lettere end at lave det som hardware, fordi det er styret via en NPN transistor.

### H-bro og motorregulering

*Skrevet af Apinayan Mohanathas*

Gruppen har besluttet at gå efter de 20 bonuspoints der i at få bilen til at bakke fra refleksbrik #6 til #5. Der skal derfor bygges noget elektronik til at vende strømretningen gennem bilens motor når vi ønsker at bakke bilen.

En H-bro er et kredsløb der gør det muligt at vende strømretningen igennem en motor. En ændring af strømretningen gør at retningen på spændingen over motoren også skifter hvilket resulterer i at motoren kører den anden vej. H-broens principelle virkemåde er illustreret herunder:



Figur 9 - Grafik fra Wikipedia

Der er flere måder man kan realisere et sådant kredsløb på. H-broer findes som IC’er, relæer og kan bygges med transistorer, men IC’er og transistorer kan være sårbare over for de store strømme (5-8A) vi kom frem til at der gik gennem motoren når den udsættes for så store belastning som i Øvelse 2.

Løsningen som vi har brugt er et relæ. Relæet vi bruger er **Finder Type 40.52** og har dens spole mellem A1 og A2 og kontakterne 12-11-14 og 22-21-24:



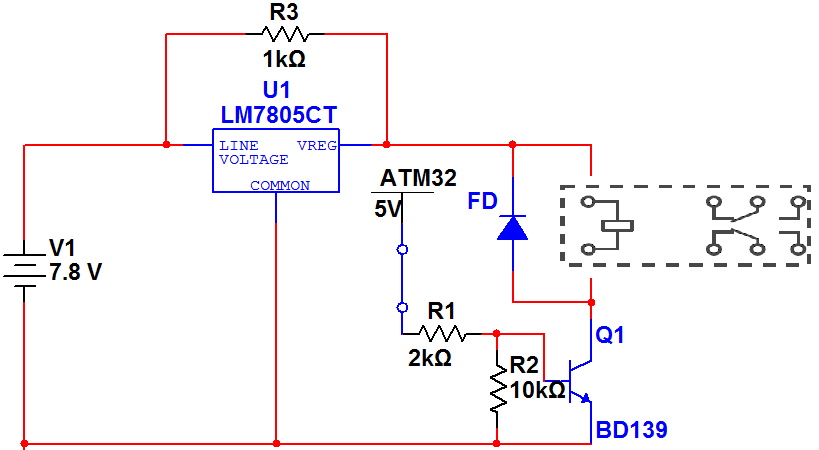
Figur 10: Fra Finder Type 40.52 datasheet

For at relæet skal omskifte skal der en spænding på mindst 5V mellem A1 og A2. Microcontrolleren (**ATMega32**) kan som udgangspunkt godt levere denne spænding, men relæet skal bruge *mindst* 60 mA for at omskifte og det ligger over de anbefalede grænser for hvor meget microcontrolleren kan trække uden at risikere skade.

Derfor er det nødvendigt at bruge en transistor til at påføre relæet en spænding til at omskifte. Transistoren der blev valgt er en BJT NPN transistor af modellen **BD139**. For at få en 5V spænding fra 7,8V batteriet bruges en spændingsregulator af modellen **LM7805**.

Spolen i relæet (mellem A1 og A2) er en induktiv belasting og det er derfor nødvendigt med en diode i parallel med A1 og A2 for at beskytte kredsløbet mod pludselige spændingsændringer.

I diagrammet nedenunder vises hvordan relæets spole er forbundet til batteriet og microcontrolleren:



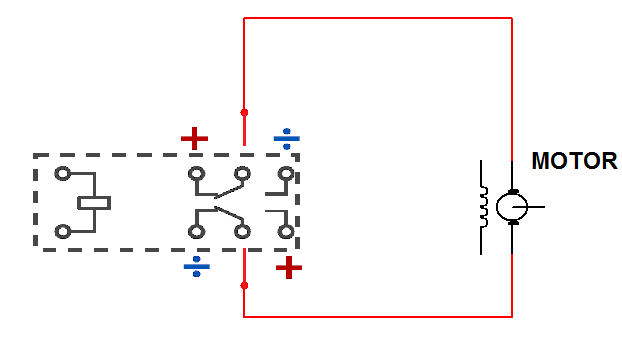
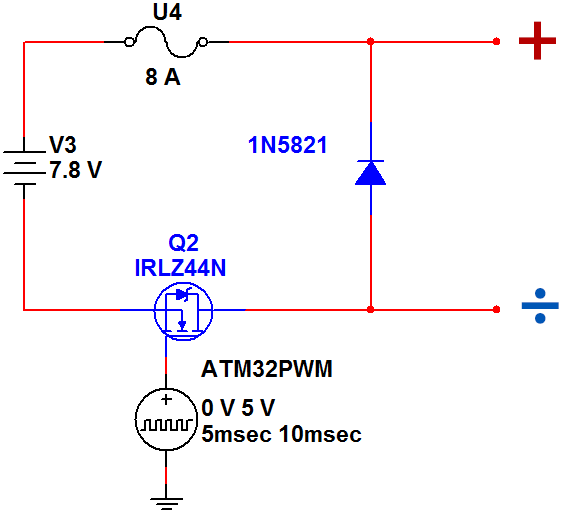
Figur 11: Diagram hvor kun relæspolens relation til batteriet og microcontrolleren kan ses

Modstanden R2 er en pull down modstand på transistorens base. Base modstanden R1s størrelse er fremkommet på følgende måde (BD139 har beta på 40 og relæspolen vælges til at trække ca. 80-85 mA):

Der vælges en lidt mindre modstandsværdi så R1 er derfor 2k.

Nedenunder vises hvordan relæets kontakter er forbundet med motorreguleringskredsløbet som består af MOSFET transistoren IRLZ44N som tænder/slukker motoren (hurtigt) i henhold til PWM-signalet og dermed regulerer motorens hastighed. Der sidder også en sikring og en diode der beskytter kredsløbet mod henholdsvis alt for store strømme og de pludselige spændingsændringer som motoren kan være skyld i.

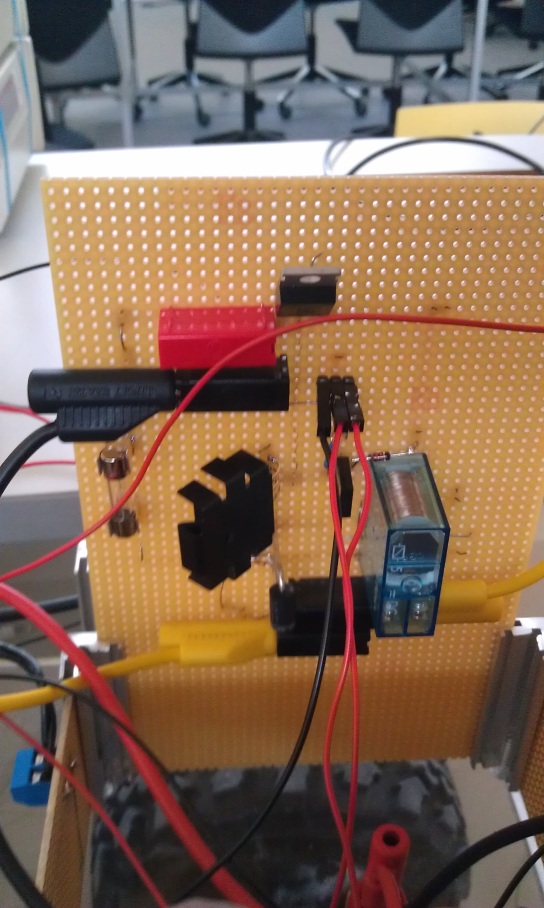
Da kredsløbet ville være svært at vise med alle forbindelser og relæspoledelen er det blevet forkortet op i to dele. Det skal forestille at + er forbundet til de to kontakter på relæet hvor der står +, og ÷ er forbundet til de to kontakter hvor der står ÷:



Figur 12: Relationen mellem relæets kontakter og motorreguleringskredsløbet

MOSFETen behøver ikke en ”gate modstand” da gaten på MOSFETs har højere input impedans

##### Implementering og test af H-bro og motorregulering

H-bro kredsløbet blev først testet på et fumlebræt sammen med motorreguleringen. En funktionsgenerator lavede et firkants-signal som blev brugt som PWM og en tryknap blev brugt til at sende 5V for at vende motorens køreretning. Alt fungerede som forventet.

H-broen blev så implementeret på samme veroboard som motorreguleringskredsløbet (MOSFET, sikring osv.) fordi begge delkredsløb hænger sammen og gør brug af samme batteri som hinanden og motoren opførsel kontrolleres af begge kredsløb.

Under lodningen af veroboardet blev der sørget for at baner hvor der ville løbe store strømme (5-8A) var fysisk mere adskilt end andre ting dvs. nok til at småting som bøjede komponentben, løst tin osv. ikke ville føre til alvorlige kortslutninger.

Der er fire banan-hunstik på kredsløbet: To til kabler fra 7,8V batteriet og to til kabler til motoren. Der er yderligere tre input pins på veroboardet: GND, pin til PWM og pin til at skifte H-broens retning.

##### Test af H-bro og motorregulering

Første gang der blev testet med microcontroller blev der brugt en frekvens på 7 kHz og duty cycle mellem 0-80%. Dette resulterede i at MOSFETen blev meget varm meget hurtigt. Frekvensen blev efterfølgende sat ned til 112 Hz og der blev sat en køleplade på MOSFETen. Dioden (1N4007) blev også varm nogle gange og der blev sat en større diode (1N5821) i stedet.

Efterfølgende lykkedes det at få bilen til at køre både med PWM-signal og en permanent 5V signal. H-broen skiftede også retning når den fik signal.

##### Mulige svagheder og fejlkilder ved H-bro og motorregulerings kredsløbet

Som man allerede kunne se på diagrammerne er det muligt at vende retningen uafhængigt af motorreguleringen fx kan man ved en softwarefejl komme til at bakke når bilen kører med fuld fart frem. Dette vil resultere i en meget kraftig opbremsning som noget af elektronikken næppe har godt af. Tælleren vil også tælle mange minuspoints i tilfælde af en kraftig opbremsning.

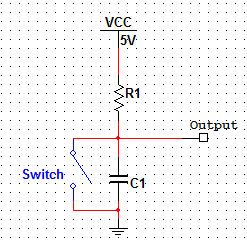
### Tæller

##### Fejltæller kredsløb

*Skrevet af: Kim Lykke Johansen*

For at skabe et kredsløb, i hvilket der indgår en kontakt(i vores til fælde et pendul), som kan registrerer et bestemt antal skift i et givet tidsrum, er det nødvendigt at sørge for at eliminerer den kontaktprel som der efter al sandsynlighed vil forekomme i systemet.

Elimination af kontaktprel kan opnås med følgende kredsløbsopbygning:



Figur 13 Kontaktprel elimination. Switchen erstattes af pendul i det egentlige kredsløb.

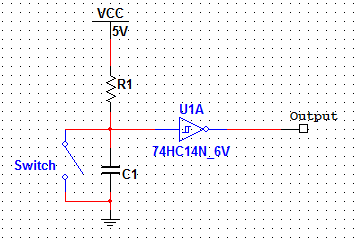
Resistorens placering sikre at kredsløbet ikke kortsluttes når switchen lukkes.

Når switchen er åben oplades kondensatoren og når switchen er lukket aflades kondensatoren. Dette giver med en afladt kondensator et lavt udgangssignal. Og når kondensatoren oplades svarer udgangssignalet til den opladte spænding over kondensatoren.

Kondensatorens opladningstid styres af størrelsen på kondensatoren og resistoren. Jo større kondensator værdi, des længere opladningstid. Jo større resistor, des længere opladningstid. Afladningstiden sker derimod næsten øjeblikkeligt.

Ovenstående kredsløbsopbygning skaber dog visse problemer, da opladningen af kondensatoren vil ske i en logaritmisk kurve, hvilket den counter som skal modtage signalet ikke er i stand til at behandle ordenligt. Dette kan resultere i usammenhængende registrerede impulser og counteren kan derfor hoppe rundt mellem høje og lave signaler, hvilket vil forsage at der tælles forkert i forhold til den ønskede præcision.

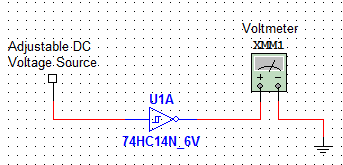
For at ændre på dette, tilføjes der en Schmitt trigger til kredsløbet, som vil konverterer det logaritmiske signal til et digitalt signal.



Figur 14 Kredsløb til kontaktprel elimination og konvertering til digital signal

En schmitt trigger konverterer den analoge indgangssignal til et digitalt udgangssignal, ved at afventer at den analoge spænding når hhv. over eller under et grænseniveau, hvorefter Schmitt triggeren vil ændre dens signal(bemærk dog den inverterende udgang). Disse grænseniveauer kaldes for positiv/negativ threshold spænding eller øvre/nedre hysterese(noteres ).

kan varierer meget fra Schmitt trigger til schmitt trigger og bør måles. Dette gøres ved at slutte en variabel spændingskilde til indgangsbenet på Schmitt triggeren og et voltmeter på udgangsbenet. I det udgangsspændingen skifter fra høj til lav aflæses på den variable spændingskilde(grundet den inverterende udgang).



Figur 15 Måleopstilling af threshold spænding

For dette kredsløb er det udelukkende der er interessant da det er denne der hænger sammen med opladningstiden for kondensatoren. Når er nået, vil udgangssignalet fra Schmitt triggeren være lavt, grundet den inverterende udgang. Hvilket betyder at når switchen/pendulet trækker Schmitt triggeren lav, vil dennes udgangssignal næsten øjeblikkeligt skifte til højt. Og det er denne rising edge som trigger counteren.

Når vi kender kan vi, ved at vælge en værdi til kondensatoren udregne størrelse af kredsløbets modstand med følgende formel:

Hvor *t* er den ønskede tidskonstant, *V0* er forsyningsspændingen, *R* er modstandsstørrelsen og *C* er kondensatorstørrelsen.

Vi målte til at være 3,7 V, valgte en kondensatoren på 10 µF og ønsker at kunne registrerer 5 udslag pr. sekund, hvilket giver en tidskonstant på .

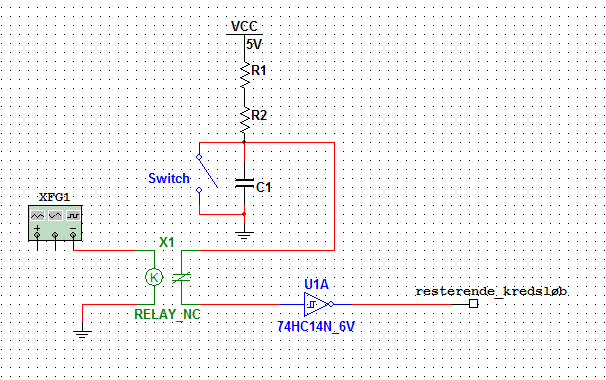
Vælger vi en modstand større end denne, vil vi få en opladningstid som overskrider vores grænse på 0,2 s. Derfor må vi vælge en mindre modstand. Den nærmeste er 13 kΩ, hvilket er en stor afvigelse og derfor har vi valgt at placere to modstande i serie for at opnå et bedre resultat. Ved at gøre dette er det dog vigtigt at tage højde for modstandenes tolerance.

Vi vælger 13 kΩ 1 % og 1,6 kΩ 1 %:

Vi tester om dette giver en opladningstid på mindre end 0,2 s og dermed en signalregistrering på minimum 5 udslag pr. sekund.

Vi har nu fundet de nødvendige komponent størrelser, og kan afslutte kredsløbet med to countere, to 7-segment display drivere og dermed to 7-segment displays med passende for modstande.[[1]](#footnote-1)

Afsluttende test af kredsløbet, foretaget med funktionsgenerator og relæ, viser dog at den teoretiske udregning ikke stemmer overens med praksis. Kredsløbet kan i stedet registrere op til 6,82 . Dette blev korrigeret med 13 kΩ og 5,6 kΩ modstande i serie hvilket registrerer op til 5,13 .

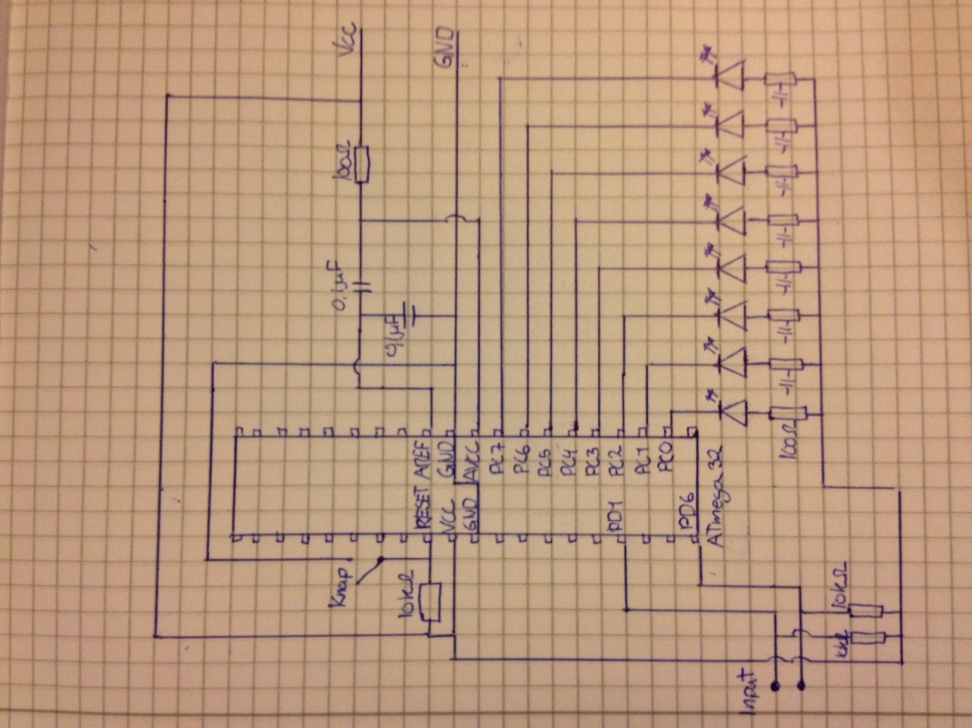


Figur 16 Test opstilling af tæller kredsløb

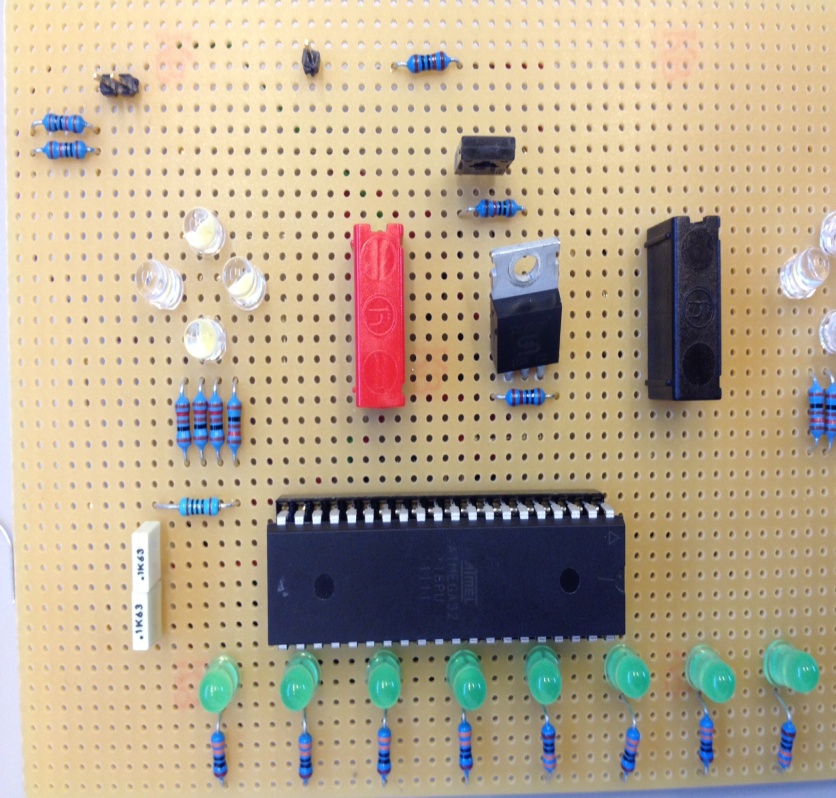
### Knight rider funktionen:

*Skrevet af Mark W. Maansson*

Knightrider funktionen bliver igangsat, når bilen starter. Funktion består af 8 grønne lys LEDer der sidder på linje. LEDerne tænder og slukker hver 200ms skiftevis, frem og tilbage. Når bilen så slukker stopper funktionen.

Er bygget op omkring en ATmega32. PC 0-7 er hver især forbundet til en grøn lys LED i serie med en 100ohm modstand, som er sat til VCC. Det gør, at når LEDs pin bliver sat lav vil LEDen lyse og slukke igen, når pinen bliver sat høj. PD 3 og PD 6 er forbundet til hver deres eksterne pin, PD3 og PD6 bliver holdet høje med hver deres 10Kohm modstand, der virker som pull-up modstande. PD6 og PD3 bliver sat til STK500 og får et lavt input derfra. RESET, bliver forbundet til VCC en via en pull-up modstand på 10Kohm, pin er også forbundet til GND via en knap, så når knappen trykkes resettes systemet. VCC benet sættes til VCC. Begge GND ben sættes til GND og forbindes til AREF benet via en kondensator på 0,1uF, AREF bliver også forbundet med ACVV benet via endnu en kondensator på 0,1uF, og videre til VCC gennem en 100ohm modstand. VCC og GND før strøm og spænding fra forlygternes spænding regulator.

Figur 1 Viser hele vores kredsløbs diagram for Knightrider funktion.

Figur 2 viser den samlede Knightrider funktion. De to pine oppe i venstre hjørne er der hvor funktion for sit input, med tilhørerne pull-up modstande. Nederst på figur 2 ser man ATmega32 sat i en passende sokkel, derunder sidder 8 grønne lys LEDer, som visuelt vil lyse, med tilhørerne modstande.

Figur 2 Knightrider på veroboard sammen med forlygterne

##### Delkonklusion:

Knightrider funktionen skulle have været sat sådan op, at man bare skulle kunne have sat ATmega32 fra STK500 over i systemet. Men system kom aldrig til at virker, en ting som vi måske kunne have ændret var 150ohm modstanden til en lidt mindre da signalet fra ATmega32 ikke sender et så høj signal ud. Så man kunne skifte dem ud med nogle lidt mindre, 51ohm eller måske 47ohm. En anden ting som kunne være et problem er måden AREF og AVCC blev sat op.

### Refleksions detektor

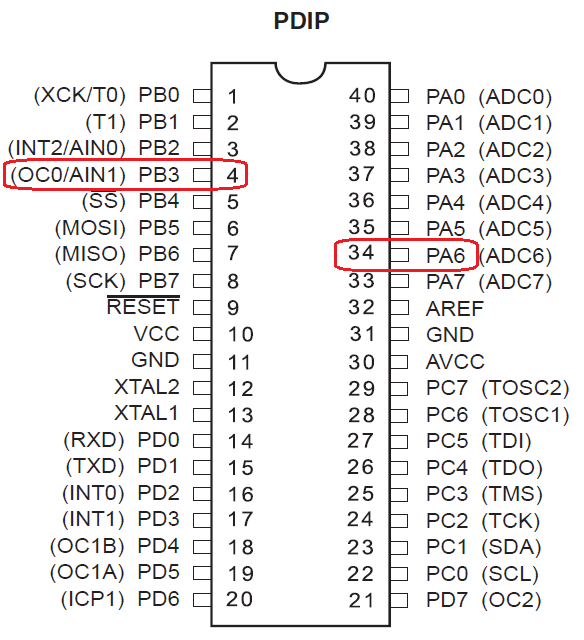
Se bilag refleksions detektor

## 

## Software

### Lys

*Skrevet af Kristoffer Sebastian Breuer*

Bilen er udstyret med et sæt forlys og baglys, som hver især kan styres individuelt via STK-500 via PORTB PIN6 hhv. PORTD PIN3, desuden kan lysintensiteten af baglyset også styres via STK-500.

Figur 17 - atmega32-microcontroller

##### Forlys

Til styring af forlyset benyttes PIN 6 på PORT A, hvor der programmeres et højt signal på udgangsbenet. Dette opnås ved at sende ”1” ud ad PORT A ben 6, dog uden at ændre andre bits på porten. Forneden ses hvordan forlyset bliver initieret.

void initFrontLEDPort()

{

DDRA |= 0b01000000; //PINA, 6 sat til udgang

unsigned char patternA = 0b10111111;

PORTA = PINA & patternA; //Det sikres at PINA 6 er sat til 0

}

For at sikre at PIN 6 er sat til 0, dvs. lyset skal være slukket, AND´er vi PIN 6 med 0, herved ændres den pågældende bit til 0 uden at ændre andre bits.

For at tænde forlyset bruges følgende kode i funktionen allLightsOn():

unsigned char patternA = 0b01000000; //BIT 6, active high

PORTA = PINA ^ patternA;

Herved ændres bit 6 til 1 og der sendes et højt signal ud som tænder for lyset vha. en transistor. Lyset er tændt fra det øjeblik motoren får tilført en spænding større end 0 V og slukkes når bilen har passeret sidste refleksbrik og stopper. For at slukke lyset skriver vi igen 0 til bit 6, koden hertil står i funktionen allLightsOff():

void allLightsOn()

{

unsigned char patternA = 0b01000000; //BIT 6, active high

PORTA = PINA ^ patternA;

OCR0 = 26; //10% duty cycle

}

##### Bremselys

Styring af baglyset/bremselyset sker vha. et PWM signal som sendes ud af PORT D ben 3. Selve PWM signalet programmeres med Timer0, da denne timer er enkel og de tilhørende registre er hurtigt initialiseret med få linjer kode. For at kunne burge Timer0 til PWM bliver vi nød til at benytte PORT D bit 3, da dette er OCR0 udgangen.

void initBackLEDPort()

{

DDRB |= 0b00001000; //PINB, 3 sat til udgang

unsigned char patternB = 0b11110111;

PORTB = PINB & patternB; //Det sikres at PINA 3 er sat til 0

TCCR0 = 0b01100010; //Timer0 ikke-fast PWM mode,

//compare match set when up counting

}

Foroven ses koden til Timer0 initialiseringen. Øverst sættes PORTD bit 3 til udgang, dernæst sikrer vi os igen at denne bit er sat til 0 vha. samme metode som hos forlyset. Til sidst initialiseres Timer0 registeres TCCR0 til at køre i fase korrekt(Phase Correct) PWM mode med en prescaler på 1. Desuden er timeren sat til at sætte OCR0 udgangen ved optælling.

Når vi tænder alt lys på bilen så snart motoren starter skal vores bremse lys ikke lyse med fuld kraft, da dette først skal ske når bilen bremser op. Derfor sætter vi OCR0 værdien således, at LED´en kun lyser med ca. 10% intensitet. Til beregning af denne værdi bruger vi fælgende formel:

Værdien af TOP afhænger af hvilken timer man bruger og hvad dens maximum værdi er. I vores tilfælde er denne værdi 255 eller 0xFF, da Timer0 er en 8-bit timer.

Hvis vi fx vil opnå en Duty Cycle på 10% ser beregningen således ud:

Den generelle formel kan omstilles så vi kan finde OCR0:

Indsætter vi vores tal i formlen ser den således ud:

Når vi indsætter 26 i OCR0 registeret får vi et PWM signal på ca. 10%. Koden hertil ser således ud:

OCR0 = 26; //10% duty cycle

Så snart bilen skal bremse forøger vi intensiteten af vores LED til 100% for at tydeliggøre selve opbremsning af motoren, den tilhørende OCR0 værdi bliver:

Funktionen for bremselyset sætter dermed OCR0 værdien til 255, som det kan ses forneden.

void brakeLightOn()

{

OCR0 = 255; //100% dutycycle

}

Efter opbremsning sættes OCR0 værdien igen til 26, som svarer til ca. 10% Duty Cycle, vha. en lignende funktion.[[2]](#footnote-2)

Så snart bilen har passeret den sidste refleksbrik og har stoppet slukkes bremselyset ved at skrive værdien 0 til OCR0 registeret, dette sker i funktionen allLightsOff():

void allLightsOn()

{

unsigned char patternA = 0b01000000; //BIT 6, active high

PORTA = PINA ^ patternA;

OCR0 = 26; //10% duty cycle

}

Lyset bliver ultimativt styret af Main programmet som kalder de enkelte funktioner alt efter hvilken refleksbrik passerer. Bremsefunktionen brakeLightOn() vil blive kaldt så snart motorens middelspænding mindskes, fx lige før bakken, således at lysets intensitet forøges og tydeligør opbremsningen. Efterfølgende sættes intensiteten tilbage til startværdien vha. brakeLightOff() funktionen. Samme procedure udføres i slutningen af banen når bilen skal til at stoppe.

I forlysets tilfælde gælder det, at lyset skal være tændt så snart middelspændingen for bilens motor er større end 0V, dvs. lyset er tændt under hele køre-sekvensen.

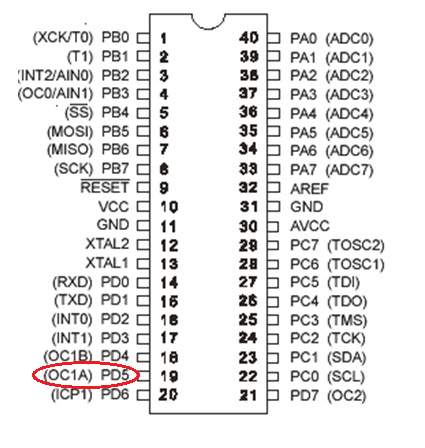
##### Delkonklusion lys

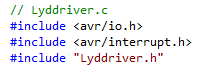
*Skrevet af Kristoffer Sebastian Breuer*

Da forlyset ikke var kompliceret at programmere, da dette kun kører over et højt dugangssignal på en af portene, er der ikke meget at sige omkring det. Derimod var beslutningen at lave bremselysets PWM programmering via Timer0 en god beslutning, da dette som nøvnt før er hurtigt, enkelt og ikke fylder meget kode. Selfølgelig kunne PWM signalet også være programmeret via Timer1 eller Timer2 uden stor forandring. Selvom der var problemer i starten med at lyd programmeringen ligeledes brugte Timer0, bare i en anden mode, kom lyset til at virke fint, især efter at problemerne med, at PWM signalet var for svagt for LED´erne at lyse, var blevet løst på veroboardene.

### Lyd

*Skrevet af David Buhauer*

Source-filen ”lyddriver.c” indeholder implementeringen. Her inkluderes header-filen ”lyddriver.h” samt de nødvendige AVR biblioteker.



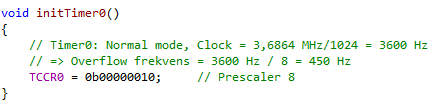
Globale variabler initialiseres. Heraf anvendes globale pointers og arrays som initialiseres i starten af implementeringen. Hver melodi er programmeret på baggrund af frekvenserne og tidsforsinkelserne[[3]](#footnote-3), som er lagt i hvert sit array, der følges ad i én funktion.

Interrupt biblioteket <avr/interrupt.h> inkluderes også, da implementeringen indeholder interrupts, så mikrokontrolleren kan lave noget andet imens, at den spiller en melodi, f.eks. PWM styring af lys på bilen.

Figur 19 - PORTD ben 5 anvendes som lydudgang.

Der skrives til PORTD ben 5, som er markeret på figur 19. Dette er lydudgangen, som tilsluttes en lydbuzzer og OCR1A benet tildeles værdier der svarer til forskellige lydfrekvenser.

Der anvendes timer0 og timer1, som er implementeret ved:

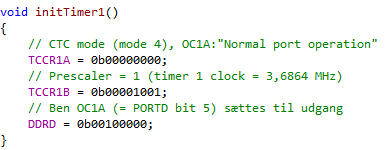


**Funktionsnavn:** initTimer0()

**Parameter:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Prescaler er sat til 8 i kontrolregistret TCCR0. Da der anvendes en clock frekvens på 3,6864 Mhz, vil overflow frekvensen være 450 Hz, dvs. periodetiden vil være . Det vil sige, at der vil være overflow hvert 2,222 ms. Dette bruges til at beregne antal overflows, der skal være for en bestemt tidsforsinkelse for hver tone i en melodi.



**Funktionsnavn:** initTimer1()

**Parameter:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Funktionen sætter kontrolregistrene TCCR1A og TCCR1B til hhv. CTC mode og en prescaler på 1. PD5 initialiseres til udgang og hermed er OC1A benet sat.

*I det følgende tages der udgangspunkt i Super-Mario melodien, men de andre melodier følger samme princip.*

**Eksempel på to globale variabler:**



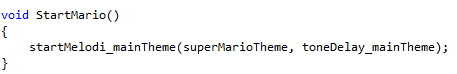
Dette array af typen char betegner en melodi, hvor hvert bogstav i arrayet er en tone og hvert ’-’ tegn er en tidsforsinkelse.



Dette array af typen int betegner tidsforsinkelsen, hvor hvert bogstav ( i dette tilfælde ’L’) er længden af hver tone defineret i starten af programmet, og tallene er antal overflows for hver tone.

##### void StartMario()

Denne funktion er implementeret ved:



**Funktionsnavn:** StartMario()

**Parameter:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Denne funktion kalder en anden funktion kaldt startMelodi\_mainTheme, som tager arrayet med mario tonerne som argument og arrayet med de forskellige delays for hver tone som argument.



Figur 20 – Aktivitets diagram for StartMario()

StartMario() kalder altså startMelodi\_mainTheme, som ud fra aktivitetsdiagrammet er implementeret her



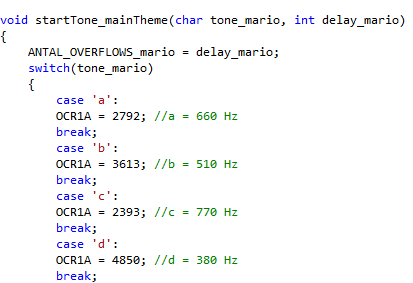
**Funktionsnavn:** startMelodi\_mainTheme( char \*marioPtr, int \*mDelayPtr)

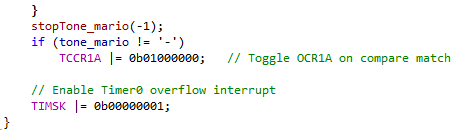
**Parameter:** char \*marioPtr – pointer til char[] indeholdende melodi og int \*mDelayPtr – pointer til int[] indeholdende delay/overflow.

**Returværdi:** Ingen

**Beskrivelse:** char \*marioPtr peger på første plads i melodi arrayet og int \*mDelayPtr peger på første plads i overflow arrayet. Hvis den modtagne pointer \*marioPtr peger på første plads i melodi arrayet, skal den globale melodi pointer mario\_melodiPtr samt delay pointer marioDelayPtr sættes lig med det modtaget. Herefter kaldes startTone\_mainTheme som tager de modtaget parameter som argument. Herefter tælles den globale tone pointer mario\_melodiPtr og den globale delay pointer marioDelayPtr én frem, så de peger på næste plads i arrayet, dvs. næste tone, som skal spilles. På denne måde vil tone arrayet og delay arrayet følges ad, og derved spille hver tone med den korrekte frekvens og tidsforsinkelse, samt længden af hver tone.

En lille del af implementeringen af startTone\_mainTheme funktionen er beskrevet her:





**Funktionsnavn:** startTone\_mainTheme( char tone, int delay\_mario )

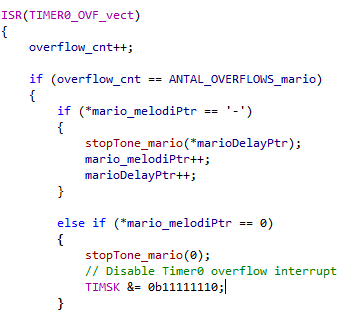
**Parameter:** char tone\_mario – arrayet til char[] indeholdende melodi og int mDelayPtr – arrayet til int[] indeholdende delay/overflow.

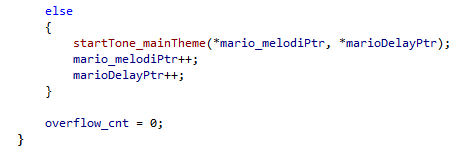
**Returværdi:** Ingen

**Beskrivelse:**

Funktion startTone\_mainTheme( char tone\_mario, int delay\_mario) sætter de rigtige frekvenser, som begynder at komme ud af lydbuzzeren der er forbundet til OC1A benet. I funktionen anvendes en switch struktur. Herved er der defineret, hvad der skal ske, når der står ’a’ et eller andet sted i melodi arrayet. F.eks. ved den, at ’a’ betyder tonen med 660 Hz, som bliver sat, når arrayet støder på ’a’. Den enabler herefter et interrupt, således at en interrupt subrutine starter.

En del af denne kode til subrutinen er implementeret her:





**Beskrivelse:** Dette interrupt bliver kørt for hver tone og sørger for at stoppe melodien og starte melodien alt afhængig af, hvad pointerne peger på.

En anden vigtig global variabel kaldt ”overflow\_cnt” inkrementeres, dvs. tælles én frem for hvert overflow, der kommer. Hvis denne globale variabel er præcis lig med det som delay pointeren peger på i et melodi array, så vil melodien stoppes i et kort øjeblik, hvis tone pointeren peger på tegnet ’-’. Ellers hvis tone pointeren peger på sidste plads i melodi arrayet, så skal melodien stoppes helt, og herefter vil timer0 overflow interruptet blive disabled. Hvis ingen af de betingelser er opfyldt, spilles næste tone. Så længe der bliver spillet en tone, vil overflow\_cnt variablen tælles én frem. Når melodien er helt slut, vil variablen sættes til 0, så næste melodi er klar til at blive spillet.

##### Delkonklusion for lyd

Lyden fungerer og er blevet enhedstestet på et STK-500. Men når vi integrerer hardware og software i main programmet, så udviser lyden konflikter. Lyden bruger en timer0 i normal mode, men lyset skulle også køres i timer0 men i PWM mode, hvilket gav konflikter med at spille startlyden, dvs. Super-Mario, og slutlyden, dvs. LevelComplete, med de rigtige tidsforsinkelser. Derfor var vores løsning at sætte Super-Mario lyden til at spille for sig selv på et andet STK-500. Dette lykkedes dog at få Coin lyden, som skulle spilles under detektion af en refleksbrik og bakkelyden som skulle spilles, når bilen bakkede, til at virke i PWM mode sammen med lyset i main programmet. Herefter når bilen var nået igennem banen, så blev kontrolregisret i timer0 ændret til normal mode igen, så levelComplete også kunne spilles korrekt i main programmet.

### M**o**tor

*Skrevet af Martin Østergaard*

Motoren bliver styret af et PWM signal fra STK-500 kittet efter anvisning fra kravsspecifikationen. Derudover bliver retningen af DC-motoren også styret vha. et signal fra STK-500 kittet, som bliver sendt ud til vores H-bro som så derefter kan trække motoren i den ønskede retning.

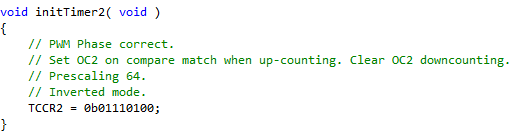
Som sagt skal motoren styres af et PWM signal, der kan gå fra 0 til maksimal batterispænding i 256 trin. Alle Mega32’s timere indeholder faktisk et PWM mode, men i forhold til at styre motoren er det Timer 2 der er blevet tildelt denne opgave. Timer 2 er en 8 bit Timer og virker derfor på samme måde som Timer 1, dog med nogle ændringer i forhold til registre, output porte og prescalere. Vi har valgt at bruge Timer 2 i dens PWM, Phase Correct mode da denne egner sig godt til styringer af motorer. Forskellen mellem Fast PWM og PWM Phase Correct mode er F.eks. at fasen forbliver den samme i Phase Correct mode uanset duty cyclen og at i non-inverted Fast PWM kan duty cyclen ikke blive 0%.

I opsætningen af Timer 2 skal der vælges en prescaler for at bestemme frekvensen. Frekvensen kan ikke vælges til hvad som helst da top værdien for timeren i mode 1 er fastsat til en konstant. Frekvensen kan udregnes vha. følgende formel:

Da der er et givet antal prescalere kan vi fremstille en tabel med de mulige frekvenser i forhold til Mega32’s clock frekvens som ligger på 3,6864MHz.

Tabel 1 Timer2 mode1 frekvenser

|  |  |
| --- | --- |
|  | Prescaler |
| 7728,24 | 1 |
| 903,529 | 8 |
| 225,882 | 32 |
| 112,941 | 64 |
| 56,471 | 128 |
| 28,235 | 256 |
| 7,059 | 1024 |

Vi har sat Timeren op således at vi får en prescaler på 64 og dermed en frekvens på omkring 112Hz. Timeren bliver initialiseret vha. initTimer2 funktionen.

Duty cyclen kan også regnes ud, men denne formel afhænger af om man har valgt inverted mode eller non-inverted mode.

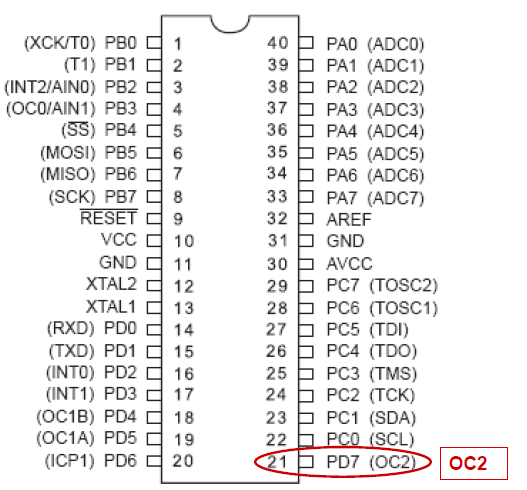
Selve styringen af motoren foregår vha. tre forskellige funktioner. Den første funktion DriveForward1() får bilen til at køre fremad i en vis hastighed i overensstemmelse med dutycyclen på PWM signalet. Vi har ikke fået bilen til at accelere konstant op til en vis hastighed. Når vi giver motoren signalet til at sætte i gang giver vi den et signal med en konstant duty cycle fra starten.

Når bilen så skal bremse bliver det gjort vha. funktionen Brake(). Denne funktion giver et PWM signal med en duty cycle på 0% som så vil få bilen til at motorbremse.

Efter bilen har bremset skal den også kunne bakke. Dette gør vi vha. Reverse() funktionen. Denne funktion er også ret simpel da den blot sender et skifte signal ud til H-broen som så vender polariteten på motoren, hvorefter et nyt PWM signal vil få bilen til at bakke.

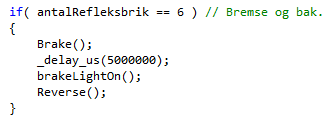
|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Parameter | Returværdi | Funktionalitet |
| DriveForward1() | - | - | Sætter til output:  PD6(motorretning)  PD7(PWM)  OCR2 sender PWM signal |
| Brake() | - | - | Duty cycle 0 % |
| Reverse() | - | - | Bakke signal til H-bro  OCR2 sender PWM signal |

Figur 21 - Motor funktioner

PWM signalet bliver sendt fra PORTD ben 7 og bakke signalet bliver sendt fra PORTD ben 6.

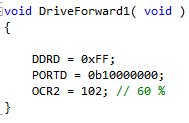
Figur 22 Styring af motor porte

Selve styringen af hvornår der bremses og bakkes og hvordan bilen orienterer sig foregår i vores main program. Kort sagt tæller vores reflektionsdetektorer detektioner af refleksbrikker op i en variabel. Denne variabel bliver så brugt i vores main program i en do-while lykke med forskellige if sætninger. Disse if sætninger styrer hvor bilen er henne på banen og hvad den skal gøre. F.eks. ved refleksbrik 6 skal bilen bremse og bakke, så hvis reflektionstælleren bliver lig med 6 kalder den Brake() funktionen og Reverse() funktionen.



Figur 23 - Brug af funktioner

Når motoren skal køre fremad bliver det gjort vha. DriveForward() funktionen:



Figur 24 - DriveForward()

Funktionen starter med at sætte PORTD op til udgange. Derefter skriver vi til PORTD at PD7 skal være høj da det er PWM signalet og PD6 er lav da motoren skal køre fremad. Efter at have testet bilen på banen fandt vi ud af bilen skulle starte med at køre med en duty cycle på 60 %.

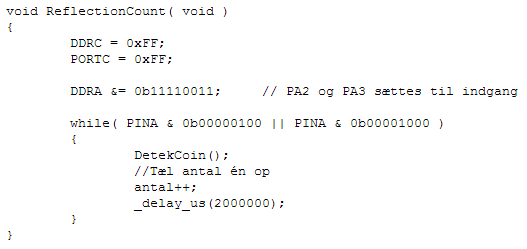


Et forbedringsforslag til styringen af motoren kunne have været, at vi i stedet for bare at sende et PWM signal ud med en bestemt duty cycle, kunne vi have lavet en konstant acceleration op til vores ønskede duty cycle så bilen ville køre mere balanceret og dermed formegentlig ville få mindre fejl detektioner under kørslen.

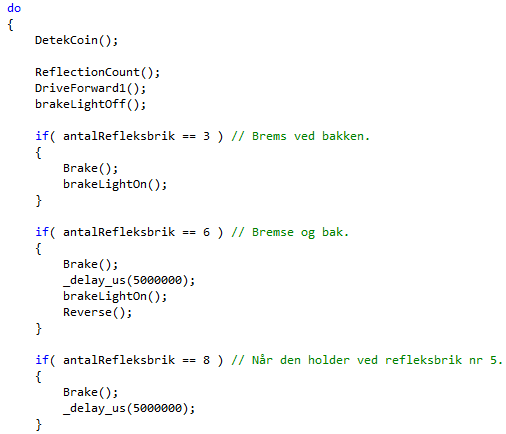
### Refleksions detektor

*Skrevet af Mathias Schlütter*

Refleksions programmering er styringen af de producerede refleksionsbrikker. Refleksionsbrikkerne & software virker som "hoved" styringen af bilen, de fungerer som bilens navigations system ved hjælp af de refleksbrikker som befinder sig på banen.



Den overstående kode er hoved koden for Refleksdetektoren, den aktiveres når PINA 2 på STK500-kittet bliver høj. Når dette sker, bliver DetekCoin(); initieret, hvilket er en kode som spiller en "coin" lyd. Ved "High" bliver der også lagt 1 til char'en "antal". Denne bliver brugt navigation af bilen, dette giver også adgang til "bonus" kørslen (bilen bakker fra brik 6 til brik 5 og derefter fortsætter fremad). \_delay\_us koden giver os en delay på refleksbrikkerne så der ikke kan bliver lavet fejl i opfangning af brikker. Dette er nødvendigt da man ellers risikerer, da der er en refleksbrik på begge sider, at man får 2 HIGH signaler og derved ville bilen kun komme halvvejs igennem banen før den ville afslutte programmet.



Selve styringen af bilen er som sagt udført med refleksdetektorene.

I overstående kode vises der hvordan bilen reagerer på de forskellige brikker og hvilke kommandoer der bliver givet ud.

Alt styringen er lavet i if-løkker da det er en nem styring, da bilen kun skal reagere ”hvis” antal brikker er blevet opfanget og derved kan bilen selv vide sin position på banen.

Coin-lyden er kodet som følgende

void DetekCoin( void )

{

startMelodi\_coin(Coin, toneDelay\_coin);

}

**Funktionsnavn:** DetekCoin()

**Parameter:** Ingen

**Returværdi:** Ingen

**Beskrivelse:** Denne funktion kalder en anden funktion kaldt startMelodi\_coin, som tager arrayet med mario tonerne som argument og arrayet med de forskellige delays for hver tone som argument.

void startMelodi\_coin(char \*coinPtr, int \*cDelayPtr)

{

if ( \*coinPtr )

{

coin\_melodiPtr = coinPtr;

coinDelayPtr = cDelayPtr;

startTone\_coin(\*coinPtr, \*cDelayPtr);

coin\_melodiPtr++;

coinDelayPtr++;

}

}

**Funktionsnavn:** startMelodi\_coin(char \*coinPtr, int \*cDelayPtr)

**Parameter:** char \*coinPtr – pointer til char[] indeholdende melodi og int \*cDelayPtr – pointer til int[] indeholdende delay/overflow.

**Returværdi:** Ingen

**Beskrivelse:** char \*coinPtr peger på første plads i melodi arrayet og int \*cDelayPtr peger på første plads i overflow arrayet. Hvis den modtagne pointer \*coinPtr peger på første plads i melodi arrayet, skal den globale melodi pointer coin\_melodiPtr samt delay pointer coinDelayPtr sættes lige med det modtaget. Herefter kaldes startTone\_coin som tager de modtaget parameter som argument. Herefter tælles den globale tone pointer coin\_melodiPtr og den globale delay pointer coinDelayPtr én frem, så de peger på næste plads i arrayet, dvs. næste tone, som skal spilles. På denne måde vil tone arrayet og delay arrayet følges ad, og derved spille hver tone med den korrekte frekvens og tidsforsinkelse, samt længden af hver tone.

### Bluetooth

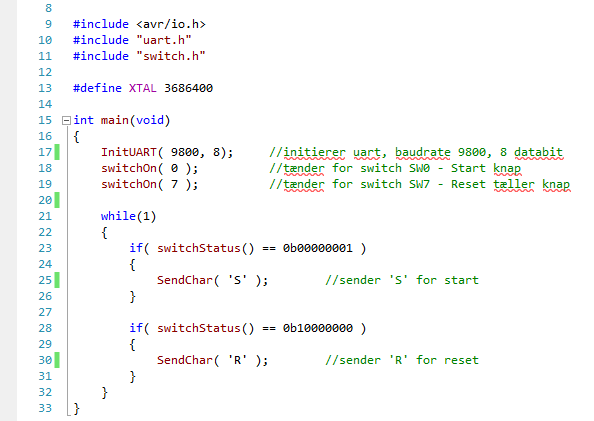
*Skrevet af: Kim Lykke Johansen*

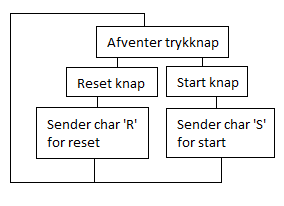
Som fjernbetjening anvendes der et STK 500 kit med en Atmel AVR ATmega32 tilsluttet. Denne skal kommunikerer med den Atmel AVR ATmega32 der styrer bilens program.

Til at skabe en kommunikation mellem disse to mega32 processorer, anvendes STK 500 kitsenes RS232 spare porte til seriel kommunikation via UART funktionen på mega32.

På fjernbetjenings STK 500 kittet, bruges to switches til hhv. start og reset funktionen.

Tidligere kreerede moduler bruges til at initierer UART og switches[[4]](#footnote-4).



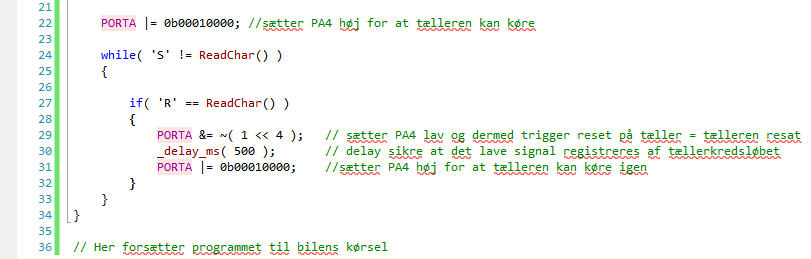
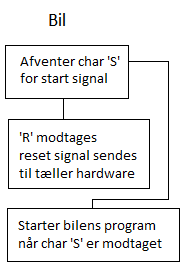
Ovenstående programkode følger aktivitetsdiagrammet fra figur 25. While(1) lykke sørger for at fjernbetjeningen bliver ved med at kører sit program.

Figur 25 - Aktivitetsdiagram for fjernbetjening

If() lykkerne med funktionskaldet switchStatus(), tjekker om der er trykket på den tilegnede switch på STK 500 kittet.

Er dette tilfældet kaldes funktionen SendChar(), som via UART’en sender hhv. char ’S’ eller char ’R’ via RS232 spare porten.

Når modtageren på bilens mega32 er initiereret, ser koden der håndterer start af bilen og reset af tælleren ud som følger:



Figur 26 Aktivitetsdiagram for modtager(bilens mega32)

Figur 27 Modtager kode på bilens ATmega32

Her står while lykke og kører så længe der ikke er modtaget et ’S’ som start signal, dvs. så snart start signalet modtages, hoppen programmet ud af while lykken, hvorefter selve kørsels programmet starte.

Inde i while lykke, er if lykken med reset funktionen. Dette sikre at tælleren ikke kan blive resat under kørsels.

For at gøre fjernbetjeningen trådløs er der anvendt et RS232 prepaired bluetooth enhed(BL-875). Man skal dog være opmærksom på at et null modem er nødvendigt for at spejlvende signalet således at det passer til modsatte STK 500 kits RS232 spare port.

### 

### Main.C

*Skrevet af Kristoffer Sebastian Breuer*

Alle tidligere beskrevne funktioner er blevet implementeret i main programmet som til sidst styrer bilen gennem banen.

I starten af Main initialiseres alle timere og porte for lys, lyd, motor og refleksbrikker, desuden benyttes funktionen sei() til at enable globale interrupts som fx skal bruges til lydstyring. Da bilen først skal startes når man trykker på en knap på fjernbetjeningen, starter main programmet med en while() løkke som venter på, at det står det programmerede startsignal/tegn i UDR registeret.

while ( sålænge der ikke modtages et signal fra bluetooth );

Tænd alle lys

Start tæller

Start motoren

Så snart signalet fra bluetooth er registreret tændes lyset, reflekscounteren startes og motor funktionen udføres. På fjernbetjeningen startes desuden Super Mario themet så snart signalet er sendt.

Efter bilen kører kontrollerer den globale reflekscounter variabel hvor bilen er på banen og hvornår bilens bremsefunktion skal udføres:

if( refleksbrik X detekteres )

{

Kald bremsefunktion

Tænd bremselygte

}

”Coin” lyden ved refleksdetection bliver spillet i selve reflectionCount() funktionen som detekterer reflekserne ved at tælle den globale countervariabel op så snart der er et lavt signal på PORTA ben 2 og 3. Desuden er der et delay til sidst for at forhindre at der registreres flere signaler per refleksbrik.

void ReflectionCount( void )

{

DDRC = 0xFF; // Fri udgang

PORTC = 0xFF;

DDRA &= 0b11110011; // PA2 og PA3 sættes til indgang

while( PINA & 0b00000100 || PINA & 0b00001000 )

{

DetekCoin();

//Tæl antal én op

antalRefleksbrik++;

\_delay\_us(2000000);

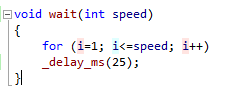
}

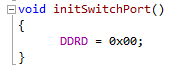
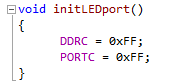
}

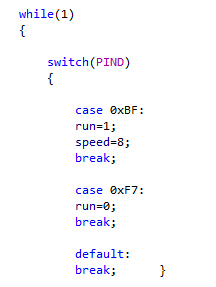
Bremsefunktionen kaldes første gang ved refleksbrik 3 og anden gang ved brik 6, hvor ligeledes bakke funktionen kaldes. Efter at den sidste refleksbrik bliver registreret stoppes motoren, alle lysene slukkes og slutlyden afspilles. Alternativt kan tælleren på bilen blive resettet via fjernbetjeningen efter at værdien er blevet aflæst.

### Knight rider funktionen:

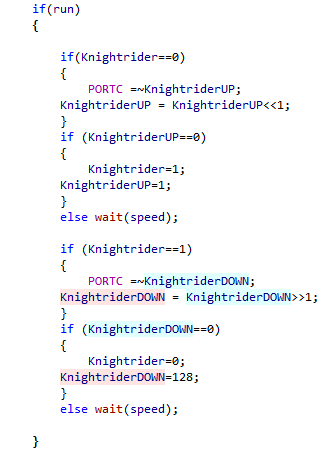
*Skrevet af Mark Maansson*

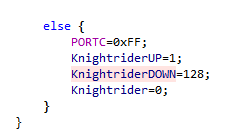
Funktionen wait, er lavet ved hjælp af en for-løkke og et delay som er for initieret ved #include <util/delay.h>. Funktionen funger sådan at wait venter på en int speed, som er et hel tal der bliver initieret i main, der bliver sat ind i for-løkke. Dette tal afgører hvor mange gange for-løkken bliver gentaget, og derved hvor mange gange delayet bliver kaldt.

initLEDport funktionen initiere Portene C til udgange og gør sådan at alle Port C udgangene vil være høje, og dermed vil alle lys LEDerne være slukket. Dette gøres fordi vi skal tænde lys LEDerne. initSwitch funktionen initiere portene D til indgange, som vi skal bruge til at få et lavt indgangs signal ind, når Knightrideren skal starte eller stoppe.



Når de forskellige porte er initieret går programmet ind en while-løkke, som vil køre programmet om og om igen ind til man slukker for Knightrider funktionen. Switch(PIND) holder øje med at Port D går lav. Men kun hvis PD6 (0xBF) blive lav vil run=1 og speed=8. Run er den bestemmende faktor for om lys LEDerne skal starte med at lyse. Speed er den bestemmende faktor, som forklaret længere op, hvor langt delayet skal være. Den eneste måde at stoppe lyset i LEDerne igen er ved at PD3 (0xF7) bliver lav, så run=0.

Programmet går kun ind i den if-sætning hvis PD6 (0xBF) bliver lav, da kun i denne case vil give run værdien 1. Og run=1 er den givende faktor for if-sætningen. Hvis Knightrider =0 bliver KnightriderUP output komplementær til PORTB så hvis der kommer et 1 ind på en af bittene vil LEDen lyse på bittens plads. Derefter bliver der flyttet et 1 tal en til venstre så lys LEDen tilhørerne bit0 tændes. Der efter vil programmet gå ind i wait funktion med den speed som var givet af casen speed = 8. Da speed=8 og \_dealy\_(25)ms vil delayet ca. vare 200ms. Her efter vil programmet gå ind i et loop som bliver ved indtil at 1 tallet er rykket nok gange til venstre til at 1 tallet er ude af de 8 bit igen og KnightriderUP igen bliver lig med 0, dermed kommer vi ind i den næste if-sætning (if (Knightrider==0)) Knightrider bliver sat til 1 og KnightriderUP bliver sat til sin oprindelige værdi. Det næste der vil ske er at KnightriderDOWN funktionen bliver sat i gang, den gør praktisk talt det samme som KnightriderUP bare omvendt, så 1 tallet sendes højre om. Knightrider bliver så lige med 0 og programmet for lader loopet. KnightriderDOWN bliver sat tilbage til dens oprindelige værdi lig med 128.

Else-sætningen køre hvis PD3 (0xF7) bliver lav, så run=0. Else-sætningen slukker alle lyse LEDerne ved at sætte alle portene høje. Også sætter alle de oprindelige værdier til det som de startede med at være. og dermed kører LEDerne ikke længere. Hele Knight rider funktion er testet på STK-500, og fungere som forventet.

##### Delkonklusion:

I forhold til softwaren kunne man også lave switchen om til "if" "if else" og lave et fast delay i stedet for det som vi har brugt som kan varieres. Da disse funktioner egentlig bare skulle være det samme hele vejen igen. Men vi har valgt at beholde lidt mere fleksible funktioner da det eneste denne ATmega32 skulle styre var lys og fleksible funktioner, giver det også os eller en kunde muligheden for at ændre fx. hvor hurtig lysende skal køre frem og tilbage, uden at skulle lave så meget om. Det er også nemt at tilføje flere Knightrider striber på bilen, hvis dette skulle blive aktuelt.

# Accepttest

## Accepttest specifikationer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Krav nr. | Krav | Test | Forventet resultat | Resultat | Godkendt/ kommentar |
| 1.1 | Middelværdien af forsyningsspændingen til bilens motor skal styres via pulsbreddemodulation (PWM) fra 0 til maksimal batterispænding i 256 trin. | Der laves et testprogram til microcontrolleren, der cyklisk genererer et PWM-signal med dutycycle fra 0 – 100% i 256 trin med en periodetid på 10 sek. Signalet aflæses med et oscilloscop. | Ved observation af oscilloscopet konstateres det at PWM-signalet ændrer sig monotont fra 0 – 100% dutycycle på 10 sek. | Resultat er som forventet | Godkendt |
| 1.2 | På bilens højre og venstre side skal placeres detektorer, der kan registrere en R80 refleksbrik i en afstand (2 cm til 25 cm) +/- 1 cm. Se Figur 3 og Figur 4. | Refleksbrikken fjernes. | Detektorerne registrerer ingen refleksbrik. | Resultatet er som forventet, dog kan eksterne lyskilder interfererer med resultatet(fejlkilde) | Godkendt  Fejlkilde forsøgt rettet ved at afskærme detektionsdioden |
| Højre/venstre Refleksbrik flyttes i detektorens detektionszone i afstanden 2 – 25 cm fra sensoren. | Højre/venstre detektor skal under hele testen registrere refleksbrikken. |
| 1.3 | Print til detektorer skal have en bredde på 80 mm +/- 1 mm. Se Figur 3 og Figur 4. | Printes bredde måles med skydelære. | Printets bredde er 80 mm +/- 1 mm. | 80 mm | Godkendt |
| 1.4 | Bilen skal kunne afgive forskellige lydsignaler ved start, stop og detektion af refleksbrik – samt afgive lydsignal, når bilen bakker. | Bilen startes. | I de 4 situationer, observeres tydeligt forskellige lydsignaler, der sammenholdes med det genkendelige lydsignal | Virker isoleret, men mangler tilpasning på slutprodukt | Delvist godkendt.  Kan fixes under videreudvikling |
| Bilen passerer en refleksbrik. | Virker efter hensigten |
| Bilen bakker. | Virker efter hensigten |
| Bilen stopper. | Virker efter hensigten |
| 1.5 | Bilen skal kunne udføre bak sekvensen for bonus point. | Ved detektion af refleksbrik nr. 6, testes om strømmens retning vendes.  Ved detektion af refleksbrik nr. 5, testes om strømmens retning vender tilbage til udgangspunktet. | Ved detektion af refleksbrik nr. 6, skal bilen begynde at bremse og bakke – indtil efterfølgende detektion af refleksbrik nr. 5., hvorefter bilen skal kørefremad igen. | Bilen bakker som den skal. | Godkendt men kan optimeres under videreudvikling |
| 1.6 | Bilen skal kunne betjenes via radio fjernbetjening, rækkevidde minimum 25 m i frit rum(på nye batterier(hvis batteridrevet model)). | Fjernbetjeningen testes fra den givne minimums afstand. Der observeres, at bilen modtager signal fra fjernbetjeningen. | Bilen kan modtage signal fra 25 m eller mere. | Bluetooth enheden kan sende og modtage over ønsket afstand.  Ikke monteret på bil pga. manglende batteri til Bluetooth enhed | Delvist godkendt.  Batteri kan under videreudvikling laves således at bilen kan forbindes til fjernbetjeningen. |
| 2.1.1 | Implementeres med 8 hvide LED, der monteres 4 i højre og 4 i venstre side på bilens front. Se Figur 3 – LTW-2S3D7 eller kompatible. | Testes visuelt. | Der er monteret 4 hvidlysende LED’er i højre og venstres på bilens front | Virker optimalt | Godkendt |
| 2.1.2 | Kørelyset må kun være tændt, når bilens motor påtrykkes en middelspænding > 0 Volt. | PWM signalet til motorens forsyningsspænding sættes til 0 % dutycycle. | Kørelys er slukket. | Kørelyset lyser kun når der kommer et PWM signal ind i systemet | Godkendt |
| PWM signalet til motorens forsyningsspænding sættes til 10 % dutycycle. | Kørelys er tændt. |
| 2.1.3 | Når det hvide lys er tændt skal middelstrømmen gennem hver LED være 50 mA +/- 1 mA. | Middelstrømmen ved tændt lys måles med et amperemeter. | Middelstrømmen er 50 mA +/- 1 mA. | 49,57 mA målt | Godkendt |
| 2.2.1 | Implementeres med 8 røde LED, der monteres 4 i højre og 4 i venstre side på bilens bagende. Se Figur 4 – LH3300 eller kompatible. | Testes visuelt. | Der er monteret 4 rødlysende LED’er i højre og venstres på bilens bagende | Virker optimalt | Godkendt |
| 2.2.2 | Når middelspændingen til bilens motor mindskes, skal middelstrømmen gennem hver bagudrettet LED være 50 mA +/- 1 mA i en periode på 2,5 s +/- 0,5 s. | Der laves et testprogram til microcontrolleren, der cyclisk skifter dutycycle til motorens forsyningsspænding mellem 40% og 80% hvert 10. sek.  Programmet afvikles på bilen uden at de trækkende hjul rører underlaget.  Strømmen gennem LED måles med DC-amperemeter. | I det øjeblik bilens hjul mindsker hastighed skal den målte strøm være 50 mA +/- 1 mA i en periode på 2,5 s +/- 0,5 s. | 50,43 mA målt | Godkendt |
| 2.2.3 | Når middelspændingen til biles motor øges eller er konstant, skal middelstrømmen gennem hver bagudrettet LED være 10 mA +/- 1 mA. Krav 2.2.2 har højere prioritet end krav 2.2.3 | Der laves et testprogram til microcontrolleren, der cyclisk skifter dutycycle til motorens forsyningsspænding mellem 40% og 80% hver 10. sek.  Programmet afvikles på bilen uden at de trækkende hjul rører underlaget.  Strømmen gennem LED måles med DC-amperemeter. | I de tilfælde hvor amperemeteret ikke viser 50 mA i henhold til test af krav 2.2.2, skal amperemeteret vise 10 mA +/- 1 mA. | 9,81 mA målt | Godkendt |
| 2.3 | Print til fremadrettet og bagudrettet kørelys skal have en bredde på 160 mm +/- 1 mm. | Printes bredde måles med skydelære. | Printets bredde er 160 mm +/- 1 mm. | 160 mm målt | Godkendt |
| 2.4 | Farvet lys til at sidde under bilen: På undervognen af bilen skal der sidde belysning, som er nedadrettet. | Testes visuelt | Grøne lyskilder er monteret på undersiden af bilen. | Ikke implementeret | Ikke godkendt.  Fremskaffelse af ønskede komponenter skabte problemer. |
| 2.5 | Knight rider belysning skal monteres lige under forlygterne, den skal bestå af 8 grønne LED på række. | Testes visuelt | Grønne LED monteres på rækkeunder forlygterne. | non funktionalitet (Hardware) | Ikke godkendt, fejl kan oprettes ved mere tid |
| 2.6 | Når Knight rider er tænkt skal middelstrømmen gennem hver LED være 50mA +/- 1 mA. | Testes visuelt | Middelstrømmen er 50 mA +/- 1 mA. | non funktionalitet  (Hardware) | Ikke godkendt, fejl kan oprettes ved mere tid |
| 2.6 | Knight rider må først starte når bilen kør og slukke når bilen slukker | Testes visuelt | Knight rider lys er tænkt når bilen køre og slukket når slukkes. | non funktionalitet  (Hardware) | Ikke godkendt, fejl kan oprettes ved mere tid |
| 3.1 | Lyden ved start skal være en genkendelig version af: Start lyden Super Mario. | Bilen startes og lydsignalet sammenholdes med ”Start” - lyden Super Mario. | Lydsignalet er en genkendelig version af ”Start” - lyden Super Mario | Selve lydsignalet virker efter hensigten, dog ikke implementeret på bilen | Delvist godkendt. Kan implementeres under videreudvikling |
| 3.2 | Lyden ved stop skal være en genkendelig version af: Level complete fra Super Mario. | Bilen stoppes og lydsignalet sammenholdes med ”Level complete” - lyden fra Super Mario. | Lydsignalet er en genkendelig version af ”Level complete” fra Super Mario | Resultat som forventet | Godkendt |
| 3.3 | Lyden ved dektektion skal være en genkendelig version af: Coin lyd fra Super Mario. | Refleksbrik detekteres og lydsignalet sammenholdes med ”Coin” - lyd fra Super Mario. | Lydsignalet er en genkendelig version af ”Coin” - lyd fra Super Mario | Resultat som forventet | Godkendt |
| 3.4 | Lyden når bilen bakker,skal være en genkendelig version af: en lastbil som bakker. | Bilen bakker og lydsignalet sammenholdes med en lastbil som bakker. | Lydsignalet er en genkendelig version af en lastbil som bakker. | Resultat som forventet | Godkendt |
| 4.1 | Bilen skal kunne accelerere hurtigst muligt, i en jævn acceleration, så der registreres så få fejl som muligt, på fejltælleren med pendulet på. | Vi lader bilen køre test på banen. | Der vurderes hvorvidt bilen accelererer jævnt, i passende hastighed, ved at se på, hvor mange fejl tælleren registrerer. | Kørsels viser nogenlunde jævn Acceleration | Delvist godkendt, kan Optimeres ved mere tid |
| 4.2 | Bilen skal være i stand til at stoppe, bakke og køre frem ad igen. | Vi lader bilen køre test på banen. | Vi observerer at bilen bakker ved refleksbrik nr. 6 og kører frem igen ved refleksbrik nr. 5 | Bilen bakker som den skal. | Godkendt men kan optimeres under videreudvikling |
| 4.3 | Bilen skal være i stand til at stoppe inden for 1,5 m målområde, i en jævn opbremsning der sikre et minimum af straf point. | Vi lader bilen køre test på banen. | Vi observerer at bilen stopper inden for det 1,5 m lange måleområde.  Og vurderer om der bremses jævnt ved at se på tælleren. | Bilen er lige på grænsen for hvor lang bremseafstanden må være | Delvist godkendt, kan optimeres ved mere tid |
| 5.1.1 | Med fjernbetjeningen skal bilen kunne startes, således at programmet, der driver bilen op startes. | Fjernbetjeningens startknap trykkes ned. | Bilen starter. | Fjernbetjening ikke forbundet til bil, men virker isoleret set | Ikke godkendt.  Kan udbedres ved videreudvikling |
| 5.1.2 | Med samme fjernbetjening skal brugeren være i stand til at nulstille tælleren, der tæller pendul fejl ved ujævn kørsel. | Fjernbetjeningens reset-knap trykkes ned. | Tælleren bliver nulstillet. | Virker efter hensigt | Godkendt |
| 5.2.1 | På bilen skal der monteres et pendul, som kan detektere ujævn kørsel. | Ved prøvekørsel testes det om pendulet sidder fast på bilen | Pendulet bliver siddende på bilen | Pendul bliver siddende | Godkendt |
| 5.2.2 | Pendulet skal være forbundet til et digitalt display, som tæller hvor mange udslag der kommer under kørsel, minimum 5 udslag/sekund + ½ udslag/sekund. | Tællerkredsløbet forbindes til en funktionsgenerator, gennem et relæ. Herfra kan grænsefrekvensen aflæses. | Tælleren registrerer fejlene på displayet.  Og er i stand til at tælle minimum 5 fejl pr. sekund + ½ fejl pr. sekund. | Tælleren kan registrere udslag med minimum 5,13 Hz.  Der forekommer dog tællefejl | Delvist godkendt, kan udbedres ved videreudvikling |
| 5.2.3 | Pendultælleren skal have en reset-knap, som nulstiller det digitale display.(Udføres evt. som element på fjernbetjeningen). | Der trykkes på reset-knappen(enten på bilen eller på fjernbetjeningen). | Tælleren bliver nulstillet. | Virker efter hensigt.  Fjernbetjening dog ikke forbundet til bil. | Delvist godkendt  Kan udbedres ved videreudvikling |
| 5.2.4 | Reset funktionen må ikke være tilgængelig under kørsel. | Når der er talt nogle fejl under kørsel, trykkes der på reset knappen. | Tæller bliver ikke nulstillet | Virker efter hensigt.  Fjernbetjening dog ikke forbundet til bil. | Delvist godkendt  Kan udbedres ved videreudvikling |

# Konklusion

## Fælles konklusion

Ser vi på vores kravspecifikation, som vi udarbejded i starten af projektet, kan vi se, at alle vores hovedkrav er blevet opfyldt. Indenfor både hardware og software virker komponenterne og programmeringen for motoren/styringen, forlys og bremselys og lyden. Alle problemer, som opstod undervejs er blevet elemineret, som fx opsætning af STK-500til at styre en anden mega32 og noget overskærende programmering. Vores fjernbetjening opfylder også dets krav, dog måtte vi konstatere, at begge Bluetooth moduler skal have tilført en ekstern spænding for at kunne operere korrekt.

Den største udfordring har dog været at sætte alle komponenterne sammen i bilen og uploade koden dertil på STK-500 og efterfølgende få det til at arbejde sammen. Selvom komponenterne var blevet testet enkeltvis sammen med den tilhørende kode tog det flere timer inden de monterede komponenterne reagerede og arbejdede sammen på selve bilen.

Især motorstyringen og H-broen gav problemer i starten og det var det der tog længst tid at løse. Ligeledes refleksdetektorerne skulle finjusteres og tjekkes igennem mange gange før de virkede, da detektoren var meget følsom overfor lys i rummet og andre lyskilder.

Da vi havde meget travlt til sidst i ugen med at få testet bilen og dets funktioner må vi konkludere, at vi til tider har været lidt for sløve med at komme igang og at det ikke havde været forkert at starte nogle dage før, dermed kunne vi have færdiggjort projektet med større succes. Med en gruppe på 8 medlemmer var det til tider heller ikke enkelt at mødes og få aftalt specifikt hvad der skulle være færdigt hvornår, det blev noget nemmere da vi delte os op i en hardware og software gruppe som under hinanden kunne afgøre hvem der skulle lave hvad og især hvordan det skulle struktureres.

Ud over at have forbedret vores færdigheder indenfor de områder vi arbejdede med i software hhv. hardware har vi fået nye erfaringer indenfor samarbejde og rapportskrivning samtidig med hvordan det føles at arbejde med et større projekt, sat sammen af mange små dele som hver for sig skulle designes, konstureres og programmeres.

## Delkonklusioner

*Skrevet af Mads Gjerrild Søgaard*

Jeg kan ud fra dette projekt konkludere, at koordination med gruppen er vigtig, da der fx bliver lavet double arbejde eller der er nogle der står og ikke har noget at lave, projektstyring og planlægning har betydet meget for mig, da jeg arbejder bedst når jeg har en klar plan over hvad jeg skal nå. Projektet har givet stor forståelse for om teori og praksis ikke altid er det samme. Opdeling og udelegering, ud fra kravspecifikationer og accepttest har været med til at give overblik til hele projektet. Den største opgave har været at fejlfinde hvor der var fejl når hardware og software ikke virkede sammen, når det virkede hver for sig. Vi har desværre været for sent ude for at prøve køre vores bil, så vi fik ikke testet det hele sammen ordentligt igennem, men det kan muligvis skyldes at der ikke har været nok biler til at alle grupper, har haft adgang til biler på samme tid. Til vores møder med vores vejleder føler jeg, vi har fået stor hjælp mht. hvis vi havde tvivler eller spørgsmål omkring hvordan det hele virkede, til møderne var der også en god mulighed for at dele ideer og få snakket sammen om hvor langt de forskellige i gruppen var kommet. Jeg har også fået viden omkring at arbejde med materiale fra de forskellige kurser. Alt i alt syntes jeg det har været sjovt at arbejde med sådan et projekt, fordi det har givet et indblik i hvordan man kan arbejde med et projekt ude i den virkelig verden.

*Skrevet af Kristoffer Sebastian Breuer*

Selvom bilen til sidst ikke var helt klar til at køre banen uden problemer mener jeg, at det har været en god og lærrig erfaring at lave projektet. Det blev isært meget tydeligt hvor vigtigt det er, at alle er klare over hvad de står for og hvad der skal laves. Desuden er det alpha og omega, at de enkelte enheder(Software hhv. Hardware) kommunikerer sammen og fastlægger præcist hvordan de enkelte dele af fx bilen hænger sammen, men at der også indenfor fx Software gruppen arbejdes sammen og sammenlignes, fx havde vi problemer fordi lydprogrammeringen kolliderede med lyskoden fordi vi havde brugt samme timer i to forskellige modes.

Vigtigheden af selve møderne før det egentlige arbejde er ligeledes blevet særdeles tydeligt, da det er smart at have en præcis arbejdsplan og forestilling om hvad projektet indebærer. Under selve arbejdet med komponenterne var det godt, at man som software gruppe alligevel havde forstand på opbygning og funktion af komponenterne, da det gjorde det nemmere at skrive koden dertil og fejlfinding ved afprøvning.

## 

*Skrevet af David Buhauer*

De forskellige faser i projektforløbet har været med til at danne overblik og strukturering, hvilket har været rigtigt fint, men jeg synes det har været svært at uddelegere specifikke ansvarsområder til sin gruppe, som er så stor. Dette synes jeg også gav lidt problemer med forskellige opfattelser af, hvordan de enkelte faser skulle gennemarbejdes. Den mest besværlige fase har været integrationstesten, hvor alt software og hardware skulle arbejde sammen. Her løb gruppen ind i problemer og mange ting skulle ændres, specielt softwaren. Men det har forholdsvis været nemt at teste systemets softwaredele samt hardwareblokke hvert for sit. I projektforløbet har jeg lært hvor vigtigt strukturering er. Dette har haft stor betydning på det endelig resultat. Min styrke i projektet har været programmeringen til bilen og hele tiden at holde overblikket ud fra kravspecifikationen samt grænsefladen og designet af softwaren. Jeg har også lært at anvende viden fra kurserne tværfagligt og at der er langt mellem teori og praksis. Det har også været vigtigt at være til stede til møderne med vejlederen, fordi man fik samlet op på de forskellige faser i projektet, og vejlederen fik skubbet mig videre i projektforløbet med nye tanker på banen. Konkurrencen har også haft betydning, fordi det har været med til at motivere mig. Dog synes jeg ikke vi har haft så meget tid, men alligevel har det været sjovt, og det har givet mig et indblik i ingeniørarbejdet.

*Skrevet af Apinayan Mohanathas*

Projektet har muliggjort at vi har brugt ting vi har lært fra forskellige fag på at løse en opgave og lave et slutprodukt. Dette er nok den vigtigste erfaring at tage fra projektet.

Vores projektstyring har blot bestået af at aftale møder og snakke sammen om hvad der skal blive lavet. Det kan måske være motiverende hvis der blev brugt en mere fast form for projektstyring med mødereferater ol.. Måske kan der bruges et eller andet webværktøj til at have en fælles tasklist over småting der skal laves ved lignende projekter.

Vi har haft travlt det sidste stykke tid hvor vi formentlig fik udarbejdet mere end 80 % af projektet de sidste 3 dage af projektforløbet.

Jeg har lavet H-bro og motorreguleringskredsløbet og jeg har lært at selvom noget ser ud til at virke ved første forsøg kan det stadigvæk have alvorlige problemer som stor varmeudvikling ol..

Det har været lang tid siden jeg har loddet så det var også praktisk læring i at lave kredsløbet.

*Skrevet af Mark W. Maansson*

Jeg synes at semesterprojekt har fungeret til tider. Hvor jeg personligt har selv følt at projekt gruppe og jeg have en sølv start, da det tog noget tid før vi rigtig fandt ud af hvad vi ville gøre med det. Det er har personligt bevirket mit gå på mod. Og det hele blev lidt presset til sidst. Samtidig har vi været en all for stor gruppe efter min smag, hvor jeg føler at det har været svært at for gruppen at få uddelt opgaverne. Gruppen har nok manglet en leder for gruppe, det har været meget været for sig (mangel på kommunikation ). Men ud over det har det virkeligt været fedt at kunne få lov til at bruge alt den tværfaglige vinden praksis og brugt muligheden for at lære noget nyt. Især rapport opsætning har givet mig en del som jeg vil kunne bruge frem over, så som at lave krav specifikationer og accepttest specifikationer så man allerede ind man gå i gang med selve projekt, ved hvad for nogle porte der skal gå til hvad. Jeg har igen har fået pointeret at der kan være stor forskel mellem tanke til teori over til praksis. Har selv haft en del problemer med hardware, hvor jeg så har fået nogle redskaber i forhold til problem løsning. Men ellers har det været et rigtig spændene forløb, som har været en super afslutning og har fået en god ide om hvordan det kunne være ude i den virkelig verden. Glæder mig allerede til næste semesterprojekt.

*Skrevet af Martin Østergaard*

Jeg har næsten udelukkende arbejdet med softwaredelen af projektet og mere specifikt motorstyringen af bilen. Jeg havde allerede en smule erfaring med denne form for projektarbejde fra en HTX uddannelse så jeg følte mig allerede fra starten af projektet forholdsvis sikker i projektarbejdsformen. Selve projektoplægget var spændende og jeg så personligt meget frem til at komme i gang med at lege med bilen. Jeg blev dog lidt overrasket da vi kom i grupper af 8 personer, da kommunikationen derfor ville blive endnu sværere end det normalt er. Det var derfor også rart at der var vejleder møder hver uge som ligesom blev ankeret i vores projektarbejde, hvor vi hver uge kunne samle op på hvad vi havde lavet og hvad vi skulle fremefter. I starten var vi lidt sløve synes jeg selvom vi dog fik stuktureret og uddelegeret opgaverne fint men der blev ikke lavet særlig meget håndfast arbejde, hvilket vi dog fik rettet op på mod enden.

Da vi skulle skrive softwaren lavede vi hver især de drivere vi havde fået til opgave at lave og fik dem lavet. Efter at have lavet driverne var stemningen, blandt os der nu havde lavet dem, at så snart vi nu satte driverne sammen ville det bare virke med det samme eller kort efter. Det blev en stor erfaring for mig i hvert fald at det ikke bare virkede og selve fejlfindingsprocessen blev utrolig irretabel, men da det så begyndte at virke blev forløsningen bare det større. Alt i alt synes jeg det har været et sjovt projekt selvom vi dog ikke fik bilen til at køre helt som vi ville og vi var lidt sløve til at starte med, men der har været en god dynamik i gruppen og vi har arbejdet godt sammen.

*Skrevet af Mathias Schlütter*

Jeg synes jeg har fået en stor del viden og oplevelser ud af projektet. Viden i form af nye måde at se programmering og opbygning af hardware på. Det har været en kæmpe oplevelse, da det er første gang jeg har arbejdet sådan med teknik og arbejde med selve kernen af komponenters styring. Det er også en stor erfaring man får ved selv at sidde med hænderne i ”gryden”, i stedet for ”bare” at skrive noget teoretisk kode og altså se en direkte effekt denne har på komponenterne.

Projektet har også givet mig rig mulighed for at lærer min gruppe bedre at kende, samt lære om ens egne og andres stærke / svage sider, mht. opgaveskrivning / kodning / hardware produktion. Jeg må dog også konkludere at jeg måske skulle have taget lidt mere arbejde til mig, da jeg har haft svært ved at skrive nok i min opgavedel.

*Skrevet af Kim Lykke Johansen*

Arbejdet med dette projekt har budt på en del udfordringer, både fra den åbenlyse faglige vinkel, men også i forbindelse med gruppearbejdet og måden hvorpå gruppearbejde kan foregå.

Set fra den faglige synsvinkel har dette projektarbejde skab en bro mellem undervisningsteorien gennem semesteret og teoriens praktiske anvendelse. Her tænkes specielt på programmeringen, hvor vi i undervisningen primært har fået nogle LEDs til at blinke eller en højtaler til at spille en melodi. Men her i projektet har vi selv skabt anvendelse af de funktioner vi har lært at kende.

Ser vi på hvordan gruppearbejdet er forløbet, virkede det allerede fra starten som om at ambitionsniveauet ikke var det samme hos alle. Dette kombineret med at gruppen har manglet en ’født’ ledertype har betydet at projektet ikke har haft en ordentlig tidsplan. Desuden har der været problemer med fremmødet til bl.a. vejleder møder, hvor fuldt fremmøde nærmere har været undtagelsen end reglen.

I den forbindelse har jeg forsøgt at tage styringen i gruppen, men har ikke været i stand til at udfylde rollen som leder til fulde. Formentlig grundet en manglende erfaring på området. Det betyder dog ikke at jeg ikke vil tage en sådan udfordring op på et senere tidspunkt, men det har derimod givet mig en indsigt i hvordan jeg i fremtiden kan sørge for en bedre struktur under projekter.

Kortfattet har jeg altså opnået tværfaglige kundskaber til at anvende pensum fra forskellige fag, samt fået en indsigt i ledelse af gruppearbejde, hvilket har givet mig erfaringer og inspiration til hvordan et gruppearbejde skal styrres af mere faste rammer.

# Litteraturliste

Kontaktprel elimination:

<http://www.all-electric.com/schematic/debounce.htm>

Super-Mario lydfrekvenser og delays:

<http://wiki.mikrotik.com/wiki/Super_Mario_Theme>

Coin-lyden lydfrekvenser og delays:

<http://www.electro-tech-online.com/microcontrollers/117231-8-bit-mario-coin-sound-effect-pic-2.html>

Digital Fundamentals, af Thomas L. Floyd 10th edition

the avr microcontroller and embedded system, af Muhammad Ali Mazidi, Sarmad Naimi og Sepehr Naimi, edition 2011

1. Se bilag: Tæller diagram for fuldt kredsløbsdiagram. [↑](#footnote-ref-1)
2. Se funktion brakeLightOff() i vedlagte C-filer for lys, Lys\_functions.c [↑](#footnote-ref-2)
3. http://wiki.mikrotik.com/wiki/Super\_Mario\_Theme [↑](#footnote-ref-3)
4. Se bilag for switch og UART modul [↑](#footnote-ref-4)