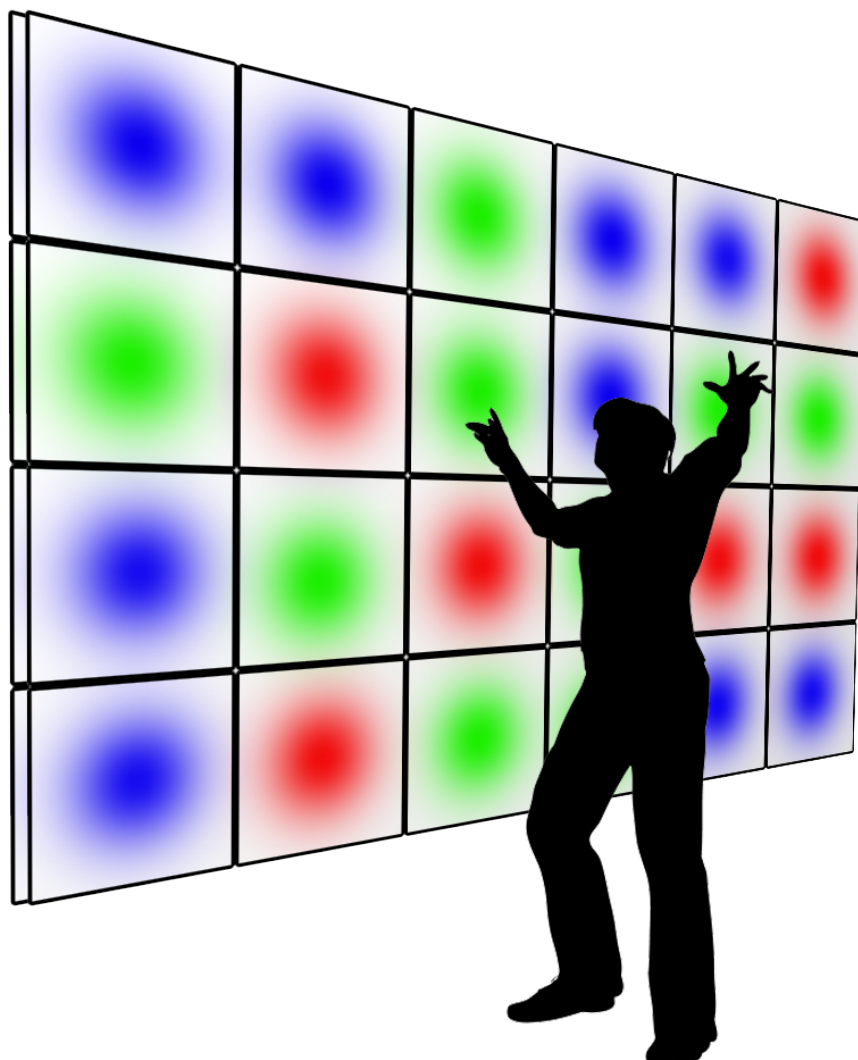




## Stimulering af syns- og hørehandicappede

Aalborg Universitet, Det Teknisk-Naturvidenskabelige Fakultet

P1, efterårssemesteret 2010

Projektgruppe B343



**Titel:**

Stimulering af syns- og  
hørehandicappede

**Tema:**

Virkelighed og modeller

**Projektperiode:**

P1, efterårssemesteret 2010

**Projektgruppe:**

B343

**Deltagere:**

Tomas Trillingsgaard  
Jacob Møller  
Johannes Hejselbæk  
Lasse Kristensen  
Mathias Tausen  
Jesper Wramberg Bundgaard

**Vejleder:**

Per Rubak

**Oplagstal: 9**

**Sidetal: 76**

**Bilagsantal og -art:**

6 stk. appendiks, 6 stk. bilag på CD

**Afsluttet den: 20. december 2010**

**Synopsis:**

Denne rapport omhandler konstruktionen af et spil til stimulering af syns- og hørehæmmede børn. Projektet tager udgangspunkt i en henvendelse fra Center for Døvblindhed og Høretab. Målet med projektet er at kunne aktivere og stimulere unge med omtalte handicap. Dette skal gøres ud fra en opstillet jernramme med 24 lysende, trykfølsomme felter. I rapporten behandles signalbehandling og dataopsamling med microprocessorer samt styring af RGB-dioder. I forbindelse med dette beskrives anvendelsen af teorien i sammenhæng med udviklingen af spil-løsningen og det elektroniske system som spillet er bygget op omkring. Løsningsforslaget giver mulighed for flere slags spil hvilket forbedrer mulighederne for at stimulere de handicappede børn. Til sidst giver systemet også mulighed for en dataopsamling således en eventuel fremgang hos eleverne kan spores og optimeres af deres lærere. Produktet opfylder kravene som blev stillet og igennem en accepttest bliver det i rapporten bevist at prototypen er brugervenlig således den opfylder kravspecifikationen.



# Forord

Denne rapport er udarbejdet af studerende på 1. semester ved Elektronik og IT på Aalborg universitet. Projektet er udarbejdet i perioden 11. oktober til 20. december 2010. Projekttemaet for dette P1-projekt er "Virkelighed og Modeller" med emnet Stimulering af Syns- og Hørehandicappede. Projektet fokuserer på konstruktionen af et elektronisk spil der kan anvendes til dette formål.

Tak til Per Rubak som har været vejleder i dette projekt og også til Ole Kiel Jensen som har bistået som kontaktperson til Center for Døvblindhed og Høretab (CDH). En speciel tak til Anders og Mads som er lærere ved CDH og har fremsat problemstillingen samt til Lærke Refstrup med datteren Johanne som har deltaget i accepttesten.

## Læsevejledning

Denne rapport henvender sig til vejledere, studerende og andre interesserede inden for området elektronik og IT, og forudsætter at læser har en grundlæggende viden indenfor dette område. Rapporten indeholder kildehenvisninger, nummereret efter Vancouvermodellen. Disse numre henviser til en litteraturliste i slutningen af rapporten. Listen indeholder relevant information om kilden, såsom forfatter, titel og udgivelsesår. Figurer og tabeller er nummereret efter deres placering i kapitel og sektion. Lister over disse findes i slutningen af rapporten kapitel 13 (s. 74). Der vil blive henvist til bilag og appendiks igennem rapporten. En fortegnelse over disse kan findes i kapitel 11 (s. 72) og kapitel 12 (s. 73).

Igennem rapporten dækker begrebet *elev* over spilleren af spillet - der er her tale om den handicappede som det ønskes at stimulere ved anvendelsen af produktet.

Begrebet *lærer* dækker over den person, der anvender produktet til at stimulere eleven. Læreren skifter mellem spil og kan se statistik, som beskriver udviklingen af elevens præstationer. Begrebet *applikationskontrol* dækker over et stykke software på en computer til valg og styring af spil. Igennem rapporten vil Center for Døvblindhed og Høretab blive henvist til som CDH.

**Alle bilag og appendiks kan findes på CD'en som ligger bagerst i rapporten**

# Indhold

|          |                                                |           |
|----------|------------------------------------------------|-----------|
| <b>1</b> | <b>Indledning</b>                              | <b>4</b>  |
| <b>2</b> | <b>Problemanalyse</b>                          | <b>5</b>  |
| 2.1      | Information om døvblindhed . . . . .           | 5         |
| 2.2      | Kortlægning af brugerønsker . . . . .          | 7         |
| 2.3      | Problemformulering . . . . .                   | 8         |
| 2.4      | Projektafgrænsning . . . . .                   | 8         |
| <b>3</b> | <b>Overordnede krav til produktet</b>          | <b>9</b>  |
| 3.1      | Overordnet systemdiagram . . . . .             | 9         |
| 3.2      | Applikationskontrol . . . . .                  | 9         |
| 3.3      | Eleinput . . . . .                             | 10        |
| 3.4      | Signalbehandling . . . . .                     | 10        |
| 3.5      | Dataopsamling- og behandling . . . . .         | 11        |
| 3.6      | Display-driver . . . . .                       | 11        |
| 3.7      | Display . . . . .                              | 12        |
| <b>4</b> | <b>Løsningsforslag</b>                         | <b>13</b> |
| 4.1      | Applikationskontrol . . . . .                  | 13        |
| 4.2      | Eleinput . . . . .                             | 14        |
| 4.3      | Signalbehandling . . . . .                     | 15        |
| 4.4      | Dataopsamling- og behandling . . . . .         | 15        |
| 4.5      | Display . . . . .                              | 17        |
| 4.6      | Display-driver . . . . .                       | 21        |
| <b>5</b> | <b>Kravspecifikation</b>                       | <b>23</b> |
| 5.1      | Accepttest specifikation . . . . .             | 23        |
| <b>6</b> | <b>Anvendt teori</b>                           | <b>24</b> |
| 6.1      | Microprocessorer og IC'er . . . . .            | 24        |
| 6.2      | PWM . . . . .                                  | 27        |
| 6.3      | RS232 . . . . .                                | 28        |
| 6.4      | ADC - Analogue to Digital Conversion . . . . . | 29        |
| 6.5      | Software . . . . .                             | 31        |
| 6.6      | Transistorer . . . . .                         | 37        |
| 6.7      | LED . . . . .                                  | 39        |
| <b>7</b> | <b>Design af Hardware</b>                      | <b>41</b> |
| 7.1      | Systemopbygning . . . . .                      | 41        |
| 7.2      | Felterne - in/out-put til eleverne . . . . .   | 42        |

|           |                                                 |           |
|-----------|-------------------------------------------------|-----------|
| 7.3       | RGB Controller . . . . .                        | 44        |
| 7.4       | Microprocessorer . . . . .                      | 46        |
| 7.5       | PC tilkobling . . . . .                         | 46        |
| <b>8</b>  | <b>Design af Software</b>                       | <b>47</b> |
| 8.1       | PC-software . . . . .                           | 47        |
| 8.2       | Arduino-software . . . . .                      | 56        |
| 8.3       | Displaydriver-software . . . . .                | 64        |
| 8.4       | Opsummering af software . . . . .               | 68        |
| <b>9</b>  | <b>Udførelse af accepttest</b>                  | <b>69</b> |
| <b>10</b> | <b>Konklusion</b>                               | <b>71</b> |
| <b>11</b> | <b>Appendiks oversigt</b>                       | <b>72</b> |
| 11.1      | Appendiks A: Overordnet systemdiagram . . . . . | 72        |
| 11.2      | Appendiks B: PC-software kildekode . . . . .    | 72        |
| 11.3      | Appendiks C: Arduino-kildekoden . . . . .       | 72        |
| 11.4      | Appendiks D: IC printudlæg . . . . .            | 72        |
| 11.5      | Appendiks E: Display-driver Software . . . . .  | 72        |
| 11.6      | Appendiks F: Manualer . . . . .                 | 72        |
| <b>12</b> | <b>Bilag oversigt</b>                           | <b>73</b> |
| 12.1      | Projektkatalog . . . . .                        | 73        |
| 12.2      | Produktforslag . . . . .                        | 73        |
| 12.3      | Interview med CDH . . . . .                     | 73        |
| 12.4      | Datablad til ATTiny261 . . . . .                | 73        |
| 12.5      | Datablad til anvendt MOSFET . . . . .           | 73        |
| 12.6      | Datablad til RGBB diode . . . . .               | 73        |
| <b>13</b> | <b>Referencelister</b>                          | <b>74</b> |

# Indledning

I Danmark findes der op mod 10.000 personer [1] som lider af nedsat syns- eller høreevne. De fleste af disse personer har i større eller mindre omfang problemer med at begå sig i dagligdagen, både pga. besvær med at befærde sig i samfundet, men også fordi folk med disse lidelser ofte har problemer med at blive aktiveret. Dette bevirker at det er meget sværere for mennesker med disse handicaps at lære igennem leg og spil fordi de ikke bliver stimuleret på samme måde som raske personer.

Der blev gjort opmærksom på dette problem, da der i projektkataloget blev efterspurgt en teknisk løsning til at afhjælpe denne problematik. Efterspørgslen kom fra CDH (Center for Døvblindhed og Høretab) i Aalborg. I denne sammenhæng forelå der et ønske om at få udviklet et elektronisk produkt som kan hjælpe med at stimulere de handicappede. Denne rapport beskriver udviklingen af dette produkt samt samarbejdet med CDH.

Projektkataloget opstillede de overordnede rammer for dette projekt og kan findes som bilag 12.1. Et uddrag gives her:

*Til at stimulere deres elever, efterspørger Center for Døvblindhed og Høretab, CDH, en aktivitet der kombinerer fysisk udfoldelse med sociale elementer og indlæring. CDH har allerede lavet den mekaniske ramme til "CDH-Væggen" vist på nedenstående figur, men de mangler noget elektronik til at fylde rammen ud.*



Figur 1.1: CDH-Væggen fra Projektkataloget

# Problemanalyse

## 2.1 Information om døvblindhed

*"Døvblindhed er en specifik funktionsnedsættelse. Døvblindhed er en kombineret syns- og hørenedsættelse. Den begrænser en persons aktiviteter og forhindrer fuld deltagelse i samfundet i et sådant omfang, at det kræver, at samfundet støtter med særligt tilrettelagte servicetilbud, tilpasninger af omgivelserne og/eller tekniske hjælpemidler." [9]*

**Hvad er et syns- eller hørehandicap?** Et handicap er en permanent fysisk eller psykisk sygdom og defineres som en funktionsnedsættelse, der har konsekvens for den handicappedes livsførelse [11]. Et syns- eller hørehandicap er derfor en psykisk eller fysisk sygdom, der har indflydelse på en persons evne til at se eller høre. Man kan imidlertid skelne mellem disse handicap - et synshandicap har typisk større indflydelse på personens øvrige sanser, da et sådant handicap ofte er fulgt af andre funktionsnedsættelser.

Dette projekt tager udgangspunkt i et specifikt ønske fra Center for Døvblindhed og Høretab (CDH), som ønsker at benytte sig af tekniske hjælpemidler, for at stimulere de syns- eller hørehæmmede børn på centeret.

**Hvordan opstår et syns- eller hørehandicap?** Døvblindhed kan være medfødt eller tilegnet. Døvblindhed defineres som medfødt, når funktionsnedsættelsen er til stede ved fødslen eller intræder før sproglig kommunikation er udviklet. En undersøgelse fra 2004 viste, at der i Danmark var ialt ca. 180 døvblindfødte. [7]

Døvblindhed kan også være tilegnet senere i livet - den hyppigste årsag er de forandringer i øjet og øret som kan opstå hos personer i høj alder. Den næsthypigste årsag er Usher syndrom. Der er her tale om en arvelig sygdom - høretabet optræder typisk i form af et sensorineuralt høretab, som betyder, at lydølgerne, der rammer øret, delvist eller slet ikke opfattes af hjernen. Hørenedsættelsen kan variere fra moderat hørenedsættelse til egentlig døvhed. Synsnedsættelsen optræder i form af øjensygdommen retinitis pigmentosa, som gradvist ødelægger øjets nethinde.

Usher-syndrom opdeles i tre kategorier afhængigt af høretabets karakter og personens balanceevne. En dansk undersøgelse fra 1997 viste, at der ud af 100.000 personer er 5 tilfælde af Usher syndrom[8].

**Hvor mange har et syns- eller hørehandicap?** I Danmark føres der ingen statistik over antallet af døvblinde - det er dog blevet anslået, at der er ca. 1250 personer med døvblindhed i Danmark. Undersøgelser fra Norge peger på, at der i Norge findes omkring 7500 "døvblind-blevne" blandt de ældre, og en Svensk undersøgelse peger på, at Sverige har mellem 8500 og 15.000 ældre med en kombineret høre- og synsnedsættelse. Der er dog i høj grad tale om en aldersbetinget funktionsnedsættelse.[16][10] CDH har ca. 30 døvblinde elever i børneskolen og over 60 døvblinde personer i alt. De har alle til fælles

at de har nedsat syn og hørelse. Derudover har de en række andre forskellige tillægshandicaps af både motorisk og kognitiv art [14].

**Hvad har syns- og hørehandicappede brug for?** For CDH's elever, hovedsageligt børn og unge, er syns- og hørenedsættelsen ofte kombineret med et motorisk handicap. Grundet disse personers nedsættelser er det vigtigt, at deres omgivelser indeholde kraftig respons, eksempelvis i form af kraftigt lys, lyd og/eller vibration. Hos CDH lægges der fokus på individet, og undervisning af eleverne foregår en til en. Børnene opdeles i grupper efter deres motoriske og kognitive formåen, og undervisningen tilpasses grupperne imellem. [19]

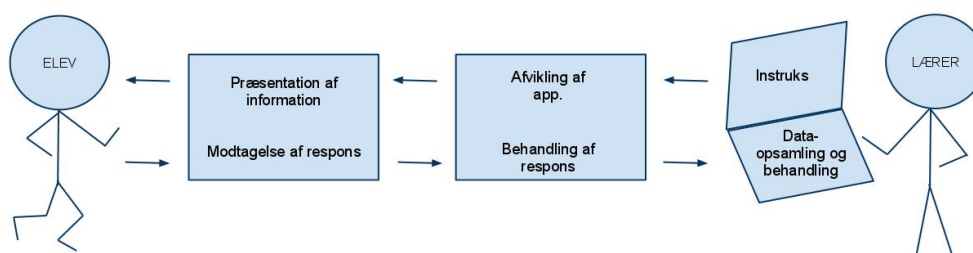
Til at stimulere eleverne efterspørger CDH en aktivitet, der kombinerer fysisk udfoldelse med sociale elementer og indlæring. Der kan læses om dette i projektkataloget som findes som bilag 12.1.

**Hvordan stimuleres syns- eller hørehandicappede?** De syns- og hørehandicappede stimuleres bedst, ved at få mulighed for at respondere på information. Denne information kan optræde i form af lys, lyd, vibration, dufte, smag, berøring eller lignende påvirkning af de handicappedes sanser. De syns- og hørehandicappedes sanser stimuleres ved at aktivere de hæmmede sanser i samspil med de velfungerende sanser - dette vil være meget individuelt, og den korrekte stimuli vil variere mellem børnene.

**Hvilke tekniske hjælpemidler kan benyttes?** De tekniske hjælpemidler skal påvirke sanserne og skal på den måde aktivere centre i hjernen som vil være mindre benyttet af den handicappede. Dette indebærer tre trin for den handicappede: Modtagelse af information, behandling af information og udførelse af handling på baggrund af informationen.

Det tekniske hjælpemiddel skal være i stand til at sende og modtage information, som er henvendt til den handicappede, så den handicappede er i stand til at modtage informationen. For at kunne analysere den handicappedes status og eventuelle fremskridt, er det desuden fordelagtigt, at det tekniske hjælpemiddel er i stand til at behandle og lagre information om det enkelte barns evner.

**Denne rapport tager udgangspunkt i hvordan et sådant teknisk hjælpemiddel kan designses.**



Figur 2.1: Blokdiagram med udgangspunkt i det tekniske hjælpemiddel

## 2.2 Kortlægning af brugerønsker

I startfasen af projektet blev der foretaget et interview med de to lærere og kontaktpersoner fra CDH, Anders Martin Rundh og Mads Kopperholdt. Ved interviewet forklarede de to lærere hvad de havde af forventninger til produktet. Forskellige forslag blev på mødet fremlagt for lærerne for at give en idé om hvad der kunne forventes af projektet. Disse forslag findes som bilag 12.2

Det følgende afsnit er en gennemgang af information fra lærerne på CDH.

Produktet skal bruges af elever fra CDH som er døvblinde. De døvblinde børn har både nedsat hørelse og syn. En del af eleverne har desuden nedsat reaktionsevne, og dette skal der tages hensyn til når spillene skal designes. Nogle af børnene er ude af stand til at bearbejde synsindtryk korrekt. Disse kan bedst se og forstå enkle og stærke farver. Det kan være svært for eleverne at tolke billeder med mange farver korrekt da de kan have ved at skelne farverne. Derfor skal spillene benytte sig af enkle og stærke farver og evt. simple figurer. De dygtigste elever ligger ca. på et 5. klasses niveau så spillene til produktet må ikke være for svære at gennemføre.

Lærernes forventninger til produktet er at det skal bruges til idræt og at det kan få eleverne til at bevæge sig. Produktet skal kunne bygges ind i en metal ramme som CDH har fået fremstillet. Rammen indeholder 24 felter i 4 rækker og 6 søjler. Rammens mål er 1,5 m. i bredden 2,5 m. i højden. Produktet skal være robust, da eleverne kan slå hårdt til knapperne. Produktet skal gerne kunne rengøres, da der er mange forskellige personer som kommer til at benytte sig af produktet stilles der derfor krav til hygiejne. Spillene må gerne benytte sig af et pointsystem, hvor man kan se hvor godt eleverne har klaret spillet. Disse point skal derfor kunne lagres så pointene kan trækkes frem igen senere, således at udviklingen hos den enkelte elev kan følges.

Lærerne fortalte at den mindste fremgang er en stor nyhed og det vil gøre både de ansatte og pårørende glade. Eleverne hepper på hinanden og der foregår ikke mobning blandt eleverne.

Der vil altid være en lærer til stede når produktet benyttes derfor behøves styring af produktet ikke at være autonomt eller være tilgængelig for eleverne. Spillene kan derfor designes så de enten afvikles automatisk eller styres af en lærer. Lærerne har ikke den store tekniske viden om computere, hvorfor styringen af produktet ikke skal være for avanceret. Hele interviewet findes som lydfil i bilag 12.3

### Ønsker til produktet fra CDH

- Det skal lyse med enkle og stærke farver
- Det skal lyse med forskellige farver
- Spillene må ikke være for avancerede
- Det skal kunne få eleverne til at bevæge sig
- Det skal være robust
- Det skal kunne rengøres
- Det skal være i stand til at lagre point som afspejler elevernes evner til at respondere på lysinformation
- Det skal være i stand at udskrive en rapport, der viser en elevs eventuelle fremgang over tid
- Det skal have et simpelt og letanvendeligt interface på pc

## 2.3 Problemformulering

Ud fra ønskerne fra CDH og den initierende problemstilling kan der opstilles denne problemformulering.

Hvordan konstrueres et elektronisk spil der kan stimulere  
syns- og hørehæmmede børn?

## 2.4 Projektafgrænsning

Der fokuseres på elektronikken og softwaren i dette projekt. Der arbejdes med ideen og løsningen og der åbnes op for videreudvikling af produktet. Målet er ikke at skabe et færdigt produkt i fuld størrelse ( $1.5 \times 2.5m$ ), men blot at konstruere en prototype, da det vurderes at tidshorisonten for et færdigt produkt med 24 felter i fuld størrelse er urealistisk. Et evt. lydindhold er ikke aktuelt i dette produkt, da der bliver derfor fokuseret på stimulering af de synsnedsatte børn på CDH. Målet er at færdiggøre prototypen med en styring af 24 felter, så software og hardware er kompatibelt med et færdigt system. Derfor vil display og trykknapper kun være en testopstilling.

**Følgende ønsker til produktet fra CDH har gruppen valgt at arbejde ud fra:**

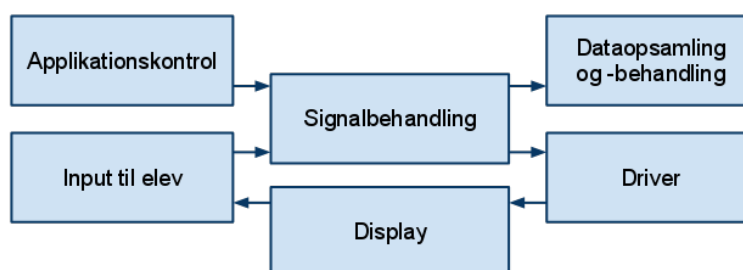
- Displayet skal lyse med forskellige farver
- Spillene må ikke være for avancerede
- Skal være i stand til at lagre point som afspejler elevernes evner til at respondere på lysinformation
- Skal være i stand til at vise en graf som viser elevernes fremgang over tid
- Skal have et anvendeligt interface på pc

## Overordnede krav til produktet

På baggrund af interview med lærerne på CDH og den afgrænsede problemstilling er der opstillet nogle ønsker til et system som kan løse problemstillingen. Disse ønsker opstiller en del tekniske krav til produktet. Set i forhold til den foreliggende studieordning vil disse ønsker og krav ikke alle kunne opfyldes. Derfor gennemgås i dette kapitel disse opstillede krav hvorefter der udvælges nogle overordnede tekniske krav at arbejde ud fra. For at løse problemstillingen og opfylde studieordningen kan man konstruere et system som er forklaret i 3.1. Dette system skal indeholde og afvikle et spil som viser farvet lys i forskellige displays som den synshandicappede har mulighed for at trykke på. Disse spil afvikles for eleven på en tavle med 24 displays. Dermed åbnes der en mulighed for at eleven bliver stimuleret gennem sine synssanser. Valg af løsningsmetode til de enkelte delelementer findes i kapitel 4 (s. 13) og gennemgangen af de endeligt opstillede krav findes i kapitel 5 (s. 23).

### 3.1 Overordnet systemdiagram

Overordnet set er der på inputsiden en applikationskontrol og input til eleven. Outputsiden består af dataopsamling og behandling af elevens respons og displays til eleven for at skabe interaktion. Alle blokkene i diagrammet vil blive uddybet i de følgende underafsnit.



Figur 3.1: Kommunikation mellem de enkelte enheder i systemet. Bemærk at eleven modtager et output fra display og kan trykke på det dertilhørende input og give respons.

### 3.2 Applikationskontrol

Elementet applikationskontrol står for styring og start af spil - den sender information til signalbehandling, som derefter eksekverer det valgte spil. Applikationskontrollen skal bestå af et interface, hvor en lærer kan vælge spil og starte spil. Jævnfør CDH's ønsker i sektion 2.2 (s. 7), er kravene til et færdigt produkt således:

Applikationskontrollen skal:

- Have et simpelt og letanvendeligt interface på PC.

- Være i stand til at starte og stoppe et spil.

Til et færdigt produkt skal applikationskontrollen naturligvis bestå af et meget simpelt og letanvendeligt interface - lærerne på CDH har ikke teknisk viden til at administrere et avanceret system. Det er ikke relevant at udføre et fuldstændigt brugervenligt og fejlfrit system på en PC, da fokus ligger på software til microcontrollere og elektronik. I forhold til projektafgrænsningen i sektion 2.4 (s. 8) laves der en prototype som opfylder disse krav:

Applikationskontrollen skal:

- Have et anvendeligt interface på PC.
- Være i stand til at starte og stoppe et spil.

### 3.3 Elevinput

Input til eleven er et væsentligt emne i produktet. Ved hjælp af dette input åbnes der for stimulering ved at give eleven mulighed for at respondere på lyset fra et display og trykke på det dertilhørende input. Eftersom CDH's elever kan have motoriske vanskeligheder og derfor sandsynligvis kan slå til spillet på en forholdsvist voldsom måde, kræver det, at inputtet på det færdige produkt er holdbart. Der er tale om en skole som naturligvis har veldefinerede krav til hygiejne er det derfor vigtigt, at produktet er let at rengøre - dette betyder naturligvis også at input til eleven skal være let at gøre rent.

Elevinputtet skal:

- Være i stand til at sende et signal til signalbehandlingen når inputtet bliver aktiveret.
- Være simpelt og let tilgængeligt.
- Være robust da det benyttes af børn op til 5 klasses niveau.
- Være let at rengøre.

Disse krav er krav til det færdige produkt - jævnfør projektafgrænsningen i sektion 2.4 (s. 8) bliver der designet og fremstillet en prototype. Derfor afgrænses elevinputtet til at

- Være i stand til at sende et signal til signalbehandlingen når inputtet bliver aktiveret.
- Kunne dimensioneres op, så videreudvikling til et færdigt produkt er mulig.

Kravet til prototypen er at den skal kunne dimensioneres op til et færdigt produkt. Med dette menes at teknikken skal kunne overføres direkte til det færdige produkt i fuld størrelse. Dette gøres ved at benytte samme type stik til tilslutning mellem input, display og signalbehandling på prototypen, som kan benyttes på et færdigt produkt.

### 3.4 Signalbehandling

Signalbehandlingens funktion i systemet er at håndtere kommunikationen mellem de centrale enheder. Den modtager således forskellige signaler, behandler disse og sender derefter signaler videre. På baggrund af dette opstilles krav til signalbehandlingen som skal være i stand til at:

- Modtage information fra applikationskontrollen ved hjælp af en USB forbindelse.
- Modtage signal fra elevinput.
- Indeholde styring og start af spil.
- Afvikle spil ved at styre displaydriveren vha. impulssignaler.
- Måle på elevinputs og sende denne information til dataopsamling.

### 3.5 Dataopsamling- og behandling

Dataopsamling og -behandlingen er den del, der står for at modtage en besked fra signalbehandlingen, og præsentere den på en overskuelig måde for en lærer. Jævnfør CDH's ønsker i sektion 2.2 (s. 7), er kravene til et færdigt produkt således:

Dataopsamling og -behandlingen skal:

- Være i stand til at modtage data fra signalbehandlingen.
- Være i stand til at lagre point som afspejler elevernes evner til at respondere på lysinformation.
- Være i stand at udskrive en rapport, der viser en elevs eventuelle fremgang over tid.

Til et færdigt produkt skal dataopsamling- og behandlingen naturligvis være i stand til, at udskrive en detaljeret, rapport, hvor lærerne kan følge elevernes fremgang på en meget overskuelig måde. Det er dog ikke relevant at udføre et fuldstændigt brugervenligt og fejlfrit system - fokus ligger på software til microcontrollere og elektronik. Jævnfør projektafgrænsningen i sektion 2.4 (s. 8) laves der en prototype som opfylder disse krav:

Dataopsamling og -behandlingen skal:

- Være i stand til at modtage data fra signalbehandlingen.
- Være i stand til at lagre point som afspejler elevernes evner til at respondere på lysinformation.
- Være i stand at vise en graf, der viser en elevs eventuelle fremgang over tid.

### 3.6 Display-driver

Kravene til displaydriveren tager udgangspunkt i dens funktion, da den skal kunne styre 24 display-felter. Dette skal den gøre på baggrund af et signal den modtager, igennem en enkelt forbindelse der indeholder et simpelt signal. Dette signal uddybes i sektion 7.3 (s. 44).

På figur 3.1 (s. 9) kan man se hvordan displaydriveren skal være placeret i forhold til resten af systemet. Denne driver skal forsyne displays som derfor kan lyse de enkelte felter kraftigt op. Samtidig skal displayfelterne kunne styres vha. et enkelt signal fra signalbehandlingen, hvilket er besværligt uden mellemlid. Når denne forbindelse er blevet behandlet af displaydriveren skal den efterfølgende kunne styre displayfelterne som kræver flere forbindelser for at kunne styres. For at lyse felterne op kan det blive nødvendigt at anvende mange dioder som derfor vil trække en vis mængde strøm. Derfor skal der vælges komponenter der kan holde til relativt store strømme op mod 1A for at have mulighed for at skabe en kraftig effekt.

Vi kan på baggrund af ovenstående opstille følgende krav til displaydriveren:

- Den skal kunne styre 24 displays.
- Den skal kunne styres vha. et signal.
- Den skal kunne styre store strømme op mod 1A.

### 3.7 Display

Displayet er den del af produktet der viser et billede, farver eller et andet output til eleven. Displayet sidder efter displaydriveren i systemet (se figur 3.1 (s. 9)). Displayet skal kunne styres af displaydriveren. Displayet skal være i stand til at vise forskellige farver for at give mulighed for flere forskellige spil.

Ønsket fra CDH er:

- Det skal lyse kraftigt.
- Det skal lyse med forskellige farver.

Da projektgruppen kun laver en prototype (se sektion 2.4 (s. 8)) er der ikke behov for at den skal lyse kraftigt. Derfor har gruppen valgt at køre videre med dette krav til displayet:

- Det skal kunne styres af displaydriveren.
- Det skal lyse med 7 forskellige farver.

## Løsningsforslag

På baggrund af de overordnede krav der er stillet til produktet, vil her komme nogle løsningsforslag til systemets delelementer. Hvert forslag vil indeholde en kort beskrivelse af det enkelte elements funktion i prototypen og et valg af løsningsmetode.

Overvejelsen i forbindelse med løsningen inddrager også løsninger til den færdige model ønsket fra CDH fordi der stilles krav om at systemet skal kunne opskaleres til de endelige displays med mange dioder. Valg af løsningsmetoder er dog taget i forhold til prototypen.

### 4.1 Applikationskontrol

Applikationskontrollen er den del, der står for at sende kommandoer til signalbehandlingen. Applikationskontrollen består hovedsageligt af et interface, hvor en lærer skal være i stand til at igangsætte forskellige spil og sende andre kommandoer til signalbehandlingen.

For at gøre processen nemmere anvendes en PC til applikationskontrollen - og i den sammenhæng anvendes også seriel kommunikation (RS232).

Applikationskontrollen kan naturligvis laves på forskellige måder. I dette afsnit gennemgås de overvejede løsningsforslag.

#### 4.1.1 Hjemmeside

En hjemmeside kan anvendes til styring af systemet. Hjemmesider bliver i stigende grad anvendt til denne type styring - en fordel er, at en hjemmeside i allerhøjeste grad er cross-platform, så den kan tilgås fra en hvilken som helst PC, smartphone el.lign.

Dog kan en hjemmeside ikke kommunikere direkte med systemet, og vil således kræve en form for program eller service, som er i stand til at oversætte kommandoer fra hjemmesiden til systemet, og afsende dem. Løsningen vil dog være i stand til at opfylde alle kravene beskrevet i sektion 3.2 (s. 9).

Fordele:

- Cross-platform - hjemmesiden kan tilgås fra en hvilken som helst enhed med internetforbindelse og en browser.

Ulemper:

- Vil kræve en form for service på en PC, som er i stand til at modtage kommandoer fra hjemmesiden.

#### 4.1.2 C#-applikation

C# er en videreudvikling af C og C++, og er et objektorienteret højniveau programmeringssprog.

En C#-applikation og Visual Studio som IDE (Integrated Development Environment) kan anvendes.

Dette gør udviklingen af software nemmere - der er nem adgang til en række libraries, så seriel kommunikation er simpel at sætte op. Desuden vil det gøre dataopsamling og -behandlingsdelen nemmere, hvis denne eventuelt kombineres i en C#-applikation, da præsentationen af data (såsom grafer) kan gøres meget nemt heri.

Det har dog den ulempe, at C# kræver adgang til en .net framework, og programmer bliver meget tunge - et stykke software skrevet i C# kræver en installeret .net framework. Et C#-program vil være i stand til opfylde alle kravene, som er beskrevet i sektion 3.2 (s. 9).

Fordele:

- Nemt og hurtigt at udvikle.

Ulemper:

- Kræver installation af .net framework.

### 4.1.3 Valg af løsning

Siden projektets hovedfokus ligger på elektronikken og udviklingen af software til microcontrolleren, anvendes PC'en hovedsageligt som et hjælpeværktøj, hvorfor C# anvendes. På denne måde kan det nødvendige software hurtigt udvikles, og der kan lægges mere fokus på elektronikken og microprocessor-softwaren.

Fremover i rapporten vil applikationskontrollen og dataopsamlingen referere til et stykke C#-software og kontrolelementer derpå.

## 4.2 Elevinput

Der var forskellige ideer til hvordan opbygningen af elevinputtet skulle udformes. Kravene fra CDH lød på, at den skulle være simpel, let tilgængelig, holdbar og rengøringsvenligt. Da CDH har stillet en ramme med 24 felter til rådighed. Ud fra denne er der overvejet flere metoder til løsningen af CDHs ønsker til det færdige produkt. Første løsningsforslag byggede på, at der skulle benyttes en fjeder til opbygningen af en trykknop. Denne trykknop skulle virke, så eleven trykkede på en mat akrylpladen, som ved hjælp af fjeder kan give sig. Idet akrylpladen giver sig ville den ramme en kontakt som sidder bag pladen. Denne kontakt sender så et signal videre til signalbehandlingen.

Andet løsningsforslag bygger på at der i stedet for en fjeder benyttes en gummiliste. Fordelen ved en gummiliste er at den gør monteringen lettere, samtidig med at den har samme funktion som fjederen. Der blev valgt løsningen med en gummiliste, det ville være den mest optimale løsning. Den giver mulighed for en bedre løsning omkring opbygning af en ramme til akrylpladen og trykknappen.

Løsningen til montering af akrylpladerne på rammen som CDH har stillet til rådighed bliver løst på følgende måde. Der laves en ramme som holder 6 akrylplader ved siden af hinanden, og som samtidig passer ind i det hulrum som er bag på rammen. På figur 4.1 (s. 15) ses rammen som CDH har stillet til rådighed bagfra og her kan det ses at der er et hulrum til 4x6 felter. Ifølge projektafgrænsning i sektion 2.4 (s. 8) fremstilles der blot en prototype, men med mulighed for at overføre teknikken til et færdigt produkt, ved hjælp af et standard stik.

Der er valgt, at erstatte løsningen beskrevet i denne sektion, med en trykknop til hver diode som er



Figur 4.1: Ramme fra CDH

monteret på prototypen. Når trykknappen aktiveres sendes et signal videre til signalbehandlingen. Dette uddybes nærmere i sektion 7.2 (s. 42).

### 4.3 Signalbehandling

Som udgangspunkt blev det overvejet at lave signalbehandlingsdelen med en microcontroller. Instituttet opfordrede de studerende til at bruge Arduino'er som er en ATmega1280 microcontroller på et board med headerudgange, komponenter til USB kommunikation og meget andet. Disse arduinoer kan programmeres ved hjælp af sin egen IDE med en simpel programmeringssyntax. Derfor blev det besluttet at anvende Arduino eller den kinesiske kopi Seeduino med samme specifikationer. Det blev overvejet at anvende en Arduino MEGA med en AT1280 CPU til både at indeholde applikationen og afvikle denne, men det viste sig hurtigt at denne MEGA havde for få outputs til at behandle kommunikationen til 24 displaydrivere. Derfor blev der besluttet at anvende flere Arduino'er. Der blev overvejet at opsætte et signalbehandlingssystem bestående af 3 af disse microcontollere. En arduino MEGA som "Master" til at modtage signal fra elevinput og til at indeholde applikationen og lade 2 slaver om at afvikle applikationen ved, at sende impulssignaler til display-driverne. Dette ville være en passende opsætning da man kan anvende 12 outputs på hver slave. Da disse microcontollere skulle kommunikere med applikationskontrollen og kommunikere internt i signalbehandlingen blev der overvejet at bruge RS232 som kommunikation til dette. RS 232 er nærmere beskrevet i sektion 6.3.

### 4.4 Dataopsamling- og behandling

Dataopsamlingen og -behandlingen er den del, der står for at modtage information fra signalbehandlingen. Der er tale om information i form af et pointsystem, der skal afspejle elevernes evne til at respondere på information. Se desuden sektion 3.5 (s. 11) for en beskrivelse af kravene til denne del. Der er naturligvis mange forskellige måder at lagre denne data på, men for at mindske det nødvendige arbejde og for at gøre det nemmest muligt at udvikle og administrere, anvendes som udgangspunkt en PC. I sektion 4.1 (s. 13) er det desuden specificeret, at en PC med en C#-applikation anvendes. C# er et højniveau-programmeringssprog, og der findes mange lettilgængelige libraries, hvori funktioner til mange forskellige formål kan findes.

#### 4.4.1 CSV-fil

En måde at administrere data på er med en simpel CSV-fil, hvori data lægges. CSV står for *Comma-Separated Values* - på dansk en kommasepareret fil. Et eksempel på en sådan fil kunne være:

```
Spillernavn,Spil,Score,
Mads,1,235
Mads,1,325
Mads,1,120
Jørgen,1,325
```

Denne opbygning kan sammenlignes med en tabel (eller et excel-dokument), hvor hver komma separerer en kolonne, og hver nye linje separerer en række. Man kan anvende denne slags CSV-filer, hvis man har brug for et let stykke software som blot skal være i stand til at administrere noget simpel data. Det er dog ikke så praktisk hvis man har store mængder data eller ønsker at lokalisere en enkelt celle, da man således først skal læse al foregående data igennem.

Siden en C#-applikation anvendes til applikationskontrollen, vil denne applikation også med fordel kunne anvendes til at administrere en CSV-fil og præsentere filens indhold på en graf. Dette løsningsforslag vil være i stand til at opfylde alle kravene til dataopsamling og -behandlingen - se sektion 4.4 (s. 15). CSV-filen har følgende fordele:

- Meget simpel og nem at administrere - kræver kun simpel fil I/O.

Men har dog følgende ulemper:

- Besværlig hvis store datamængder skal behandles.

#### 4.4.2 SQL-server

En anden - og meget populær måde - at lagre data på er at bruge en SQL-server. SQL står for *Structured Query Language*, og en SQL-server er en server, som består af en database, med et antal tabeller, som har nogle kolonner og nogle rækker. Ofte anvendes et sæt forskellige tabeller til at indeholde forskellige typer data, og nogle kolonner vil have til opgave, at specificere relationen imellem disse tabeller. På tabel 4.1 (s. 16) ses et eksempel på, hvordan samme data, som CSV-filen indeholder, kan præsenteres i en SQL-tabel.

| Spillernavn | Spil | Score |
|-------------|------|-------|
| Mads        | 1    | 235   |
| Mads        | 1    | 325   |
| Mads        | 1    | 120   |
| Jørgen      | 1    | 325   |

Tabel 4.1: Eksempel på en SQL-tabel

En SQL-server er meget praktisk, når store mængder data skal administreres, men kræver, at serveren installeres og sættes op på en computer. Desuden kræver det en del opsætning fra klientens side når

kommunikationen skal sættes op - det er derfor en fordel at anvende et højniveau-programmeringssprog som C#, hvor der findes let tilgængelige libraries til dette brug.

SQL-serveren har følgende fordele:

- Nem administration af store mængder data.
- Nemt at søge og udvælge enkelte datamængder.

Men har dog følgende ulemper:

- Kræver mere installation og opsætning af server.
- Kræver mere installation og opsætning af klient.

### 4.4.3 Valg af løsning

Siden C# anvendes, vil opsætning af og anvendelse af en SQL-server ikke være en større opgave - og der kan sandsynligvis være tale om store mængder data, hvis eleverne vil bruge spillet hyppigt. Derfor vælges en SQL-server, som vil være i stand til at lagre og behandle data på en overskuelig, praktisk og letanvendelig måde.

I resten af rapporten dækker begrebet dataopsamling- og behandling over SQL-serveren og en del af det tilhørende interface, som vil være kombineret med elementer fra applikationskontrollen.

## 4.5 Display

I det følgende gennemgås løsningsforslag til displayløsninger. Der var flere muligheder i opbygningen af displays og beslutningen blev taget på baggrund af fordele og ulemper ved de enkelte modeller samt et møde med CDH hvor lærerne udtrykte deres holdning til de enkelte løsningsmodeller. RGB-LED'er anvendes som lysgiver i hver af de beskrevne løsningsforslag - læs mere om LED i sektion 6.7 (s. 39).

### 4.5.1 8x8 matrix RGB LED-display

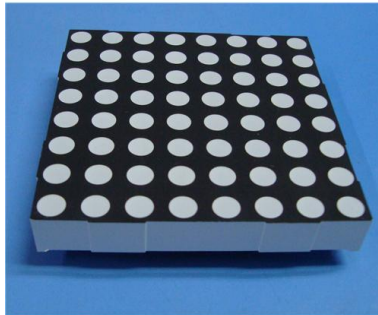
Et 8x8 RGB-LED setup overvejedes - dette ville bestå af 8 rækker med hver 8 RGB dioder i en færdigbygget ramme på 6x6 cm. I den umiddelbare planlægning lå, at denne 8x8 matrix skulle købes som en færdig model med driver fra en elektronikbutik. Fordelen ved sådan en løsning ligger i det grafiske layout med 64 dioder som:

- Ville kunne vise bogstaver og tal i alle farver.
- Ville kunne vise 7 kraftige farver.
- Ville give mulighed for at arbejde meget med programmering.

Ulemperne ved denne model ville være at:

- Den ville blive dyr at indkøbe.
- Størrelsen ville ikke være fleksibel.

Der ville pga. den færdigbyggede driver kunne produceres avancerede spil som kunne udfordre eleven. Denne løsning blev dog sorteret fra, da indkøbsprisen ville blive alt for høj. Desuden ville udviklingen af et helt system fra bunden være mere givende set i et studiemæssigt perspektiv. Se figur 4.2 (s. 18).



Figur 4.2: 8x8 matrix LED display - købes som færdigt produkt

#### 4.5.2 RGB 7-segment

Et display i stand til at vise tal ved hjælp af et RGB-display med mange dioder i hvert segment overvejedes. Dioderne ville placeres som et 7-segment og dioderne i de enkelte segmenter ville da være parallelforbundne, så de ville kun være i stand til at vise samme farve indenfor hvert segment. Se figur 4.3(a) (s. 19).

Fordele:

- De mange dioder ville kunne producere en høj lysstyrke til gavn for de svagtseende elever.
- Den ville kunne tal i forskellige farver.

Ulemper:

- Den ville være kompliceret at opbygge med de mange dioder.
- Den ville være tidskrævende at samle og programmere.
- Den ville være dyr med de mange dioder.

Et simplere løsningsforslag bygget på samme ide blot med færre dioder overvejedes.

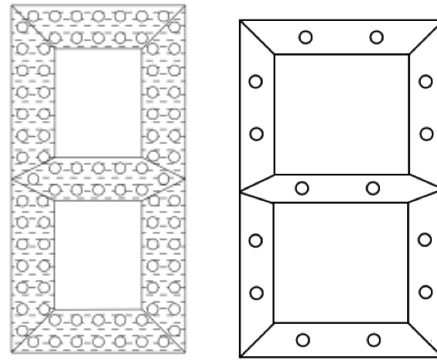
Denne model ville bestå af 7 segmenter med hver 2 dioder. Der ville i denne model være implementeret en afskærmning foran hvert segment i en mat acrylplade som ville bryde lyskeglen og sprede lyset til hele segmentet. Se figur 4.3(b) (s. 19).

Fordelen ved denne løsning frem for den anden 7-segment:

- Den ville være billigere at bygge.
- Den ville være hurtigere at konstruere med kun 2 dioder i hvert segment.
- Den ville være strøm- og elektronikbesparende .

Den avancerede opbygning i 7 segmenter vurderes til at være en klar fordel hvis lærerne på CDH ønsker at benytte sig af en model med tal og bogstaver. Uanset hvilken af disse 2 løsninger der skulle arbejdes med ville det kræve en del i både opsætning og programmering. Denne model blev taget med i løsningsforslag til mødet m. CDH.

En færdigbygget 7-segment model overvejedes også. Denne model var relativt billig, men kunne dog kun vise en rød farve og var derfor ikke passende til formålet.



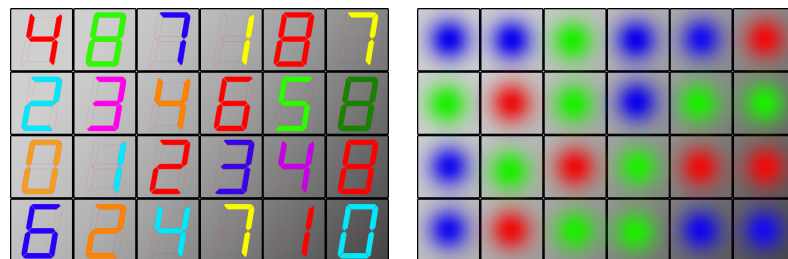
(a) 7-segment display opbygget af løse dioder  
(b) 7-segment display opbygget af løse dioder med plexiglas afskærmning

Figur 4.3: Illustration af de to forskellige 7-segment displays

### 4.5.3 Farvede felter

I forlængelse af idéen med plexiglasafskærmning foran lysdioderne overvejedes en version af 24 felter som kunne vise alle farver ved brug af RGB dioder. Felterne skulle være opbygget af ca. 16 RGB-dioder med plexiglasafskærmning foran. Der ville derfor være mulighed for at hvert felt kunne vise hver sin farve i hele det synlige farvespektrum. Disse felter ville desuden kunne kontrolleres i forhold til lysstyrke og vise kraftige farver. På denne måde kunne man blænde op eller ned for lyset efter behov. Se figur 4.4(b) (s. 19).

Fordele ved denne model:



(a) Eksempel på 24 felter opbygget med 7-segment displays  
(b) Muligheder for visning af forskellige farvefelter

Figur 4.4: Illustration af felter m. 7-segment displays (a) og forskellige farvefelter (b)

- Den ville være forholdsvis simpel at konstruere, så produktet kunne produceres færdigt inden for en realistisk tidshorisont.
- Den ville være simplere at programmere.
- Der ville kunne gåes i dybden med spillenes formål, spiludviklingen og stimuleringen i forhold til elevernes behov.

Ulemper:

- Den ville kun være i stand til at vise farver.

Denne version havde større fordele i forhold til de andre modeller. Modellen åbnede pga. sin simplicitet op for mange muligheder for forskellige spilvariationer kun ved brug af farver. Da spil i denne model ville være forholdsvis simple at programmere kunne CDH få mulighed for stor indflydelse på design af spil med det formål at optimere stimuleringen af brugerne. Denne model blev prioriteret højt i ideudviklingsfasen i projektgruppen.

#### 4.5.4 7-segment med RGB-baggrundslys

Det overvejedes at konstruere en model med en baggrundsbelysning som nævnt i sektion 4.5.3 (s. 20) og med en front med den tidligere nævnte, færdigbyggede 7-segment der giver mulighed for at vise et tal i rødt.

Fordelene ville være at:

- Der indgår både tal og farver.

Ulemper:

- Det ville være avanceret at opbygge pga. de 2 systemer sammen, specielt i forhold placeringen i forhold til hinanden.
- Det ville sandsynligvis være svært at skelne baggrundslyset og tallets farve fra hinanden

Denne løsning ville give mulighed for at lave mere komplekse spil. I stedet for kun at vise farver, ville man kunne trykke på forskellige tal kombineret med forskellige baggrundsfarver. Denne løsning medbringes til mødet m. CDH. Den blev der klassificeret som værende for avanceret i forhold til elevernes niveau. Se figur 4.5 (s. 20).

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 3 | 8 | 1 | 2 | 8 |
| 0 | 4 | 3 | 9 | 6 | 2 |
| 5 | 2 | 4 | 7 | 0 | 8 |
| 6 | 9 | 8 | 5 | 9 | 4 |

Figur 4.5: Muligheder for visning af forskellige tal og farver

#### 4.5.5 Valg af løsning

Løsningen med de farvede felter vælges som den ideelle løsning på baggrund af mødet med CDH. Dette skyldes, at løsningen giver gode muligheder for udvikling af spil. Løsningen vil samtidig passe godt til eleverne hos CDH, da de har et niveau svarende til børn op til 5 klasse.

Da projektet er afgrænset til en prototype af systemet, bliver felterne ikke lavet med 64 dioder, men i stedet med én RGB diode. Denne løsning giver mulighed for at vise 7 farver, og kan derfor simulere den færdige løsning.

## 4.6 Display-driver

Her gennemgås løsningsforslag til en driver til det valgte display. Da der er et krav om at displayet skal være i stand til at kommunikere med et signal / ledningsføring bliver det nødvendigt med en driver. Driveren vil fremover blive kaldt kaldt RGB-controller, da RGB-dioder er valgt som display.

### 4.6.1 Arduino/Seeeduino

Det første løsningsforslag byggede på at lave et fuldt system udelukkende af færdige microcontrollere som Arduinoer og Seeeduinoer. Dette var en nem løsning rent konstruktionsmæssigt, men det var også en meget dyr løsning. Fordelen ved denne løsning var muligheden for at fokusere på programmering og konstruktion af displays.

Løsningen med at benytte sig af 24 Arduinoer eller Seeeduinoer var dog ikke en økonomisk mulighed. Der blev undersøgt en Rainbowduino, en specialbygget microcontroller til styring af en 8x8 RGB matrix, men denne havde en effektbegrænsning på udgangen, således den ikke kunne levere nok strøm til at forsyne en kraftig lyskilde. Derfor blev denne løsning også fravalgt. Den mest optimale løsning måtte være at konstruere en driver som en programmerbar kreds.

### 4.6.2 Programmerbarkreds

For at opfylde kravene til displaydriveren fra sektion 3.6 (s. 12) blev der udvalgt 2 microcontrollere. En ST7 microcontroller og en ATTiny261. Disse microcontrollere blev valgt fordi de begge opfyldte kravene om at kunne modtage et signal med pulser og derudfra styre RGB-dioderne med 3 PWM signaler.

Fordelene ved ST7 er:

- Den er billig.
- Den opfylder kravene meget præcist da den har lige det som skal bruges.

Ulemperne ved ST7 er:

- Den kræver en opstillet kreds således den kan programmeres.
- Den skal programmeres i ret kompleks c-kode.

Fordelene ved ATTiny261 er:

- Den er nem at programmere da der er et AVR udviklingsboard til rådighed.
- Den skal kodes i BASIC på et forholdsvist simpelt niveau.

Ulemperne ved ATTiny261 er:

- Den er lidt dyrere end ST7.
- Den har en del funktioner som ikke er nødvendige.

På baggrund af dette valgte blev ATTiny261 valgt, da den var nemmest tilgængelig for os.

### 4.6.3 RGB styring

Det er den programmerbare kreds som bestemmer om RGB-dioderne skal have strøm, men da microcontrollerne max kan levere 40mA (12.4) per ben er det ikke en mulighed at forbinde dem direkte da det ville være i strid med kravene til displaydriveren som kan findes i afsnit sektion 3.6 (s. 12)

Til gengæld kan man sagtens, med en lille strømstyrke på f.eks. 10 mA, styre om en større strømstyrke skal løbe. Dette kan gøres ved at anvende en transistor. Hertil findes der mange forskellige typer transistorer. I dette projekt er primært blevet fokuseret på 2, BJT'er og MOSFET'er. Teorien om transistorer er uddybet i kapitel 6.6 (s. 37)

En af de mest nævneværdige forskelle imellem disse to typer transistorer er at BJT'er åbnes vha. strøm og MOSFET'er åbnes vha. spænding. En MOSFET er desuden simpel at opsætte da den forbindes direkte med microcontrolleren og derefter "åbner op" når microcontrolleren regulerer spændingen på det forbundne I/O ben hvilket gør at en MOSFET nemt kan fungerer som en switch. En BJT er til gengæld billigere og er god til at behandle PWM signaler sektion 6.2 (s. 27) hertil skal det dog siges at en MOSFET er bedre til at håndtere store strømstyrker.

### 4.6.4 Valg af løsning

Som displaydriver er det valgt at bruge ATTiny261 microcontrolleren på trods af at den kostede lidt mere end ST7'eren og den havde nogle funktioner som ikke var nødvendige. Dette valg skyldes at syntaxen er simplere og selve programmeringsprocessen kan gøres på det førnævnte AVR udviklingsboard. Generelt set er ATTiny261 microcontrolleren også bedre understøttet og det gør at det er nemmere at finde data til den og informationer om den. I sammenligningen af Rainbowduino og ATTiny er ATTiny den bedste løsning da den er langt billigere. Desuden er det ikke nødvendigt at et felt kan lyse flere forskellige farver samtidigt eller at det består af en 8x8matrix som de dioder der følger med Rainbowduino.

Til RGB-styring valgtes MOSFETs pga. den store strømstyrke de kan holde til samt at det ikke kræver en stor strømstyrke at åbne dem. Desuden er MOSFETs meget hurtige til at åbne og lukke hvilket betyder at vi kan bruge dem til at puls vidde modulere den spænding som forsyner dioderne. Ulempen ved dette valg er at MOSFETs er dyrere end BJT'erne, men da det er et klart krav at systemet skal kunne klare høje strømstyrker faldt valget på MOSFETs.

# Kravspecifikation

Efter at have fastlagt en løsning kan der opstilles specifikke krav til produktet. Disse krav beskriver hvordan interaktionen som foregår imellem produktet og brugeren skal udformes, således produktet opnår den ønskede brugervenlighed. Grunden til at disse krav opstilles skyldes at det er utrolig vigtigt at produktet ikke er for komplekst til målgruppen hvilket ville forårsage tab af nytte og accept.

Krav til anvendelsen af produktet:

- 1 En elev eller person skal kunne ”spille” et spil med det samme efter at reglerne er blevet forklaret.
- 2 En almindeligt fornuftig person skal kunne vælge og igangsætte spil fra computeren via USB efter at have set et eksempel med kort instruks.
- 3 Produktet skal kunne måle, sende og lagre data omkring spilleren til et stykke software på PC’en via USB.
- 4 Produktet skal kunne behandle spillerdata og vise/udskrive grafer som en almindeligt fornuftig person kan forstå uden forklaringer.
- 5 Prototypen skal kunne opskaleres således den i teori og praksis kan styre en færdig ramme med 24 felter.
- 6 Prototypen skal kunne vise 7 forskellige farver på alle 24 dioder
- 7 Prototypen skal kunne holde til at levere 1A.
- 8 Prototypen skal kunne transmittere til displaydriveren med et PWM signal.
- 9 Prototypen skal kunne kommunikere med applikation på PC via RS232.
- 10 Prototypen skal kunne reagere på inputs fra 24 trykknapper

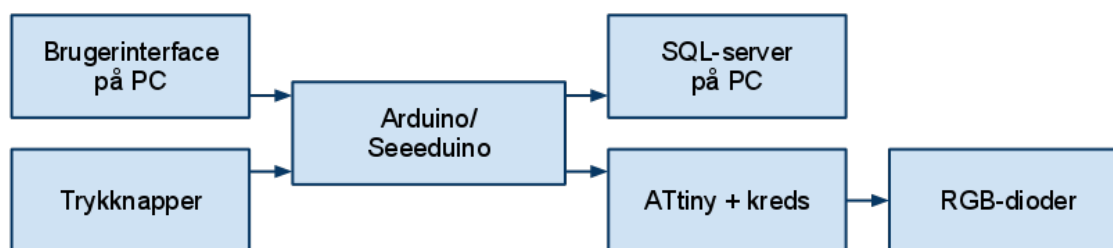
## 5.1 Accepttest specifikation

For at undersøge om produktet opfylder de krav som er stillet i kravspecifikationen planlægges en testfase. I testfasen ønskes en vurdering af produktet i forhold til brugervenligheden vha. brugertests. Det skal hertil siges at produktet som testes er en prototype og derfor ikke udgør det færdige produkt hvorfor der også fokuseres primært på det elektroniske system.

Brugertesten foregår således at en række tests udføres med forskellige brugere hvortil en testleder tager notater. Disse notater danner senere grundlag for vurderingen af produktet og vil blive holdt op imod kravspecifikationen hvorpå en konklusion kan drages. Hvis brugertesten imidlertid ikke gav tilfredsstillende resultater vil softwaren blive rettet til således nye tests, med forhåbentlig bedre resultater, kan udføres.

## Anvendt teori

På baggrund af de valg der er taget omkring opbygningen af en prototype, vil der i dette kapitel blive beskrevet den anvendte teori. Det vil blive beskrevet på et brugervenligt niveau, hvor enkelte emner vil blive beskrevet mere dybdegående.



Figur 6.1: Diagram over system

På figur 6.1 (s. 24) vises et systemdiagramet hvor komponenterne er indsat. Arduino/seeeduino erstatter signalbehandling, ATtiny + kreds erstatter display driver og display bliver erstattet af RGB-dioder.

### 6.1 Microprocessorer og IC'er

For at læse dette afsnit kræver det en definition af disse begreber:

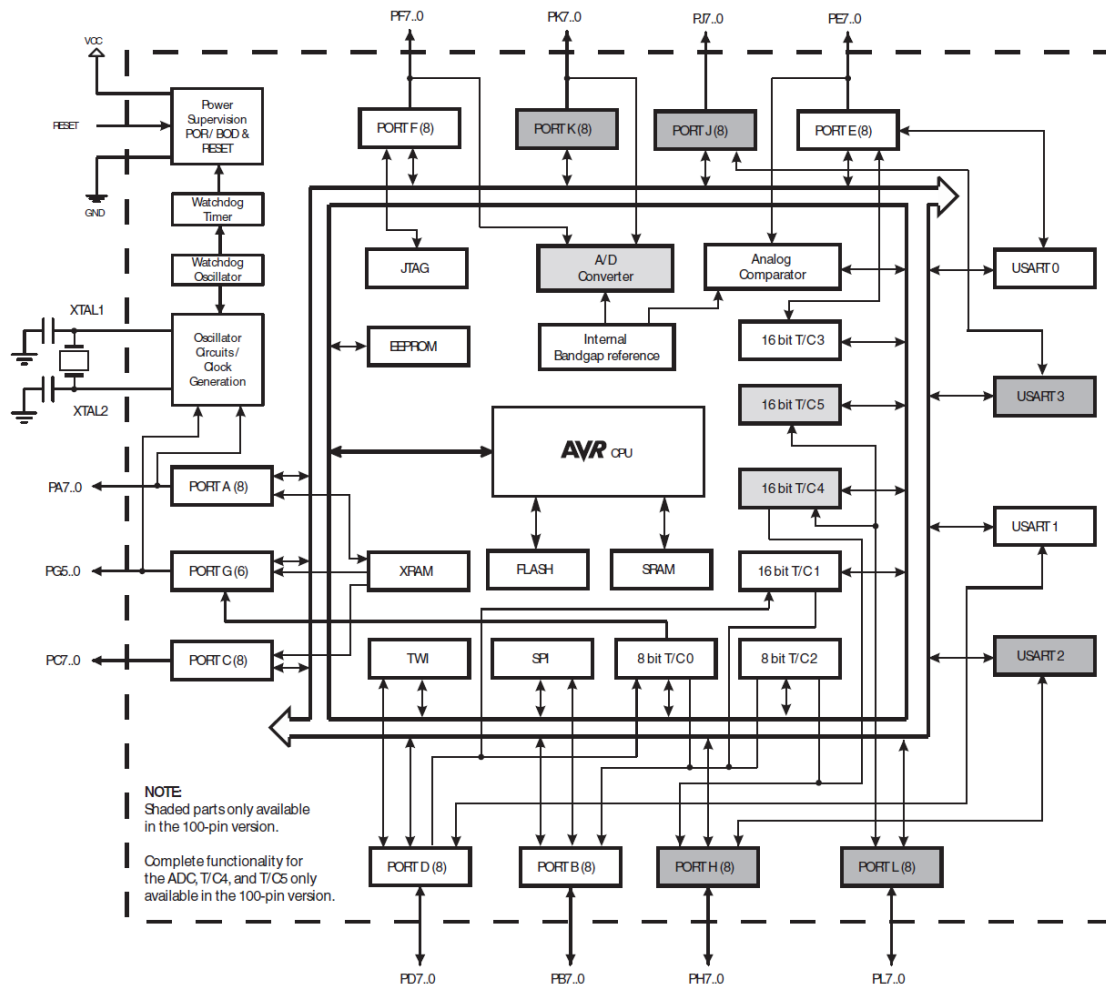
**IC** står for Integrated Circuit (integreret kredsløb), og består af flere komponenter og kredse, samlet i en plastik casing. F.eks. er ATtiny'en en IC bestående af bla. transistorer.

En **microprocessor** er en CPU (Central Processing Unit). Den afvikler instrukser, som ligger på en flash hukommelse. Den står for at udføre elementære aritmetiske, logiske og in/output operationer.

Kilde: "A Third Survey of Domestic Electronic Digital Computing Systems" [24]

En microcontroller er i stand til at styre et elektronisk system fordi den er opbygget af en række delelementer, der hver sørger for at varetage en bestemt funktion. Den indeholder en CPU til at foretage beregninger og elementære regneoperationer. Den indeholder også ROM (Read Only Memory) til lagring af applikationen. Den indeholder også typisk RAM (Random Access Memory) som er hukommelse til data og desuden et antal input og output porte til at kommunikere med omverdenen. Afhængigt af hvilken funktion, microcontrolleren er bygget til at varetage, vil den indeholde nogle forskellige delelementer - her kan bl.a. nævnes A/D converter, EEPROM og USART.

På figur 6.2 (s. 25) ses et eksempel på et blokdiagram, som det fremgår af *ATmega1280 databladet* [4, s. 5]. Der er her tale om en ATmega1280-microcontroller af samme type, som sidder på en Arduino Mega eller Seeeduino Mega. Den har altså mange forskellige delelementer, der alle er styret af AVR<sup>1</sup> CPU-en.



Figur 6.2: Blokdiagram over ATmega1280 [4]

De vigtigste elementer i projektforsøget er:

- AVR CPU
- Flash
- SRAM
- EEPROM
- PORTA..H
- A/D Converter
- 8/16bit T/C0..5 (Timer/Counter)
- USART 0..3

<sup>1</sup>Microcontroller-serie fra producenten Atmel

*AVR CPU'en* står som nævnt for at modtage data fra hvert element, behandle data, samt styre hvordan elementerne agerer. Det gør den ved hjælp af 32 registre (standard for alle AVR microcontrollere), som er forbundet til et "Arithmetic Logic Unit" (ALU), der udfører sammenligninger mellem information i de forskellige registre.

Disse registre indeholder desuden information om hvordan ovennævnte delelementer skal agere - man kan derfor, ved at ændre på indholdet af registrene, opsætte og konfigurere microprocessorens adfærd. Selve programmet, som udføres på CPU'en, ligger på *Flashen*, som eksempelvis kan bestå af en 128K bytes hukommelse. Der er tale om en speciel type EEPROM (Electrically Erasable Programmable Read-Only Memory), hvor data skrives til eller slettes fra i større blokke. Eftersom der er tale om en EEPROM er flashen begrænset til et antal skrivninger - antallet på ATmega1280 er 10.000. **Bemærk** at flashen *ikke* er det samme som EEPROM'en - EEPROM'en som refereres til senere i dette afsnit anvendes udelukkende til lagring af variabler, mens flashen anvendes til lagring af programmet, som eksekveres i microprocessoren.

For at gemme forskellig data, som i softwaren refereres til som variabler, anvendes *SRAM* (Static Random Access Memory). Forskellen mellem Flashen og SRAM'en er, at data i Flashen gemmes i såkaldt "Non-Volatile" hukommelse, så al data, der gemmes deri *ikke* bliver slettet, når strømmen kobles fra. Dette er ikke tilfældet med SRAM (som er "Volatile" hukommelse) - al data heri forsvinder, når spændingsforsyning frakobles. ATmega1280 har en 8KB SRAM.

Ønsker man at gemme forskellige variabler i sin software, selvom controlleren slukkes, bruger man således i stedet den EEPROM, microcontrolleren er udstyret med. På ATmega1280 sidder en 4KB EEPROM dedikeret til data, som skal gemmes, når strømmen afbrydes. Det er vigtigt at bemærke, at en EEPROM har et begrænset antal gange, man kan skrive til den - for ATmega1280 er dette tal 100.000.

Selve kommunikationen med andre elektroniske komponenter fungerer ved hjælp af microcontrollerens porte. Som det ses på figur 6.2 (s. 25), har ATmega1280 otte porte, angivet PortA, PortB, ..., PortH. Hver af disse porte afspejles i hver sit register, bestående af otte bit (en byte) i controlleren. Dette betyder således, at sættes en bit høj i registret, vil dette afspejle sig på den pågældende pin, som tilsvarende går høj (typisk 5V).

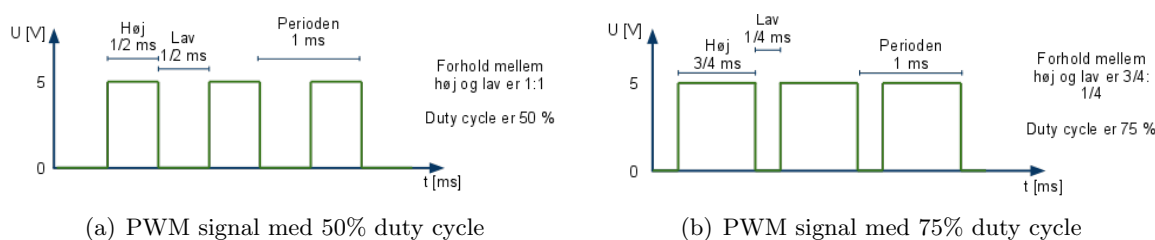
A/D Converteren (Analogue to Digital Converter) sørger for at konvertere en analog spænding til et digitalt signal. En beskrivelse af denne kan findes i sektion 6.4 (s. 29).

Udover A/D Converteren har microcontrolleren også en række andre "hjelpeværktøjer". Der er her tale om en række timers og U(S)ARTS. Timerne kan varetage en række forskellige funktioner, såsom måle tid, bruges til interrupts og PWM, se sektion 6.2 (s. 27). U(S)ARTS kan styre seriel kommunikation såsom RS232 - se sektion 6.3 (s. 28).

Reference til ovenstående information findes i *ATmega1280-databladet* [4] og *Flash Memory Cells - An Overview* [20].

## 6.2 PWM

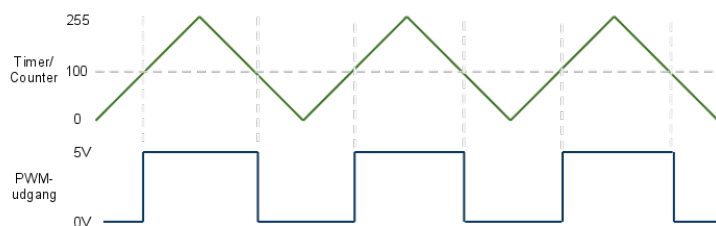
Når man ønsker at nedsætte hastigheden af en motor eller intensiteten af lyset fra en lysdiode, kan man naturligvis nedsætte strøm/spændingsforsyningen til motoren eller dioden. Dette har dog en del ulemper - det er mere effektivt at slukke og tænde meget hurtigt for enheden. Forestiller man sig, at en motor forsynes med en spændingsforsyning som illustreret på figur 6.3(a) (s. 27), ville motoren slukke og tænde så hurtigt, at den ikke når at slukke og tænde helt, men blot vil køre med halv hastighed.



Figur 6.3: Eksempler på to forskellige duty cycles med PWM

På tilsvarende måde vil en diode forsynet med en spænding som vist på figur 6.3(b) (s. 27) lyse med  $3/4$  lysstyrke. *Duty cycle* betegner således procentdelen af tiden, signalet er højt. Bemærk at *perioden* er konstant, og det kun er duty cyclen som ændres. Mange microcontrollere er udstyret med hardware til at skabe et PWM-signal. Som del af denne indgår en Timer/Counter.

I softwaren specificeres en værdi, som bestemmer hvordan timeren skal agere, og igennem denne hvor stor duty cycle signalet har. Eksempelvis har ATmega1280 en timer som kan sættes til at køre i 8-bit mode - dette betyder, at en Timeren tæller fra 0-255. Konfigureres timeren til "Phase and frequency correct PWM", se sektion 8.3 (s. 64), vil timeren tælle fra 0 til 255, og derefter fra 255 til 0. Hver gang timeren rammer den værdi, der er blevet specificeret, sættes PWM-udgangen til at skifte mellem lav-høj eller høj-lav.



Figur 6.4: 8-bit timer og PWM-udgang v. sammenligningsværdi på 100

På figur 6.4 (s. 27) ses et eksempel på hvordan en 8-bit timer kan tælle og ændre på en tilhørende PWM-udgang, hvis værdien, som sammenlignes med sættes til 100. Her vil timeren således lave 310 tællinger, mens PWM-udgangen er høj, og 200 tællinger mens PWM-udgangen er lav. Dette vil give en duty cycle på  $\frac{200}{310} \approx 64,5\%$ .

Microcontrolleren kan naturligvis køre med forskellige frekvenser, og afhængigt af denne vil timeren også tælle med forskellige hastigheder. Hvis man således ønsker, at en timer ikke skal tælle lige så hurtigt som microcontrolleren (så pwm-udgangens frekvens kan ændres), kan man sætte timerens *prescale*. Sættes eksempelvis timerens prescale til 8, vil timeren kun tælle hver 8. svingning på microcontrolle-

rens clock. Ved en prescale på 8 og en clock på 8MHz, vil frekvensen på PWM-udgangen således blive  $\frac{8MHz}{8} = 1MHz$ .

Sættes et PWM-signal til tilstanden **Clear up**, vil PWM-udgangen således *cleares* (gå lav), når timeren tæller op og rammer værdien, mens **Clear down** betyder, at PWM-udgangen *cleares* når timeren tæller nedad. I projektet anvendes et PWM-signal bla. til kontrol af RGB-dioder, så lysstyrken af hhv. den røde, grønne og blå diode kan justeres, og alle mulige farvekombinationer kan skabes - dette uddybes i *Design af Hardware*, sektion 7.3 (s. 44). Et PWM-signal anvendes desuden til kommunikation mellem Slaver og RGB-controllere - dette uddybes i *Design af Software*, sektion 8.3 (s. 64)

Baggrundsviden til ovenstående er fundet i *An Introduction to Practical Electronics, Microcontrollers and Software Design* [3].

## 6.3 RS232

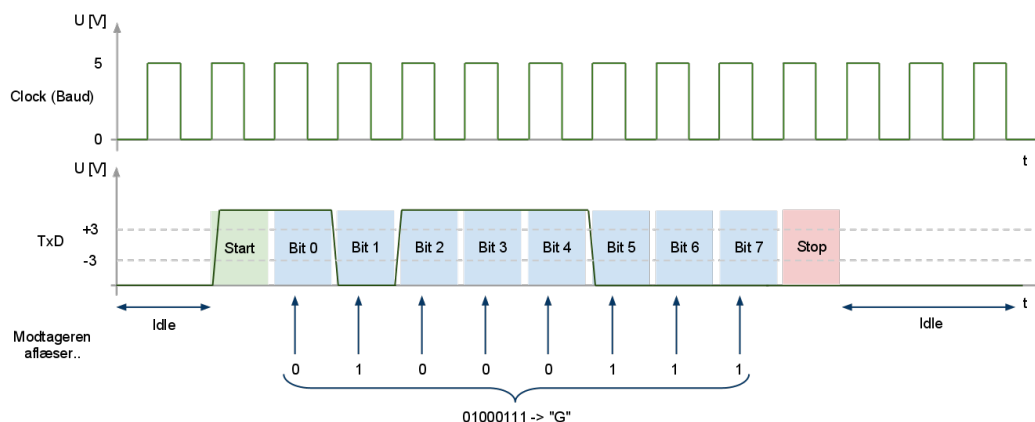
RS232 er en standard indenfor seriel datatransmission. Standarden fastsætter en række parametre - herunder ses et udsnit af dem, som det fremgår af *Interfacing the Serial/RS232 Port* [21].

- Logisk 0 ligger mellem +3 og +25 Volt
- Logisk 1 ligger mellem -3 og -25 Volt
- -3 til +3 Volt er ikke defineret
- En kortslutning må aldrig overstige 25V (med GND som reference)
- En kortslutning må aldrig overstige 500mA, og driveren skal ikke tage skade ved en sådan kortslutning

Mange microcontrollere har to pins til at administrere denne kommunikation - disse navngives typisk TxD og RxD. Der findes desuden en række pins som ofte anvendes til RS232-kommunikation, men de anvendes ikke i projektet, da de ikke er nødvendige for simpel kommunikation.

Seriel kommunikation efter RS232-protokollen er en *asynkron* kommunikation, hvilket betyder, at der ikke sendes en clock med fra senderen til modtageren. Derfor skal kommunikationshastigheden være predefineret - dette gøres ved at definere en *baud* (et antal bit pr. sek). Et eksempel på seriel kommunikation vha. RS232 ses på figur 6.5 (s. 29). Afsenderen sætter den ønskede bit i TxD når clocken går fra lav til høj - modtageren aflæser således på RxD en smule efter dens clock går fra lav til høj for at kompensere for den tid, det kan tage før TxD bliver høj.

Mange microcontrollere har en intern UART (Universal Asynchronous Receiver/Transmitter), som sørger for at sende og modtage data med den rigtige baud. På transmittersiden vil UART'en læse en række bits (typisk en byte) fra microcontrollerens transmitbuffer. En start-bit sendes, derefter de forskellige databits, og til slut en eller flere stop-bits. På receiversiden vil UART'en således gøre klar til datamodtagelse når en start-bit modtages, og hver databit opsamles og gemmes i et buffer. Når en slut-bit modtages kan UARTen kalde et processor interrupt, så microcontrolleren kan indlæse dataen fra UARTens buffer. Det er vigtigt at bemærke, at UART'en ikke styrer spændingsniveauerne, men en anden komponent er nødvendig for at opnå RS232-standardens spændingsniveauer på +3/+25V og -3/-25V.



Figur 6.5: Eksempel på RS232-kommunikation hvor et G sendes

Kommunikationen som ses på figur 6.5 (s. 29) er formateret ved hjælp af ASCII (American Standard Code for Information Interchange), som oftest anvendes ved seriel kommunikation. Eksempelvis er 'G' således decimaltallet 71, altså binært 01000111.

## 6.4 ADC - Analogue to Digital Conversion

Når man taler om analog til digital konvertering menes der at man konverterer et analogt (kontinuert) signal til et digitalt signal.

Et analogt signal er kontinuert og ville kunne beskrives som et decimaltal med et uendeligt antal decimaler. Dette signal omsat til digitalt er ikke kontinuert, da det består af binære tal og er begrænset af hvor mange bit der er til rådighed under konverteringen. Det analoge og det digitale signal vil derfor aldrig være 100% identiske. For at lave sådan en konvertering benytter man sig af en ADC.

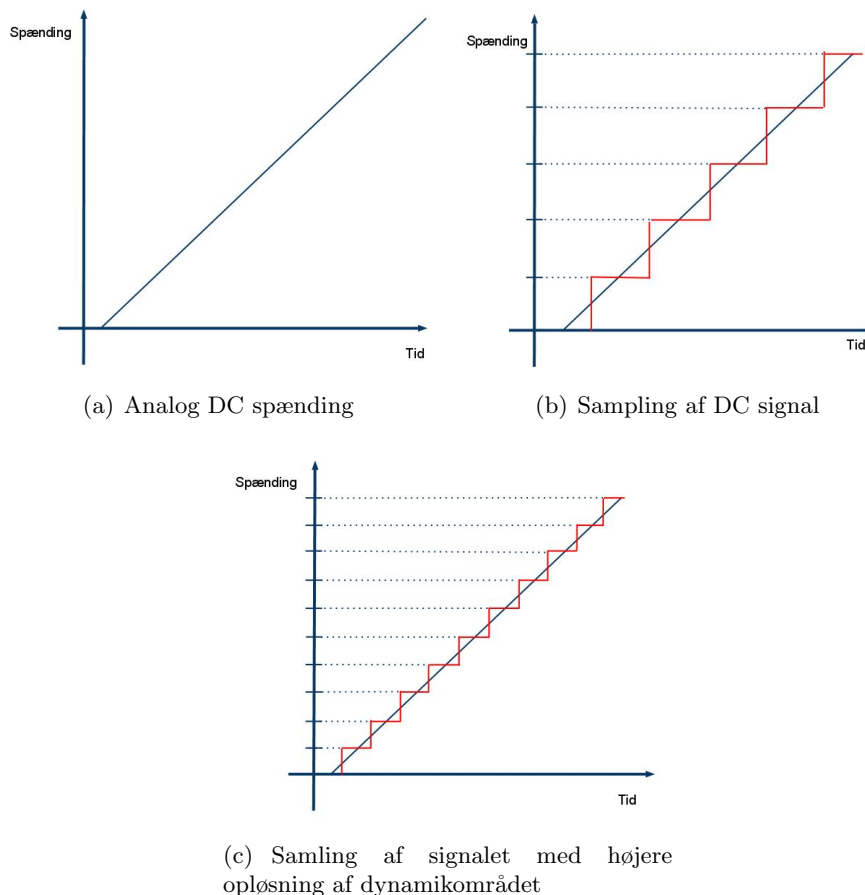
AD converteren vil aflæse og omskrive signalets værdi hver gang et givent tidsinterval er passeret. Denne tidsdiskretisering er bestemt af ADC'ens samplingsfrekvens. Jo højere samplingsfrekvens, jo flere aflæsninger. Disse aflæsninger bliver så omskrevet til binære tal som beskriver dynamikområdet i det analoge signal. Præcisionen i dynamikområdet er derfor bestemt af ADC'ens bitantal. Derfor vil en hyppig aflæsning (høj samplerate) af et analogt signal ikke direkte svare til en mere præcis gengivelse af det analoge signal, men forhøje muligheden for en præcis gengivelse fordi ADC'en foretager flere målinger i sekundet.

Det skal hertil nævnes at hvis ADC'en foretager for mange unødvendige målinger vil man modtage store mængder ubrugelige data og det er derfor vigtigt at tilpasse samplefrekvensen til signalet som modtages. Dette kan gøres ved at lave en frekvensanalyse på signalet hvorefter en passende samplefrekvens kan udregnes. Det bør nævnes, at en teoretisk perfekt rekonstruktion af et signal kun kan lade sig gøre, hvis samplingfrekvensen som minimum er større end det dobbelte af den maksimale frekvens, som det ønskes AD-konverteret. Dette kaldes Nyquist-Shannons samplingsteori [23].

Et højere bitantal vil samtidig give større mulighed for en præcis gengivelse af den aflæste værdi i dynamikområdet. F.eks. vil en 24bit ADC give i alt  $2^{24}$  (altså 16.777.216) mulige trin i dynamikområdet,

hvilket giver mulighed for en præcis inddeling af signalet.

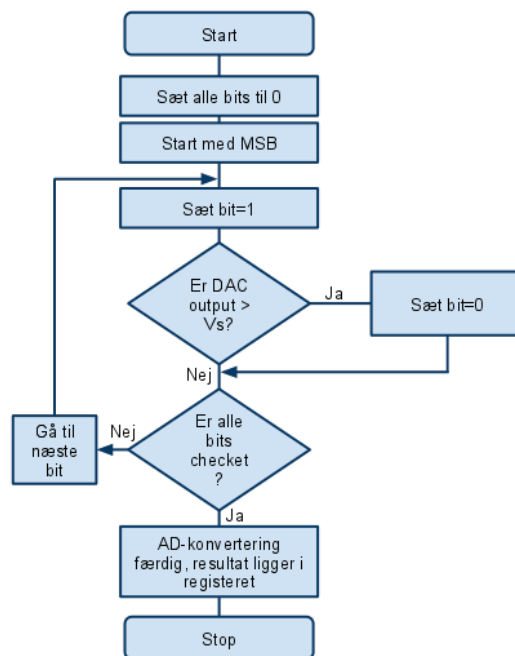
Jo højere bitantal desto større præcision i den digitale gengivelse af det analoge signal. Eksempler på dette kan ses på figur 6.6(a), (b) og (c) (s. 30).



Figur 6.6: Eksempler på forskellige bitantal

På figur 6.6(a) (s. 30), kan man se det analoge signal, som ønskes konverteret, med blåt. På (b), ser man hvordan ADC'en udfører målinger(samplinger) med rødt og hvordan spændingen bliver aflæst og placeret som en binær værdi i hver nærmeste "trin". På (c), bliver der foretaget samme samplinger af det samme signal men her er antallet af "trin" fordoblet således at dynamikområdet har en dobbelt så høj opløsning hvilket giver en mere nøjagtigt inddeling.

Der er mange forskellige måder hvorpå man i praksis kan udføre en AD-konvertering - en af de mest anvendte er kendt som *Successive Approximation ADC*. Denne metode fungerer efter princippet som vist på figur 6.7 (s. 31). Til eksempelvis en 10-bit ADC anvendes 10 bit i et register. Startende fra MSB (Most Significant Bit) sættes denne bit til 1, en DAC (Digital to Analogue Conversion) udføres af hele registret, og resultatet af denne DAC sammenlignes med spændingen det ønskes at konvertere ( $V_S$ ). Er resultatet af DAC'en højere end  $V_S$  cleares denne bit - proceduren gentages indtil alle bits i registret har været anvendt til sammenligningen. Denne information stammer fra *HyperPhysics* af Dr. Rod Nave [17].



Figur 6.7: Successive Approximation ADC

Der bliver i dette projekt ikke anvendt ADC, men derimod fortolkninger af PWM signaler som på sin vis ville kunne opfattes som en AD konvertering.

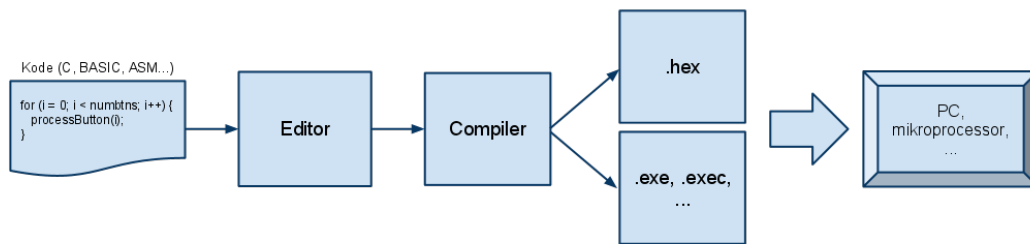
## 6.5 Software

I projektet er der udviklet software til forskellige formål. Herunder software til PC, Arduinoer/se-eduinoer og RGB controllere. I de følgende afsnit forklares den anvendte softwareteori. Der findes en lang række forskellige programmeringssprog, hvori man kan skrive kode til microprocessorer - heriblandt kan C, ASM og BASIC nævnes. Fælles for processen, uanset hvilket som helst sprog man benytter sig af, er, at koden skrives i en editor, kontrolleres af en debugger, kompileres af en compiler, og til sidst overføres til og afvikles på en microprocessor. Princippet kan ses på fig. 6.8. Ofte kan man med fordel anvende softwarepakke der indeholder både en editor, debugger og compiler i et - en sådan softwarepakke er kendt som en IDE (Integrated Development Environment)[18].

For at programmere flashen i en microcontroller anvendes et specielt, hexadecimalt format - dette repræsenteres typisk med filtypen .hex, og i de følgende afsnit vil dette hex-format betegnes sådan.

Et program på en PC afvikles efter samme princip - her kompileres filen blot til en eksekverbar fil, som operativsystemet på den pågældende platform er i stand til at afvikle. Herunder følger en kort beskrivelse over de kommandoer og den syntaks, der normalt anvendes i de forskellige programmeringssprog, som er anvendt i dette projekt.

Når man arbejder med software på microprocessorer anvendes ofte en række forskellige udtryk for det



Figur 6.8: Fra kode til program

|                                           |                      |               |
|-------------------------------------------|----------------------|---------------|
| Logik/Boolean                             | TRUE (Sandt)         | FALSE (Falsk) |
| Binær                                     | 1                    | 0             |
| Typisk brugt når man refererer til en pin | HIGH (Høj)           | LOW (Lav)     |
| Spænding på pin v. HIGH/LOW               | 5V (eller $V_{cc}$ ) | 0V            |

Tabel 6.1: Forskellige udtryk, der ofte anvendes indenfor software til microprocessorer

samme - derfor ses på tabel 6.1 (s. 32) et skema med en beskrivelse af en række ord, der vil blive brugt hyppigt i resten af kapitlet. De anvendes ofte som synonyme for hinanden.

**Bemærk** at al software i resten af rapporten vil præsenteres på engelsk for fuld kompatibilitet med udviklingsværktøjer og nemmere videreudvikling for eventuelle udenlandske programmører.

### 6.5.1 C

I dette projekt er C primært anvendt som basis for den software, der er udviklet til styringen af en Arduino/Seeeduino. Ved købet af en Arduino/Seeeduino medfølger en "Arduino-IDE, som giver programmøren adgang til en lang række predefinerede funktioner, der gør udviklingsprocessen betydeligt nemmere.

På denne måde bliver syntaksen i et arduino-baseret projekt meget lig almindelig C-syntaks, men er dog forskellig fra anden C-kode rettet mod microprocessorer, da programmøren sjældent skal arbejde med mikroprocessorens registre.

Arduino'en tager udgangspunkt i en `setup()`-funktion, hvori eventuel konfiguration opsættes - her konfigureres porte til at varetage eventuelle funktioner, eller timers, interrupts og andre features opsættes. Efter denne `setup()`-funktion er udført, vil microprocessoren fortsætte til en `loop()`-funktion, der (som navnet antyder) vil gentages indtil microprocessoren slukkes.

Denne adfærd er typisk for et embeddede system, da et sådant system er dedikeret til at varetage en funktion, og afhængigt af forskellige parametre og forskellige input varierer eventuelle output.

Her følger en beskrivelse af nogle basale funktioner og syntaks.

```
1 // Comments are written like this
2 #define CONSTANT 123           // Defining a constant
3 void setup() {
4     pinMode(25, OUTPUT);        // Setting a pin to OUTPUT-mode
5     pinMode(24, INPUT);         // Setting a pin to INPUT-mode MARKER AT DISSE ER DIGITALE
6
7     int anInteger;              // Declaring a variable
8     pinMode(10, OUTPUT);        // Pin 2-13 can be used for PWM-output
9     analogWrite(10, 128);       // Sets a 50% dutycycle
10    Serial.begin(115200);        // Serial communication is set up
11 }
12
13 void loop() {
14     digitalWrite(25, HIGH);     // Writing to a pin
15     int status = digitalRead(24); // Reading from a pin
16
17     delay(1000);                // Pauses the processor for 1000ms
18
19     // A simple for-loop that loops 10 times
20     int i;
21     for (i = 0; i < 10; i++) {
22         Serial.print("i is now "); // Printing to the serial port
23         Serial.print(i);
24         Serial.print("\n");
25     }
26 }
```

Code snippet 6.1: A simple Arduino-C example

Der henvises desuden til sektion 8.2 for en beskrivelse af softwaren anvendt i dette projekt, hvor yderligere funktioner anvendes i en større sammenhæng. Der henvises desuden til *Arduino Language Reference* [5] for en beskrivelse af størstedelen af de tilgængelige kommandoer.

## 6.5.2 BASIC

I dette projekt er BASIC primært brugt til at udvikle softwaren til RGB-controlleren. Der er tale om en variant af det originale BASIC, som er tilpasset brug til microprocessor-softwareudvikling. I den sammenhæng er en IDE ved navn BASCOM anvendt, som understøtter en lang række microcontrollere i Atmel's AVR-serie. BASCOM giver programmøren adgang til en lang række predefinerede funktioner, så en udvikler typisk ikke behøver at sætte sig ind i microcontrollerens forskellige registre.

Et program skrevet i BASCOM tager typisk udgangspunkt i en række compiler-direktiver. Disse direktiver "fortæller" compileren til hvilken microprocessor, programmet skal kompileres (og på denne måde hvilke pins der er tilgængelige og meget andet), og hvilken clockhastighed, processoren skal konfigureres til at køre med.

```

1 ' Comments are made with an apostrophe
2 $regfile "attiny261.DAT"           ' Defines ATTiny261-specific in/outs
3 $crystal = 8000000                 ' An internal clock of 8MHz

```

Code snippet 6.2: Compiler directives

```

1 Config PORTB = Output              ' Sets a port as OUTPUT
2 Config PORTA = Input               ' Sets a port as INPUT
3 Config Timer1 = Pwm , Prescale = 32 ' Set Timer1: PWM-mode w/ prescale=32
4
5 Dim Aninteger As Int               ' Declaring a variable
6 Dim Aword As Word
7
8 Do
9     Aninteger = 327
10    Aword = 21343
11
12    If Aninteger = 327 And Aword < 400 Then 'If-statement
13        PORTB.0 = 1                    ' Sets pin 0 on PortB to 1
14        PORTB = &B00100001             ' Sets pin 0 and 5 on PortB to 1
15        OCR1A = 128                    ' Sets PWM-output A to 128
16    End If
17
18    If PINA.0 = 1 Then                 ' If pin 0 on PortB is high
19        Do
20            PORTB = &B00100101
21            Waitms 100                  ' Halts the processor for 100ms
22            PORTB = &B00000000
23            Waitms 100
24        Loop Until PINA.0 = 0          ' An example of a do-until loop
25    End If
26 Loop

```

Code snippet 6.3: Simple BASCOM example

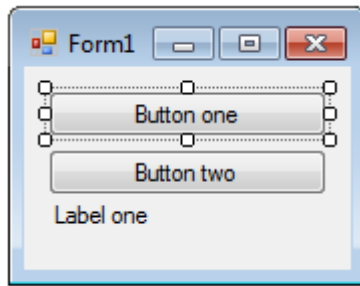
Ovenstående information er hentet fra *BASCOM AVR Help Reference* [6].

### 6.5.3 C#

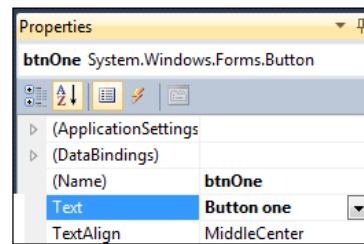
I modsætning til BASIC og C, som er beskrevet i sektion 6.5.2 og 6.5.1, er C#-softwaren som anvendes til dette projekt event-drevent. Dette betyder, at i stedet for at køre et predefineret stykke kode fuldstændigt lineært, så vil programmets udførelse være bestemt af, hvornår brugeren udfører forskellige handlinger - såsom trykker på en knap.

Selve programmeringssproget C# er en videreudvikling af C/C++, men er i modsætning til disse i stor grad objektorienteret. Dette betyder, at store dele af koden er opdelt i såkaldte objekter - dette har dog ikke større indflydelse for udviklingen af C#-software til dette projekt, hvorfor en nærmere gennemgang af dette ikke beskrives.

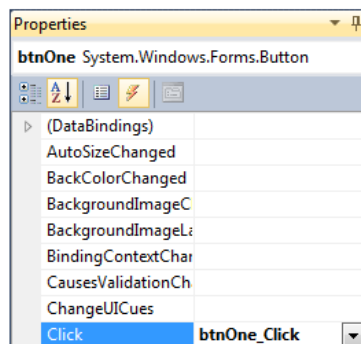
C# er valgt som grundlag for softwaren på PC'en, fordi sproget og IDE'en (kendt som Visual Studio) tillader meget hurtig udvikling af små applikationer med grafiske interfaces. Herunder følger et eksempel på opsætning af en "Windows Form" og behandlingen af et simpelt brugerinput.



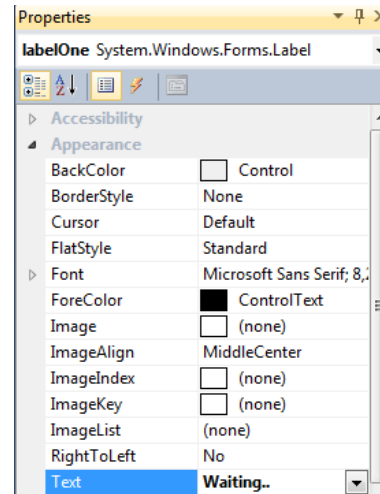
(a) Design af en Windows form



(b) Properties for knap 1



(c) Events bundne til knap 1



(d) Events bundne til label 1

Figur 6.9: Eksempel på opsætning af simpel Windows Forms applikation

På fig. 6.9 ses en opsætning af en meget simpel Windows form - der er her blot tale om et vindue med to knapper og en label. I fig. 6.9(b) ses knap 1's egenskaber, såsom *Name* og *Caption* - her vælges udseende og diverse egenskaber for et objekt. På fig. 6.9(c) ses de events, der er bundne til knappen - her angives navnet på det event, der skal køres ved klik på knappen.

```

1 using System; // Determines other namespaces to include methods from
2 using System.Windows.Forms;
3
4 namespace ExampleProject // The namespace of the current project
5 {
6     public partial class Form1 : Form // Represents the form
7     {
8         public Form1() // Event called when starting the form, autogenerated
9         {
10             InitializeComponent();
11         }
12
13         private void btnOne_Click(object sender, EventArgs e) // The btnOne_Click event
14         {
15             labelOne.Text = "User clicked btn one";
16         }
17     }
18 }

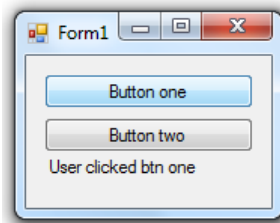
```

```

17
18     private void btnTwo_Click(object sender, EventArgs e) // The btnTwo_Click event
19     {
20         labelOne.Text = "User clicked btn two";
21     }
22 }
23

```

Code snippet 6.4: Changing the properties of a form element with C#



Figur 6.10: Eksempel-programmet efter tryk på knap 1

På fig. 6.10 ses, at label 1's "text-egenskab er blevet ændret til "User clicked btn one" efter tryk på knappen. Efter samme princip kan man således ændre på mange forskellige egenskaber på de mange forskellige elementer, man kan tilføje til sin form. Herunder ses en kort gennemgang af basal C# syntaks.

```

1  int anInt = 254; // Declaring an integer
2  string aString = "This is a string"; // Declaring a string
3
4  if (anInt > 20 && aString.Contains("string")) // A simple if-statement
5  {
6      for (int i = 0; i < 10; i++) // A simple for loop that loops 10 times
7      {
8          anInt += i * 3;
9      }
10 }

```

Code snippet 6.5: Simple C# example

Ovenstående information er hentet fra *Microsoft Visual C# .NET Step by Step* [13].

#### 6.5.4 SQL

SQL - Structured Query Language - er et sprog, der ofte anvendes ved kommunikation med såkaldte SQL-databaser. I dette projekt anvendes kun simple queries (forespørgelser), så det er kun nogle få kommandoer, der er brug for. Der er tale om kommandoer som **SELECT** og **INSERT INTO**.

SQL-databaser bygger på et antal tabeller. Alle disse tabeller indeholder nogle kolonner, og hvert "entry" i tabellen repræsenterer en række. Eksempelvis kunne man forestille sig en database på en skole kunne indeholde en tabel der beskriver elever, som det ses på tabel 6.2.

Ønsker man således at finde et bestemt *felt* i tabellen kan dette gøres ved at filtrere efter nogle bestemte kriterier. Eksempelvis kunne man finde Mads Madsens e-mail vha. SQL-kommandoen:

| Elever       |              |                        |
|--------------|--------------|------------------------|
| Studienummer | Navn         | Email                  |
| 1            | Jens Jensen  | jens.jensen@gmail.com  |
| 2            | Mads Madsen  | mads.madsen@gmail.com  |
| 3            | Kurt Kurtsen | kurt.kurtsen@gmail.com |

Tabel 6.2: Eksempel på opbygningen af en tabel, der viser elever

```
SELECT Email FROM Elever WHERE Navn='Mads Madsen'
```

Naturligvis vil dette give problemer, hvis to elever havde samme navn - af samme grund er kolonnen "Studienummer" også vist. Det er meget almindeligt at tilføje en kolonne med et unikt ID på alle tabeller - så alle entries i tabellen har en såkaldt "Unique Identifier". Dette ID sættes også typisk til at have egenskaben "Primary Key", så databasen automatisk vil sortere og arbejde udfra dette unikke ID. Nedenstående kodeeksempel viser således denne filtrering.

```
SELECT Email FROM Elever WHERE Studienummer=2
```

Ønsker man at indsætte information i tabellen kan dette gøre smed kommandoen `INSERT INTO`. Ønsker man således at indsætte endnu en elev i "Elever-tabellen på tabel 6.2, kunne dette gøres med:

```
INSERT INTO Elever (Studienummer, Name, Email) VALUES
(4, 'Rasmus Rasmussen', 'rasmus.rasmussen@gmail.com')
```

Der refereres desuden til sektion 8.1 for en beskrivelse af PC-softwaren udviklet til dette projekt. Her findes også en beskrivelse af SQL-serveren, tabeller deri, og kommandoer, som bruges til at finde data deri.

Ovenstående information er hentet fra *SQL Fundamentals* [2].

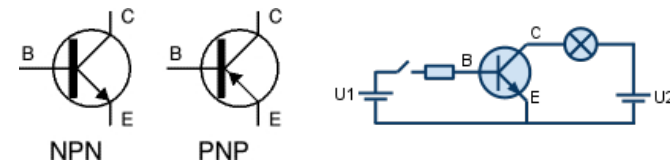
Dette afslutter beskrivelsen af de forskellige programmeringssprog, som er anvendt til projektet. En detaljeret beskrivelse med den egentlige kode og udviklingen af samme kan ses under sektion 8 (s. 47).

## 6.6 Transistorer

### 6.6.1 Bipolar transistor (BJT-transistorer)

BJT-transistorer (Bipolar Junction Transistor) har tre indgange som er navngivet base (B), collector (C) og emitter (E). Der findes to varianter: NPN og PNP (se figur 6.11(a) (s. 38)).

Transistorerne benyttes i såkaldt switchmode i dette projekt. Dette foregår ved at en strøm går gennem Base-Emitter forbindelsen i transistoren og derved bliver transistorens indre elektrisk ledende og lader en strøm gå gennem Collector-Emitter forbindelsen. Transistorens indre er kun ledende så længe der går en strøm gennem Base-Emitter forbindelsen. Hvis der slukkes for denne forbindelse kan der ikke længere gå en strøm gennem Collector-Emitter forbindelsen.

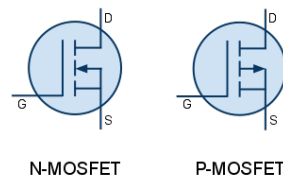


(a) Symbol for NPN og (b) Tegning over hvordan en BJN transistor i switchmode fungerer

Figur 6.11: Forskellige transistortyper

Hvis kontakten på figur 6.11(b) (s. 38) aktiveres vil der gå en strøm fra U1 gennem B-E forbindelsen. Dette tillader at en strøm fra U2 kan gå gennem C-E forbindelsen og få dioden til at tænde.

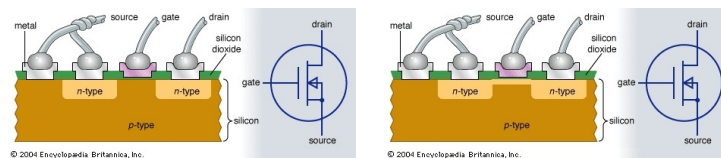
## 6.6.2 MOSFET transistor



Figur 6.12: Symboler for P-MOSFET og N-MOSFET transistorer [15]

En MOSFET transistor har også tre ben men de er navngivet drain (D), gate (G) og source(S) (se figur 6.12). Ligesom der med bipolare transistorer findes NPN og PNP transistorer finder der N-channel MOSFET og P-channel MOSFET. I dette projekt benyttes der P-channel MOSFET. MOSFET er et akronym for Metal-Oxide-Semiconductor Field-Effect Transistor.

## Opbygning af MOSFET transistor



(a) Opbygning af N-MOSFET transistor som er "slukket" [15] (b) Opbygning af MOSFET transistor som er "tændt" [15]

Figur 6.13: Mosfets som tændt eller slukket

På figur 6.13(a) (s. 38) kan man se opbygningen af en MOSFET transistor, hvor der ikke er en spænding på gate benet. Dette gør at der ikke er forbindelse mellem source benet og drain benet. Hvis der tilføres en positiv spænding på gate benet i forhold til source benet skaber det positive ladninger på gate benets tilslutning. De positive ladninger vil tiltrække negative ladninger i silicium-laget. Dette danner en kanal mellem source benet og drain benet hvilket tillader forbindelse mellem de to ben. Se

figur 6.13(b) (s. 38). Kanalen mellem source benet og drain benet kan også betragtes som en modstand. Når MOSFET'en er slukket er modstanden mellem drain og source i mega ohm området men når MOSFET'en er helt åbnet op er modstanden i mili ohm området. Der bruges meget lidt strøm på at åbne en MOSFET transistor i forhold til en bipolar transistor og der bruges kun strøm i den korte tid det tager at åbne kanalen. Da det kun kræver lidt strøm at tænde MOSFET-transistorene er det en god ide at have en modstand mellem gate og source benet. Denne modstand sørger for at aflade gate benet, hvis forbindelsen til microcontrolleren afbrydes. Samtidigt fungerer modstanden som en pull-down modstand, som forhindrer at støjkluder kan tænde MOSFET-transistoren.

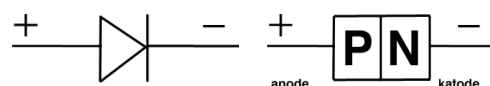
Forskellen på en P-MOSFET og en N-MOSFET er at p-materialet og n-materialet har byttet plads. Dette gør at en P-MOSFET skal aktiveres omvendt i forhold til en N-MOSFET så der skal være en negativ spænding på gate benet i forhold til source benet.

Alle overstående information kommer fra *Electronic Components Reference* [12], *Hvad er FET'en for en fætter* [15] og *Analog Teknik* [22].

## 6.7 LED

En diode er opbygget af en anode og en katode. Dioden har en lederetning og en spærreretning og tillader kun strømmen at gå en vej. Denne retning er fra anoden mod katoden. Konventionelle dioder bruges til mange forskellige formål indenfor elektronik som for eksempel som ensrettere.

Dioder bliver fremstillet af silicium eller germanium som begge er halvledere fra 4. hovedgruppe. Man blander stoffet i anodesiden med et stof med et underskud af elektroner for at skabe positivt potentiale, derfor det kaldes P-delen. Ligeledes blandes katodesiden til et negativt potentiale. På denne vis udnyttes elektronerne egenskaber i dioder og transistorer.



(a) Illustration af (b) P delen i anoden og strømretning i dioden N delen i katoden

Figur 6.14: Diodesymboler

LED (Light Emitting Diode) betyder lys emitterende diode. En LED er ikke opbygget af samme halvledere som konventionelle dioder. Lysdioden lyser ved brug af halvledere, som ved spændingstilslutning udsender lys i stedet for varme. LED omsætter strømmen direkte til lys ved strømgennemløb i lederetningen. Dette lys forstærkes yderligere ved at bygge lysdiodens hus af gennemsigtige materialer med bestemte optiske egenskaber. Kort beskrevet udnytter man elektronerne fotonenergi. Væsentlige fordele ved LED belysning:

- Den tænder og slukker hurtigere end en glødetråd
- Den har et lavt effektforbrug i forhold til lyseffekten

Oftentimes bruger man LED som indikator dioder. En lysdiode bruges altid med en formodstand og man

tilslutter typisk mellem 2 og 5 volt til diodens anode. Spændingsfaldet over dioden vil oftest være omkring 2V. Imidlertid er et anode/katode par kun i stand til at vise en enkelt farve. Derfor har man udviklet RGB dioder med minimum 3 anoder til at fremstille henholdsvis rødt, grønt og blå lys. RGB dioden anvendes i dette projekt til oplysning af de enkelte felter.

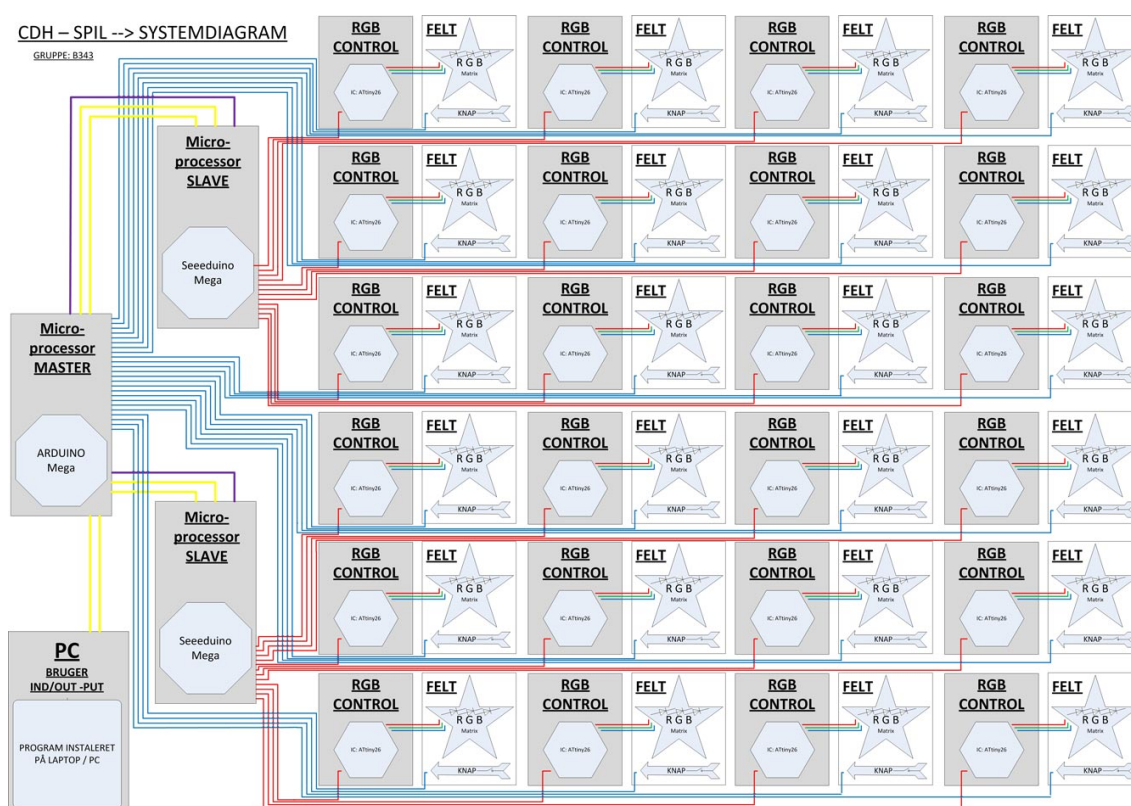
Der benyttes en RGBB LED med 4 anoder og 2 katoder. Huset er en diffused model for at undgå, at lyskeglen skal blænde når man anvender prototypen. Grunden til anvendelsen af RGB er mulighederne for farveblandinger vha. de 3 grundfarver rød, grøn og blå. Dette gøres ved hjælp af et PWM-signal fra displaydriveren, som det fremgår af *Design af Hardware*, sektion 7.3 (s. 44) og *Design af Software*, sektion 8.3 (s. 64).

Baggrundsinformationen til dette afsnit er hentet fra *Analog Teknik* [22].

# Design af Hardware

I dette kapitel gennemgås hardwaren i det samlede system. De enkelte delelementer vil blive gennemgået. Vi vil behandle design, beregninger og konstruktion.

## 7.1 Systemopbygning



Figur 7.1: Detaljeret systemdiagram

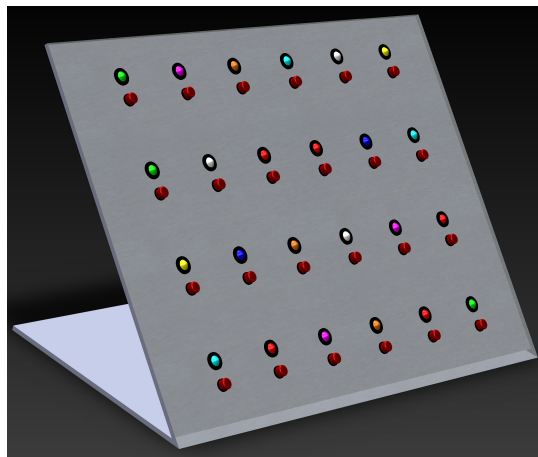
Oversigten på figur 7.1 (s. 41) viser systemet og forbindelserne mellem de forskellige delelementer. Man kan også finde figur 7.1 (s. 41) i appendiks 11.1 De enkle delelementer uddybes i de følgende Afsnit. Overordnet består systemet af fem dele:

1. Felterne bestående af Display (RGB-Diode) og Input (Knap)
2. RGB-Control bestående af en programmerbar kreds (ATTiny26) og tilhørende kreds til styring af RGB-diode
3. Microprocessorer slaver. Disse styrer hver 12 RGB-Controllere.

4. Microprocessorer master. Grundstenen i systemet som styre de to slave microprocessorer og samtidig har kommunikation med en PC.
5. En tilkøbet PC som rummer brugerinterface og databehandling.

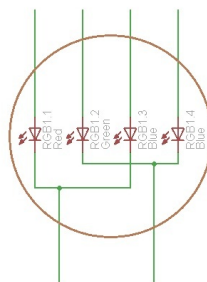
## 7.2 Felterne - in/out-put til eleverne

Som beskrevet i afsnit 7.1 består felterne i vores prototype af en enkelt RGB diode og en trykknop. Da vores prototype afspejler det færdige produkt med 24 felter har vi valgt at repræsentere disse felter med 24 dioder. Hvert felt har en diode og en tilhørende trykknop monteret i en 2mm aluminiumsplade hvilket er illustreret på figur 7.2



Figur 7.2: 3D tegning af vores testopstilling

Hvis man ser på et diagram over dioden er de forskellige underdioder som findes inde i 1 RGB-diode er forbundet som på figur 7.3



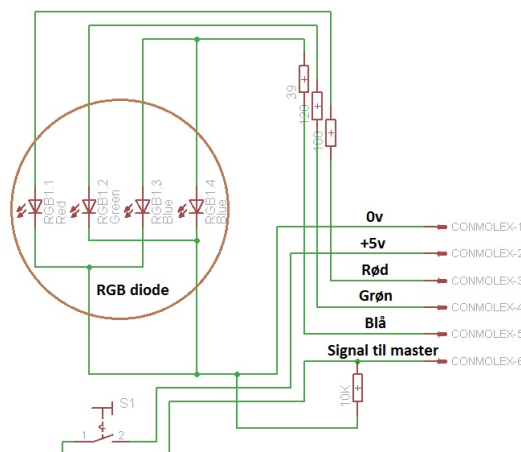
Figur 7.3: Simpelt diagram over RGBdiode

Dette betyder at der fra hvert "felt" eller diode i testopstilling skal føres i alt 6 forbindelser ud som består af:

- 1. Fælles stel forbindelse til alle "under-dioderne"

- 2. Rød forbindelse som skal forbindes med +5v for at forsyne den røde diode med strøm
- 3. Grøn forbindelse som skal forsyne den grønne diode
- 4. Blå forbindelse til at forsyne den blå diode
- 5. Knap forbindelse som skal gå til det respektive ben på microprocessoren
- 6. +5v som skal kobles til det andet ben på knappen således der skabes forbindelse mellem +5v og microprocesoren når knappen aktiveres.

Disse forbindelser føres i testopstillingen ud til et stik hvilket forenkler ledningsforbindelserne. Dette kan ses på figur 7.4



Figur 7.4: Diagram af et enkelt felts forbindelser

### 7.2.1 Beregninger af formodstande

For at kontrollere den strøm og spænding som vores RGB dioder forsynes med har vi placeret en modstand mellem dioderne og +5v som det kan ses på figur 7.4

Størrelserne af disse modstande er bestemt vha. ohms lov ud fra de data man kan finde i databladet om hvilket spænding og strøm dioderne skal forsynes med. Dette kan ses på figur 7.5 og 7.6. Yderligere oplysninger om dioden kan findes i databladet som ligger som bilag 12.6

| Parameter          | High Efficiency Red | Green | Blue | Units |
|--------------------|---------------------|-------|------|-------|
| Power dissipation  | 105                 | 105   | 105  | mW    |
| DC Forward Current | 30                  | 25    | 30   | mA    |

Figur 7.5: Udsnit af RGB dioders datablad

| $V_F$ | Forward Voltage | High Efficiency Red | Green | Blue | V | $I_F=20mA$ |
|-------|-----------------|---------------------|-------|------|---|------------|
|       |                 | 2.0                 | 2.5   |      |   |            |
|       |                 | 2.2                 | 2.5   |      |   |            |
|       |                 | 4.0                 | 4.5   |      |   |            |

Figur 7.6: Udsnit af RGB dioders datablad

Når disse data er indhentet kan man beregne størrelserne på modstande på følgende måde

$$R = U/I$$

↓

$$R = (5v - U_{diode})/I_{diode}$$

↓

$$R_{Red} = (5v - 2v)/0,03A = 100\Omega$$

$$R_{Green} = (5v - 2.2v)/0,025A = 112\Omega \approx 120\Omega$$

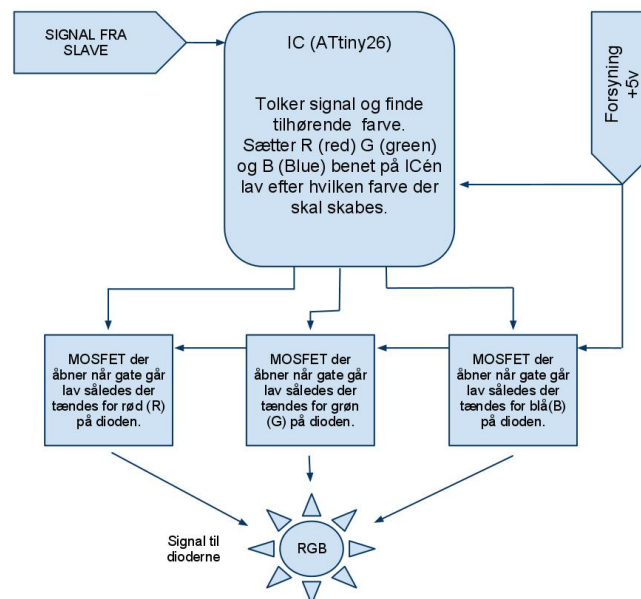
$$R_{Blue} = (5v - 4v)/0,03A = 33\Omega \approx 39\Omega$$

Bemærk at afrundingerne er til de nærmeste modstande til rådighed.

## 7.3 RGB Controller

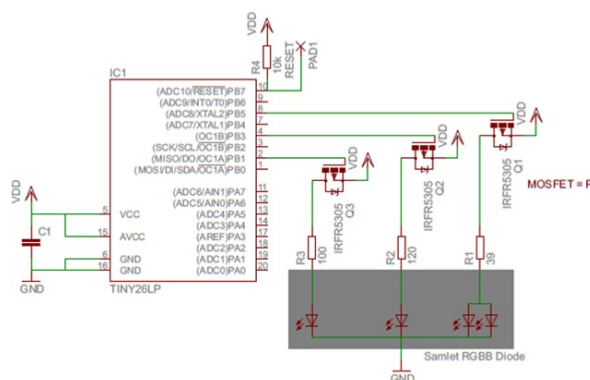
Det har været nødvendigt at indføre en RGB-controller i systemet. Grunden til dette er at der ellers skulle bruges et meget stort antal microcontrollere for at have nok udgange til hver enkel RGB diode. RGB-controllerne styres via et PWM signal fra de to slave microprocessorer som beskrives i sektion 7.4 (s. 46). RGB-controllerne tolker signalet fra slaverne og med den kode, der ligger på de dertil anvendte ATtiny'er, "blander" den farverne i RGB-dioderne. I praksis pulses/blinkes der med de tre grundfarver således de ønskede farver fremkommer. Metoden til dette og koden der anvendes forklares i sektion 8.3 (s. 64)

Et diagram over RGB-controllerens funktion vises på figur 7.7 (s. 44) Opbygningen af RGB-controllerne



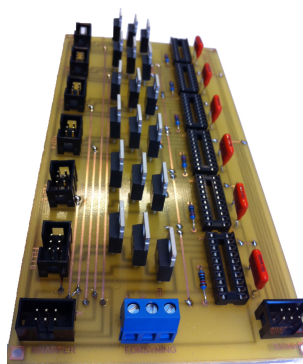
Figur 7.7: Diagram over funktionen af RGB-controllerne

på diagramform er vist på figur 7.8 (s. 45) Opbygningen vist på figur 7.8 (s. 45) er systemet for en "kanal" det vil sige for en RGB-diode. I praksis er RGB-controlleren opbygget på modul basis således at den har seks "kanaler" på hvert print. Formodstandene til RGB-dioderne som er vist på diagrammet er ikke en del af disse moduler da de sidder som en del af felterne forklaret under sektion 7.2 (s. 42).



Figur 7.8: Diagram over ideen med RGB-controlleren

Diagram og printudlæg over et fuldt modul med seks kanaler kan findes som Appendiks D 11.4. På figur 7.9 (s. 45) kan man se et færdigt modul.



Figur 7.9: fotografi af displaydriver

### 7.3.1 Komponentvalg

RGB-controllerne indeholder to vigtige elementer:

- ATTinys Microprocessor
- P-channel MOSFETS

ATTinyerne er sat op med en 100nF kondensator over forsyningen til og eliminerer eventuelt støj og dermed sikre en jævn forsyning som er nødvendigt for en pålidelig afvikling af koden. Teorien bag anvendelsen af kondensatorer som støjfilter og anvendelsen her er ikke gennemgået nærmere i dette projekt. ATTinyerne er opbygget efter samme princip som microprocessoren beskrevet i sektion 6.1 (s. 26) de indeholder dog ikke helt så mange features. Databladet over dem kan findes som bilag 12.4 Der benyttes MOSFETs i switchmode til at drive vores RGB-dioder. Opstillingen af MOSFETs i switchmode kan ses på figur 7.8 (s. 45) og der kan læses om teorien bag MOSFETS i 6.6. Der er valgt P-Chanel MOSFETs da der skal åbnes for strømmen ved at trække gate-benet "G" på MOSFETen lav. Når G er lav så åbnes der fra source "S" til drain "D". Det er vigtigt man får en stor nok potential difference mellem G og S til at der åbnes helt for MOSFETen. Hvis man ikke får åbnet helt vil man have

en meget stor modstand i MOSFETen. Dette er beskrevet i MOSFETens datablad der findes som bilag 12.5.

## 7.4 Microprocessorer

Der er en microprocessor i master microcontrolleren, i de to slaver microcontrollerer og i ATtiny'erne. Masteren og de to slaves har ATmega1280 microprocessorer imens ATtiny'erne har en AVR cpu. Masteren er forbundet med de to slaver med seriel kommunikation som benytter sig af RS232-kommunikation-protokol. Slaverne sender derefter et PWM signal videre til ATtiny'erne. ATtiny'erne har tre ben som skal styre transistorerne som bruges til at styre de tre farver på felterne. Der kommer en forbindelse fra knapperne i felterne og ind til masteren. Masteren har også en forbindelse over til en PC som forklares nærmere i sektion 7.5 (s. 46).

## 7.5 PC tilkobling

Arduino Mega microcontrolleren som fungere som master i prototypen koples til en PC via USB. Computeren kan ved hjælp af Arduino software overføre data til microcontrolleren, samt at den kan modtage signaler fra den. Arduino Mega indeholder fire UARTs controller. En UARTs er ofte en selvstændig del, men kan også være integreret i en ICer (integreret kredsløb). UARTen sørger for at microprocessoren kan læse de data som sendes og modtages fra en computer eller fra en anden microprocessor. Ud over at sende og modtage signaler, fungere USB tilslutningen også som strømforsyning til microcontrollerne. De to slave microcontroller benytter ligeledes computeren som strømforsyning vha. en USB tilslutning. UARTen er uddybet i sektion 6.3 (s. 28)

# Design af Software

## 8.1 PC-software

Pc-softwaren er lærerens redskab. Læreren kan vha. GUI'et (Graphical User Interface) på PC'en vælge spil og indsamle data, som beskriver elevernes præstationer. Dette kapitel beskriver udviklingen af denne software.

Som det ses af kravene i sektion 3.2 (s. 9) og sektion 3.5 (s. 11), er der på forhånd opstillet en række krav til softwaren. Softwaren på PC'en skal:

- Have et anvendeligt interface
- Være i stand til at sende kommandoer, som signalbehandlingen er i stand til at behandle
- Være i stand til at modtage information fra signalbehandlingen, og lagre det
- Være i stand til at præsentere information på en graf

På baggrund af ovenstående krav udvikles softwaren til PC'en. Både dataopsamling, databehandling og applikationskontrollen samles i et program. Programmet udvikles i C#, som beskrevet i sektion 4.1 (s. 13) og sektion 4.4 (s. 15). Det forudsættes, at læseren har kendskab til den grundlæggende C# og SQL-syntaks samt basal programmering. En kort introduktion til disse emner kan desuden findes i sektion 6.5 (s. 31).

Bemærk, at *score* i resten af dette kapitel refererer til en værdi, der afspejler en elevs præstation i et spil. Denne score kan repræsenteres af eksempelvis en tidsenhed, en procentdel rigtige svar - dette er afhængigt af Arduino-softwaren. Læser bør være opmærksom på, at kommentarene i kodeudsnittene er væsentlige for forståelsen af kapitlet, mens læser kan vælge at se bort fra selve koden.

### 8.1.1 Design af SQL-serveren

Før udviklingen af softwaren, er det naturligvis nødvendigt at lave en grundig planlægning og et detaljeret design af systemets database og programflow. Siden det er et meget basalt krav, at en lærer skal være i stand til at se en elevs evner og eventuel udvikling over tid, er det naturligvis vigtigt, at systemet skal have kontrol over et lager, hvor denne data kan gemmes. Som følge af ideudviklingen i sektion 4.1 (s. 13) og sektion 4.4 (s. 15) anvendes en SQL-server til at gemme disse data. Følgende data er således interessant at gemme:

- Et elev-id for at have nem tilgang til de forskellige elever
- Elevens navn
- Hvilket spil, eleven har spillet - når systemet skal være i stand til at styre flere spil, er det naturligvis vigtigt at vide hvilket spil, der er tale om

- Hvilken score, eleven har fået - der er her tale om en pointenhed, der afspejler elevens niveau i de forskellige spil. I nogle spil kan dette være en tidsenhed, men kan også eksempelvis være en procentdel rigtige gæt el.lign.

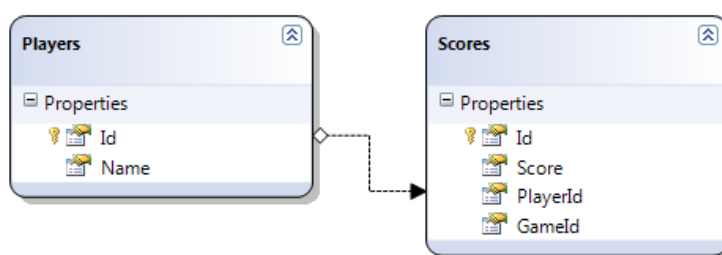
En SQL-database sættes op, og informationen deles ud over to tabeller. En tabel vil indeholde elever, og en vil indeholde scores - dette vil gøre arbejdet med og behandlingen af dataen lettere og mere overskuelig.

| PLAYERS |             |             |                     |                                                          |
|---------|-------------|-------------|---------------------|----------------------------------------------------------|
| Navn    | Datatype    | Attributter |                     | Beskrivelse                                              |
| Id      | Integer     | Primary     | Key, Auto-increment | En "Unique identifier", som identificerer eleven         |
| Name    | Varchar(25) |             |                     | Elevens navn, så en lærer også kan identificere eleverne |

Tabel 8.1: Tabellen Players

| SCORES   |          |             |                     |                                                                       |
|----------|----------|-------------|---------------------|-----------------------------------------------------------------------|
| Navn     | Datatype | Attributter |                     | Beskrivelse                                                           |
| Id       | Integer  | Primary     | Key, Auto-increment | En "Unique identifier", som identificerer scoren                      |
| Score    | Integer  |             |                     | Selve scoren - tidsenhed, procentdel, ...                             |
| PlayerId | Integer  |             |                     | Elevens (spillerens) Id - relateret til tabellen Players, kolonnen Id |
| GameId   | Integer  |             |                     | Spillets Id - identificerer spillet i applikationsstyrings-softwaren  |

Tabel 8.2: Tabellen Scores



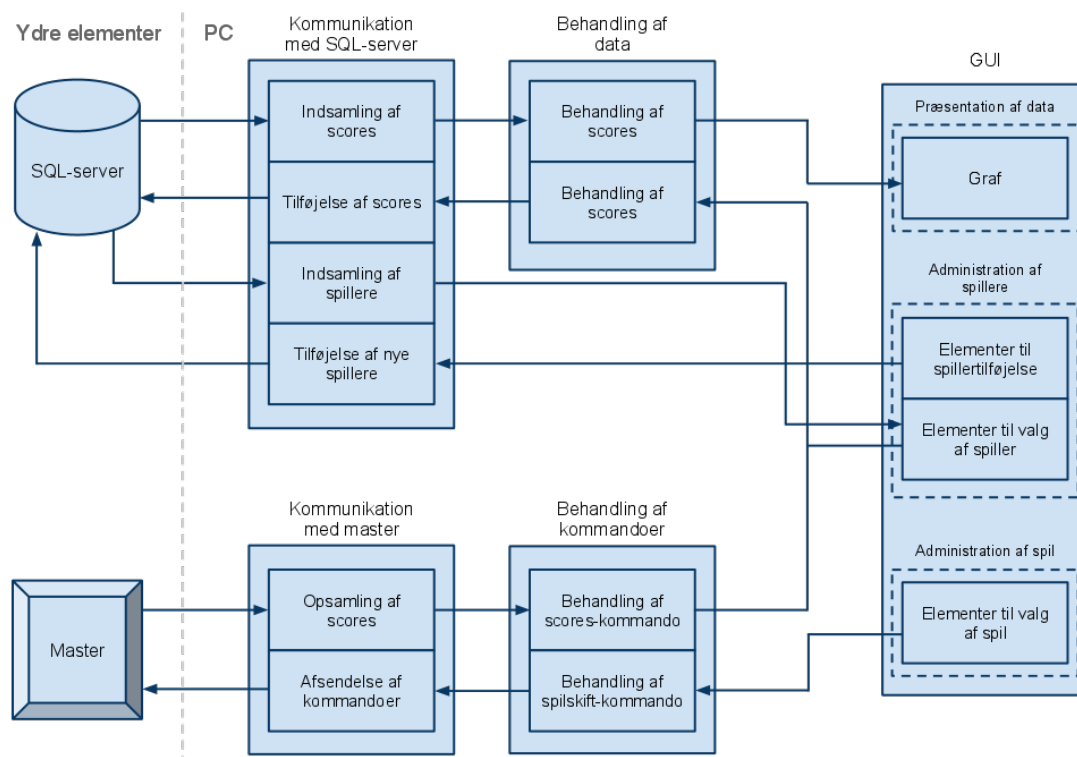
Figur 8.1: Klassesdiagram, som viser relationen mellem players og scores

På tabel 8.1 (s. 48) og tabel 8.2 (s. 48) samt figur 8.1 (s. 48) ses således en oversigt over de to tabeller, der er i databasen samt den indbyrdes relation mellem Players.Id og Scores.PlayerId.

### 8.1.2 Design af PC-softwaren

Siden PC-softwaren skal være i stand til at varetage både dataopsamlingen, databehandlingen og applikationskontrollen, skal softwaren naturligvis nøje designes derefter. På figur 8.2 (s. 49) ses et blokdiagram af softwarens forskellige funktioner.

Hvert afsnit i dette kapitel vil være opbygget med udgangspunkt i dette blokdiagram - en overskrift **Markeret med fed** vil svare til overskriften på en blok i blokdiagrammet. Under hver af disse overskrifter vil de forskellige funktioner, som blokken skal varetage være beskrevet. De indbyrdes relationer mellem blokkene og afsnittene kan således ses på figur 8.2 (s. 49).



Figur 8.2: Illustration af PC-softwarens kommunikation med de ydre elementer

### GUI

PC-softwarens GUI (Graphical User Interface) designes - som det ses på figur 8.2 (s. 49) skal GUI'et mindst have følgende elementer:

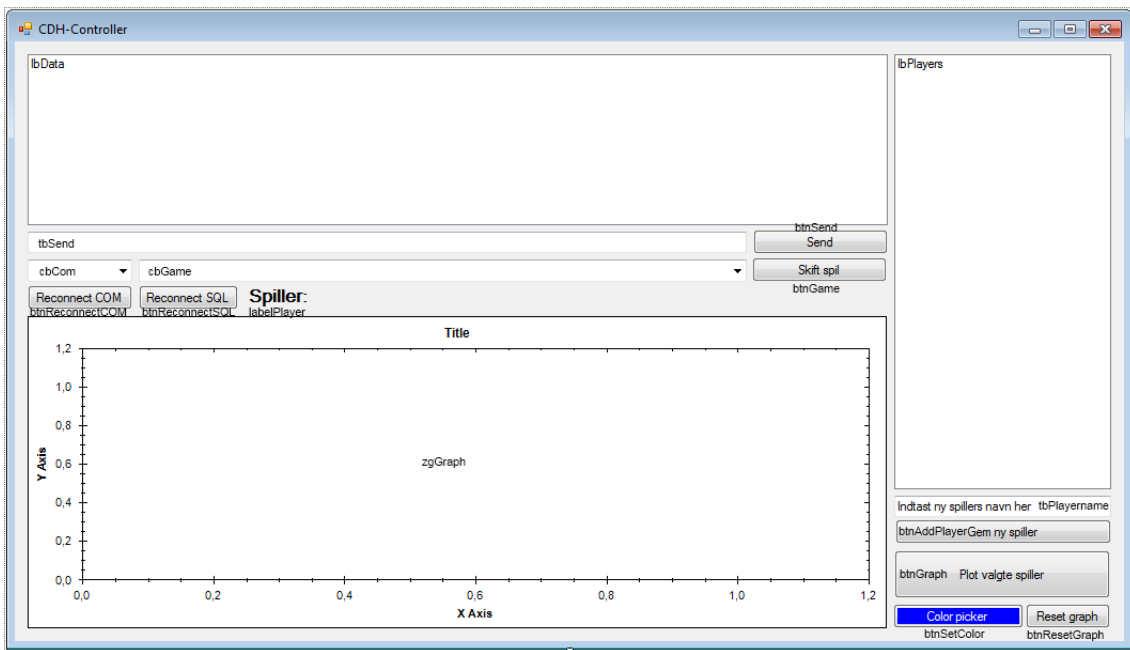
- Graf - her anvendes et eksternt library ved navn ZedGraph
- Elementer til spillertilføjelse - her anvendes en knap og en tekstboks
- Elementer til valg af spiller - her anvendes en såkaldt listboks
- Elementer til valg af spil - her anvendes en combobox og en knap
- Elementer til test og debugging - her anvendes en listboks, tekstboks og knap

Disse elementer placeres på en såkaldt windows form - en beskrivelse af alle de forskellige elementer ses på tabel 8.3 (s. 50), og placeringen af de forskellige elementer ses på figur 8.3 (s. 51).

I resten af kapitlet vil *globale* variabler betyde, at variablerne er globale i Form1 og således for hele programmet. Udtrykket *bruger* og *user* vil i resten af kapitlet dække over den lærer, som styrer og anvender PC'en - *player* betegner eleven, som spiller spillet.

| GUI                             |                 |                                                                                                              |
|---------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------|
| Elementnavn                     | Elementtype     | Beskrivelse                                                                                                  |
| Graf                            |                 |                                                                                                              |
| zgGraph                         | ZedGraphControl | Grafvinduet, hvori spilleres scores kan ses                                                                  |
| btnGraph                        | Button          | Knap, som trykkes på, når brugeren ønsker at plotte en spillers scores                                       |
| btnSetColor                     | Button          | Knap, som bringer farvedialogen "ColorDialog" op, så brugeren kan vælge farven på den næste graf, der tegnes |
| btnResetGraph                   | Button          | Knap, som bruges til at fjerne alle grafer fra grafvinduet                                                   |
| Elementer til spillertilføjelse |                 |                                                                                                              |
| tbPlayername                    | Textbox         | Tekstboks, hvori brugeren indtaster navnet på en ny spiller                                                  |
| btnAddPlayer                    | Button          | Knap som trykkes på, når brugeren ønsker at tilføje en ny spiller                                            |
| Elementer til valg af spiller   |                 |                                                                                                              |
| lbPlayers                       | Listbox         | Listboks, hvori brugeren kan se en liste over spillere og vælge den nuværende spiller                        |
| Elementer til valg af spil      |                 |                                                                                                              |
| cbGame                          | Combobox        | Dropdown-boks, hvor brugeren kan vælge spil                                                                  |
| btnGame                         | Knap            | Knap, som ved tryk sørger for at skifte spil                                                                 |
| Elementer til test og debugging |                 |                                                                                                              |
| lbData                          | Listbox         | Listboks, hvori debugging information vises                                                                  |
| tbSend                          | Textbox         | Tekstboks, hvori "rå"kommandoer kan skrives til masteren                                                     |
| btnSend                         | Button          | Knap, der ved knap sender den "rå"kommando til masteren                                                      |

Tabel 8.3: Elementer i GUI'et



Figur 8.3: PC-softwarens GUI med angivelse af elementnavne

## Kommunikation med SQL-server

Som det ses på figur 8.2 (s. 49), skal denne del af koden administrere indsamling og tilføjelse af scores og spillere. Først skal der oprettes forbindelse til SQL-serveren - dette er en MySQL-server. En MySQL-server er let at tilgå via en hjemmeside, hvilket åbner mulighed for videreudvikling. For at få adgang til denne anvendes en klasse ved navn `MySQL.Data.MySqlClient.MySqlConnection`. `OpenSql()` opretter forbindelse til databasen, og hvis forbindelsen oprettes, afspejles dette af en besked i loggen `lbData` - opstår der en fejl, vil dette fremgå af loggen.

Når denne funktion således er blevet kaldt, kan resten af programmet gøre brug af den globale instans af klassen `MySQLConnection` v. navn `conDatabase`. Som beskrevet i figur 8.2 (s. 49), skal systemet desuden være i stand til at gemme scores i databasen - dette gøres ved hjælp af funktionen `SaveScore()`, som er beskrevet i kodeudsnit 8.1 (s. 51).

```

1  /// <summary>
2  /// Saves the score of a person playing a game
3  /// NOTE: Uses the global variable "game"
4  /// </summary>
5  /// <param name="score">The score</param>
6  /// <param name="name">The name of the player</param>
7  private void SaveScore(int score, string name)
8  {
9      // If the connection is opened...
10     if (conDatabase.State == System.Data.ConnectionState.Open)
11     {
12         // Assemble the SQL-command
13         // string.format combines a string with a number of substrings
14         MySqlCommand cmd = new MySqlCommand(

```

```

15         string.Format("INSERT INTO scores (Score, PlayerId, GameId) VALUES ({0}, {1}, {2})",
16             score,
17             getPlayerId(lbPlayers.SelectedItem.ToString()), // The Id of the player is fetched
18             game),
19         conDatabase);
20     // Execute the command, thereby inserting the score in the database
21     cmd.ExecuteNonQuery();
22 }
23 }

```

Code snippet 8.1: Saving a score to the database

På linje 17 i kodeudsnit 8.1 (s. 51) ses desuden et kald til funktionen `getPlayerId()`. Denne funktion er blot i stand til at finde en spillers Id ud fra spillerens navn - dette forudsætter, at der ikke er to spillere med samme navn.

Et kald til `SaveScore()` vil således indsætte en score i databasen - `Score`, `PlayerId` og `GameId` angives, og er uddybet nærmere i figur 8.1 (s. 48). Id specificeres *ikke*, da Id (som beskrevet i tabel kodeudsnit 8.2 (s. 48)) har egenskaben *Auto-increment*, så værdien af Id automatisk forhøjes med en for hver række, der indsættes.

Iflg. figur 8.2 (s. 49) mangler nu *Indsamling af spillere* og *Tilføjelse af nye spillere*. Spillere indsamles (hentes fra databasen) ved hjælp af funktionen `RefreshPlayers()`, som ses på kodeudsnit 8.2 (s. 52).

```

1  /// <summary>
2  /// Fetches players from the SQL-server and adds them to lbPlayers
3  /// </summary>
4  private void RefreshPlayers()
5  {
6      try
7      {
8          // Empty lbPlayers, first
9          lbPlayers.Items.Clear();
10         // Initialize and set the command
11         MySqlCommand cmd = conDatabase.CreateCommand();
12         cmd.CommandText = "SELECT Name FROM players ORDER BY Id";
13         // Start the DataReader
14         MySqlDataReader rdr = cmd.ExecuteReader();
15         while (rdr.Read()) // While there's more data to read
16         {
17             // Add the string in column 0 (the player name) of the current row to lbPlayers
18             lbPlayers.Items.Add(rdr.GetString(0));
19         }
20         // Close the reader
21         rdr.Close();
22     }
23     catch { } // If something goes wrong nothing currently happens. Just ignore the error.
24 }

```

Code snippet 8.2: Udfyldning af spillerlisten

Denne funktion vil således fylde listboksen `lbPlayers` med de forskellige spillere, der er gemt i databasen. Ønsker man at tilføje nye spillere gøres dette ved at udfylde tekstboksen `tbPlayername` med navnet og derefter et klik på knappen `btnAddPlayer`. Alle disse controls' placering på GUI'et kan ses

på figur 8.3 (s. 51).

Ved tryk på `btnAddPlayer` tilføjes den nye spiller til databasen, og spillerlisten `lbPlayers` opdateres. Selve kommunikationen med SQL-serveren er således færdiggjort. Før scores kan vises i grafvinduet (`zgGraph`) kræver det dog stadig en behandling af dataen fra databasen.

## Behandling af data

Eftersom det ønskes at illustrere information fra databasen når brugeren klikker på knappen `btnGraph`, sættes behandlingen af data og overførslen til grafvinduet `zgGraph` i `btnGraph`'s click-event. I kodeudsnit 8.3 (s. 53) ses koden med forklarende kommentarer.

```
1  /// <summary>
2  /// Collects scores from the database related to the game w/ ID cbGame.SelectedIndex+1 and the player
   /// selected in lbPlayers.
3  /// </summary>
4  private void btnGraph_Click(object sender, EventArgs e)
5  {
6      // Create a new instance of the PointParList class
       PointParList list = new PointParList();
7
8
9      // Create the MySqlCommand and start the DataReader
       MySqlCommand cmd = conDatabase.CreateCommand();
10
11     cmd.CommandText = string.Format("SELECT Score FROM scores WHERE PlayerId={0} AND
       GameId={1} ORDER BY Id", getPlayerId(lbPlayers.SelectedItem.ToString()), cbGame.
       SelectedIndex+1);
12     MySqlDataReader rdr = cmd.ExecuteReader();
13
14     // While there's still new data to read, add the data to the PointParList
       int i = 0;
15     while (rdr.Read())
16     {
17         list.Add(i, rdr.GetInt32(0));
18         i++;
19     }
20     rdr.Close();
21
22
23     // Finally, put the data in the list on the graph
       // Note that the symbol is a Diamond for all datasets
24     AddToGraph(zgGraph, list, string.Format("{0}'s scores", lbPlayers.SelectedItem.ToString()), colorDialog
       .Color, SymbolType.Diamond);
25
26 }
```

Code snippet 8.3: Visning af en spillers scores på en graf

Som det ses på linje 25 i kodeudsnit 8.3 (s. 53), laves et kald til funktionen `AddToGraph`. Denne funktion står, som navnet antyder, for at tilføje punkterne til grafen.

Ved tryk på knappen `btnGraph` vil der således i grafvinduet `zgGraph` vises en graf over en elevs scores. Ønsker brugeren at zoome, se værdier, gemme grafen som billede el.lign. kan dette gøres ved at højreklikke på grafvinduet. Alle disse funktioner er del af `ZedGraph`, hvorfor denne kode ikke beskrives.

## Kommunikation med master

En vigtig del af softwaren er kommunikationen med masteren - se figur 8.2 (s. 49). Dette foregår ved hjælp af seriel kommunikation og RS232-kommunikations-protokollen, se sektion 6.3 (s. 28). Ved hjælp af namespace `System.IO.Ports` kan klassen `SerialPort` tilgås. En ny klasse ved navn `SerialPortController` udvikles.

Denne klasse vil stå for afsendelse af kommandoer til masteren, og laves en instans af klassen, vil man ved hjælp af nogle simple funktionskald være i stand til at etablere forbindelse til computerens COM-port, samt sende data.

Når forbindelsen til COM-porten skal åbnes, kaldes funktionen `OpenCom()`, som således blot styrer en instans af klassen `SerialPortController`. Denne funktion opretter forbindelse til COM-porten specificeret af den globale variabel `m_portStr`, som indeholder navnet på porten, der anvendes. Desuden tilføjes tekst til loggen `lbData`, så brugeren af programmet ved, at forbindelsen er blevet oprettet - opstår der således en fejl, vil brugeren få besked derom. Det kan nu lade sig gøre at sende kommandoer til masteren ved hjælp af metoden `SerialPortController.SendString()`.

Et event ved navn `port.DataReceived` står for modtagelse af beskeder fra masteren. Dette event ses på kodeudsnit 8.4 (s. 54) med forklarende kommentarer.

```
1  /// <summary>
2  /// Triggered when data is received on the COM-port
3  /// </summary>
4  private void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
5  {
6      // Collect the message waiting in the input buffer
7      string inp = m_comPort.port.ReadExisting();
8
9      // The message is appended to the global stringbuilder str
10     str.Append(inp);
11
12     // If there's a new line character, the message is complete, and we continue processing
13     // if not, the semi-completed message will wait until the next part of the message is received
14     if (inp.Contains("\n"))
15     {
16         // The message is added to the log
17         addToLb("Received: " + str.ToString());
18
19         // If the message contains "SCORE:", its a score-entry, that should be added to the database
20         if (str.ToString().Contains("SCORE:"))
21         {
22             // Catch the actual score (int), and cut the "SCORE:"-part off
23             currScore = Convert.ToInt32(str.ToString().Split(':')[1]);
24
25             // Save the score to the database
26             SaveScore(currScore, lbPlayers.SelectedItem.ToString());
27         }
28         // If the message contains "STATUS", its a STATUS-request from the master
29         else if (str.ToString().Contains("STATUS"))
30         {
```

```

31 |         // This message means the master is trying to verify the connection – we reply with "OK"
32 |         m_comPort.SendString("OK\n");
33 |
34 |         // And lets add the data to the log
35 |         addToLb("Sent: OK");
36 |         addToLb("–Connection to Master verified–");
37 |     }
38 |
39 |     // Since the message is over and processed, clear the stringBuilder
40 |     str.Clear();
41 | }
42 | }

```

Code snippet 8.4: Modtagelse af data fra COM-porten

Brugeren skal desuden være i stand til at sende kommandoer til masteren, så der kan skiftes mellem spil. Dette gøres ved hjælp af formens **cbGame** (dropdown) og **btnGame** (knap), som det fremgår af oversigten over GUI'ets elementer på tabel 8.3 (s. 50). På kodeudsnit 8.5 (s. 55) ses en beskrivelse af koden bag tryk på **btnGame** med forklarende kommentarer.

```

1 | /// <summary>
2 | /// When clicked, a "switch game"–command is sent to the master
3 | /// </summary>
4 | private void btnGame_Click(object sender, EventArgs e)
5 | {
6 |     // The global variable game is set to the selected game in cbGame.
7 |     // The index of cbGame starts from 0, so add 1
8 |     game = cbGame.SelectedIndex + 1;
9 |     string gameStr;
10 |
11 |     // If games with no. > 99 should be supported, a change in the master–software should be made
12 |     if (game > 99)
13 |         throw new Exception("Games with no. > 99 is currently not supported!!");
14 |
15 |     // If the game no. is < 10, add a leading 0
16 |     if (game < 10)
17 |         gameStr = string.Format("G0{0}", game);
18 |     else
19 |         gameStr = string.Format("G{0}", game);
20 |
21 |     // Send the command
22 |     m_comPort.SendString(gameStr);
23 |     addToLb(string.Format("Sent: {0}", gameStr));
24 | }

```

Code snippet 8.5: Skift mellem spil

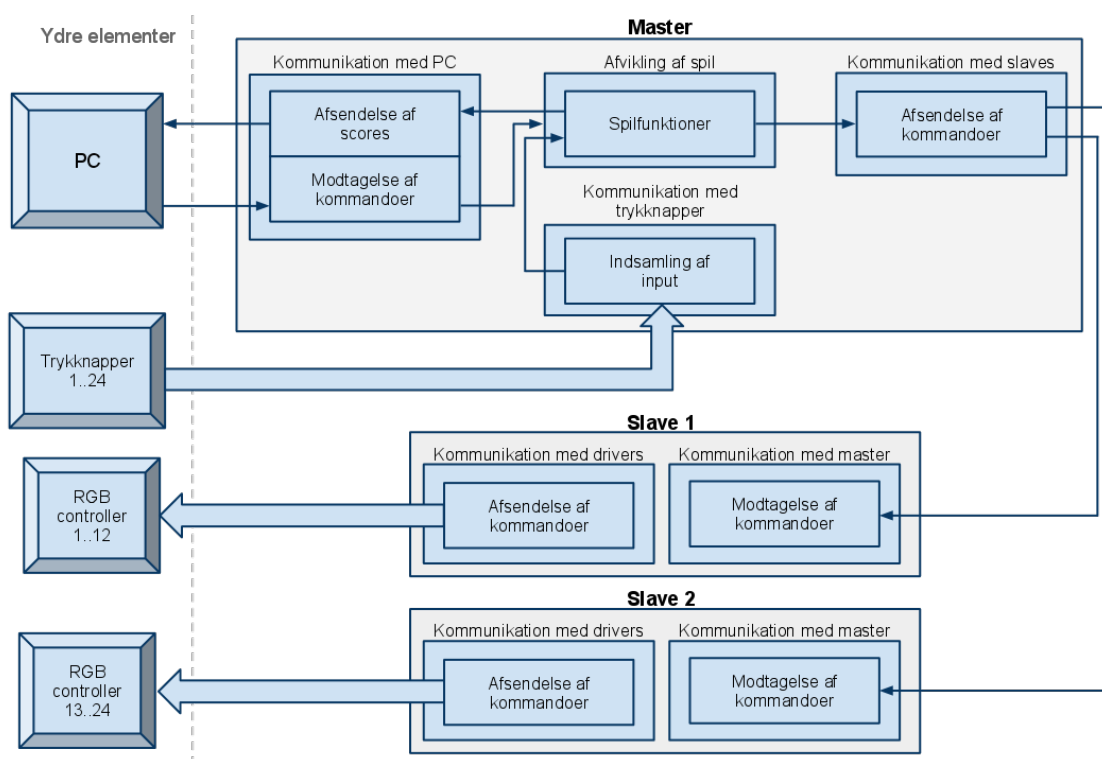
Dette afslutter beskrivelsen af softwaren på PC'en - nogle funktioner er ikke beskrevet, men kan ses i Appendiks B, hvor hele kildekoden vises. Der refereres desuden til sektion 8.2 (s. 56), hvor koden til Arduinoen beskrives - her ses hvordan kommandoerne sendt fra PC'en modtages.

## 8.2 Arduino-software

Arduino-softwaren dækker over softwaren, der ligger i *signalbehandlingen*. Der er - som det fremgår af løsningsforslagene i sektion 4.3 (s. 15) - tale om tre arduino'er, som sammen står for at:

- Modtage kommandoer fra applikationskontrollen
- Sende data til dataopsamling- og behandlingen
- Modtage input fra elev-inputtet
- Sende signal til RGB-controllere

Arduino-softwaren er opdelt i to. Den ene del er *Masteren*, som står for at modtage information fra computeren, afvikle spillene, og sende kommandoerne til den anden del, *Slaverne*, som således står for at videresende kommandoer til RGB-controllerne.



Figur 8.4: Blokdiagram over arduino-softwaren

På figur 8.4 (s. 56) ses en illustration af softwaren, som således ligger til grund for signalbehandlingen. I dette kapitel vil hver blok i diagrammet beskrives, og koden til hver funktion gennemgås. Hvert afsnit vil være markeret med **En fed overskrift**, som markerer begyndelsen på beskrivelsen af den tilsvarende blok i blokdiagrammet. Nogle funktioner vil ikke være beskrevet i kapitlet, men kun fremgå af Appendiks C, hvor hele kildekoden ses.

Bemærk, at følgende notation anvendes til funktioner og variabler i programmet:

- Konstanter skrives med stort
- Globale variabler/funktioner skrives med stort forbogstav
- Lokale variabler/funktioner skrives med lille forbogstav

Bemærk at nogle predefinerede funktioner ikke overholder denne notation.

### 8.2.1 Master

Masteren står som tidligere nævnt for at modtage information fra computeren, afvikle spillene, og sende kommandoer til Slaverne. Som udgangspunkt vil en Arduino - som beskrevet i teori-afsnittet, sektion 6.5 (s. 31) - eksekvere en `setup()`-funktion. I denne funktion initialiseres porte, serial kommunikation o.lign. På kodeudsnit 8.6 (s. 57) ses `setup()` og definitionen af konstanter og globale variabler.

```
1 #define BAUD 115200      // Baud rate for all RS232-communication
2 #define MAXINP 30        // Max chars in input from PC
3 #define MINBTN 1         // Button-numbering minimum (digital IO pins)
4 #define MAXBTN 24        // Button-numbering maximum (digital IO pins)
5
6 // Color definitions
7 #define OFF 13
8 #define RED 38
9 #define GREEN 63
10 #define BLUE 88
11 #define YELLOW 113
12 #define PURPLE 138
13 #define TURQUOISE 163
14 #define WHITE 188
15 #define EXTRA 213
16 #define PULSE 240
17
18 char InpFromPc[MAXINP]; // Used for reading input from the PC
19 int PcStatus = 0;        // Goes 1 if PC-handshaking was successful
20 int SwitchGameFlag = 0;  // For aborting a running game to switch to a new one
21 int Game = -1;           // Current game
22 int BtnsPressed[MAXBTN]; // Holds info for all pressed buttons
23 int I = 0;               // Used for reading serial input from the PC
24
25 void setup() {
26     int p;
27     for (p = MINBTN; p <= MAXBTN; p++) {
28         pinMode(p, INPUT); // Set button as digital input
29     }
30
31     Serial.begin(BAUD); // Master
32     Serial.print("STATUS\n"); // Request status from master
33     Serial1.begin(BAUD); // Slave 1
34     Serial2.begin(BAUD); // Slave 2
35 }
```

Code snippet 8.6: Initialisering og setup()

I kodeudsnittet ses således, hvordan programmets umiddelbare opsætning udføres. Opsætning af konstanter for farverne ses også. Efter endt udførsel af `setup()`, eksekveres funktionen `loop()`. Denne funktion vil blive gentaget indtil spændingsforsyningen til Arduinoen slukkes - derfor skal denne funktion naturligvis stå for at eksekvere resten af programmet samt funktioner til spil. Denne funktion ses på kodeudsnit 8.7 (s. 58).

```

1 void loop() {
2   CollectSerialInput();    // Collect serial input from the computer
3   switch (Game) {          // Switch to the appropriate game
4     case 0:
5       SwitchGameFlag = 0; // Reset SwitchGameFlag
6       Game0();
7       break;
8     case 1:
9       SwitchGameFlag = 0;
10      Game1();
11      break;
12     case 2:
13       SwitchGameFlag = 0;
14       Game2();
15       break;
16   }
17 }

```

Code snippet 8.7: Masterens loop()

Det ses her, at en række andre funktioner kaldes - heriblandt `CollectSerialInput()`, `Game0()` og `Game1()`. `CollectSerialInput()` står, som navnet antyder, for at samle input fra serial-porten - der er her tale om kommandoer fra PC'en. Med henblik på blokdiagrammet på figur 8.4 (s. 56), er der her tale om boksen: *Modtagelse af kommandoer under Kommunikation med PC*.

## Kommunikation med PC

Funktionen `CollectSerialInput()` samler computerens kommandoer ved at lægge hvert tegn, der modtages, ind i et array. Når et newline-character modtages (svarende til `\n`), betragtes det som slutningen på en kommando, og kommandoen behandles. `CollectSerialInput()` skal således kaldes løbende, så kommandoer, der sendes til bufferen kan bearbejdes - funktionen ses i kodeudsnit 8.8 (s. 58) med forklarende kommentarer.

```

1 // Collects, assembles and processes any serial input from the PC
2 void CollectSerialInput() {
3   // If there's something to read from the PC
4   while (Serial.available()) {
5     // The InpFromPc[]-array contains the partial command
6     InpFromPc[I] = Serial.read();
7     // If there's a newline character, process the input
8     if (InpFromPc[I] == '\n') {
9       // If its an OK signal
10      if (InpFromPc[0] == 'O' && InpFromPc[1] == 'K') {
11        PcStatus = 1;
12      }
13
14      // If its a switch-game command (G01 for game one)
15      if (InpFromPc[0] == 'G') {
16        // Fetch the game no, -48 converts from ASCII decimal to int
17        int game = 10*(int(InpFromPc[1])-48);
18        game += (int(InpFromPc[2])-48);
19
20        // Switch to the game
21        SwitchGame(game);

```

```

22 |
23 | // Let the PC know that we're switching to another game
24 | Serial.print("Switching to game ");
25 | Serial.print(game);
26 | Serial.print("\n");
27 | }
28 |
29 | // If its a switch-color-on-display-command (D01,255 for display 01 color 255)
30 | // This command is only for testing/debugging purposes
31 | if (InpFromPc[0] == 'D') {
32 |     // Fetch the display no
33 |     int disp = 10*(int(InpFromPc[1])-48);
34 |     disp += int(InpFromPc[2])-48;
35 |
36 |     // Get the color no
37 |     int color = 100*(int(InpFromPc[4])-48);
38 |     color += 10*(int(InpFromPc[5])-48);
39 |     color += int(InpFromPc[6])-48;
40 |
41 |     // Let the PC know we're switching the color on a display
42 |     Serial.print("Switching display ");
43 |     Serial.print(disp);
44 |     Serial.print(" to color ");
45 |     Serial.print(color);
46 |     Serial.print("\n");
47 |
48 |     // Send the switch-color command to the slave
49 |     SetDisplayColor(disp, color);
50 | }
51 |
52 | // Flush the string array
53 | InpFromPc[0] = '0'; // Numeric zero, not the letter
54 | I = -1;
55 | }
56 | I++;
57 | }
58 | }

```

Code snippet 8.8: Modtagelse af kommandoer fra PC

Bemærk, at funktionen **SwitchGame()** anvendes til at skifte spil. Denne funktion sørger for at stoppe spil, som allerede kører - funktionen ses på kodeudsnit 8.8 (s. 58). Systemet skal naturligvis - jævnfør figur 8.4 (s. 56) - også være i stand til at sende scores til PC'en. Dette gøres ved hjælp af funktionen **SubmitScore()**, som ses på kodeudsnit 8.9 (s. 59). Der er her tale om en simpel kommando som sendes til PC'en - i beskrivelsen af PC-softwaren i sektion 8.1.2 (s. 54), kan læseren se metoden, hvormed disse beskeder modtages.

```

1 | // Submits a score to the PC
2 | void SubmitScore(long score) {
3 |     Serial.print("SCORE:");
4 |     Serial.print(score);
5 |     Serial.print("\n");
6 | }

```

Code snippet 8.9: Afsendelse af scores til PC

## Kommunikation med slaves

Som tidligere nævnt står Slaverne for at styre RGB-controllerne. Masteren skal således sende kommandoer til Slaven, når en bestemt farve ønskes på et display. Dette gøres ved hjælp af funktionen `SetDisplayColor()`, som kaldes fra spil-afviklingen. På kodeudsnit 8.10 (s. 60) ses funktionen med forklarende kommentarer.

```
1 // Sets a specific color on a specific display
2 void SetDisplayColor(int dispNo, int color) {
3     // Don't allow a full pwm signal – won't work when using timers
4     if (color == 255)
5         color = 254;
6     // Don't allow an empty pwm signal for the same reason
7     if (color == 0)
8         color = 1;
9
10    // The "set display color"–command is formatted like this:
11    // DXX,YYY
12    // Where XX is the number of the display, 00–24
13    // And YYY is the color, 001–254
14    if (dispNo <= 12) { // Disp 1–12: Slave 1
15        Serial1.print("D");
16        if (dispNo < 10)
17            Serial1.print("0");
18        Serial1.print(dispNo);
19        Serial1.print(",");
20        if (color < 100)
21            Serial1.print("0");
22        if (color < 10)
23            Serial1.print("0");
24        Serial1.print(color);
25        Serial1.print("\n");
26    } else if (dispNo > 12) { // Disp 13–24: Slave 2
27        dispNo -= 12; // Both slaves operate with display numbering 1–12
28        if (dispNo < 10)
29            Serial2.print("0");
30        Serial2.print("D");
31        Serial2.print(dispNo);
32        Serial2.print(",");
33        if (color < 100)
34            Serial2.print("0");
35        if (color < 10)
36            Serial2.print("0");
37        Serial2.print(color);
38        Serial2.print("\n");
39    }
40 }
```

Code snippet 8.10: Transmittering af farvevalgs-kommandoer til Slaves

## Kommunikation med trykknapper

For at være i stand til at kommunikere med trykknapperne, anvendes som beskrevet i *Design af hardware*, sektion 7.4 (s. 46), en række digitale input-pins på Masteren. Ved hjælp af den simple funk-

tion `CollectButtonInput()`, læses hvor vidt der er trykket på hver trykknapp, og denne information gemmes i det globale array `BtnsPressed[]`. Funktionen, og den relaterede `ClearButtonInput()`, ses i kodeudsnit 8.11 (s. 61).

```

1 // Fills the BtnsPressed-array with all the pressed buttons
2 // Clear the button input with ClearButtonInput()
3 void CollectButtonInput() {
4     int a;
5     for (a = MINBTN; a <= MAXBTN; a++) {
6         if (digitalRead(a) == HIGH)
7             BtnsPressed[a] = 1;
8     }
9 }
10
11 // Clears the button-input set with CollectButtonInput()
12 void ClearButtonInput() {
13     int a;
14     for (a = MINBTN; a <= MAXBTN; a++) {
15         BtnsPressed[a] = 0;
16     }
17 }

```

Code snippet 8.11: Modtagelse af input fra trykknapper

## Afvikling af spil

Til selve afviklingen af spillene anvendes - som det fremgik af `loop()` i kodeudsnit 8.7 (s. 58) - en række funktioner v. navn `Game0()`, `Game1()` og `Game2()`. Disse funktioner er opbygget efter en standard-opstilling, så videreudvikling af softwaren med flere spil er let tilgængelig. På kodeudsnit 8.12 (s. 61) ses en beskrivelse af spil 0 - `Game0()`. Der er her tale om et standby-stadie, som sætter alle displays i "pulse mode", hvor hvert display pulserer i alle de mulige farver. På denne måde kan eventuelle fejl nemt opdages - desuden illustrerer funktionen den generelle opbygning af hvert spil.

```

1 // Game #0
2 // Puts all displays in "Pulse"-mode.
3 void Game0(void) {
4     // Variable initialization
5     int state = 0;
6
7     // The game will continue to run until a "switch game"-command is received
8     while (!SwitchGameFlag) {
9         // Collect commands from the PC
10        CollectSerialInput();
11
12        // Actions to do in the first state
13        if (state == 0) {
14            // Send the "pulse"-command to all displays
15            int i;
16            for (i = 1; i <= 24; i++)
17                SetDisplayColor(i, PULSE);
18
19            // Send the game to the next state

```

```

20     state = 1;
21 }
22
23 // Here, state 1 could be initiated with:
24 // else if (state == 1) { Code for state 1 }
25 }
26 }

```

Code snippet 8.12: Det første spil - Game0

I kodeudsnittet ses det, at spillet bygges op omkring en række forskellige stadier, som Masteren kører igennem. Funktionen `CollectSerialInput()` køres ofte, så en eventuel ”skift spil-kommando fanges, og funktionen `SwitchGame()` kaldes. Denne funktion ses på kodeudsnit 8.13 (s. 62).

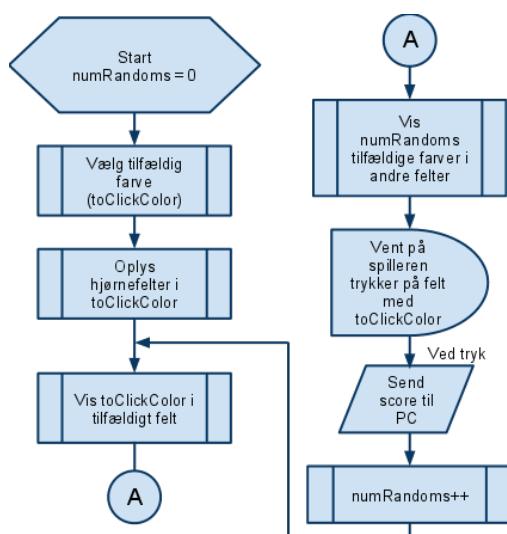
```

1 // Switches to a specific game
2 void SwitchGame(int gameNo) {
3     // Set the SwitchGameFlag, breaking the game-loops
4     SwitchGameFlag = 1;
5
6     // Set the global Game variable to the game
7     Game = gameNo;
8
9     // Turn off all displays
10    int a;
11    for (a = 1; a <= 24; a++)
12        SetDisplayColor(a, OFF);
13 }

```

Code snippet 8.13: Skift af spil

Et eksempel på et spil beskrives - der er her tale om funktionen `Game2()` - resten af spillene kan ses i hele kildekoden i Appendiks C. På figur 8.5 (s. 62) ses et flowchart over funktionen `Game2()`, og derved opbygningen af spil nr. 2.



Figur 8.5: Flowchart over `Game2()`

En tilfældig farve bliver valgt - denne navngives `toClickColor`, og hjørnerne lyser op i denne farve. Et ikke-hjørne display lyser op med `toClickColor`, og `numRandoms` (som starter med at være 0) til-

fældige andre displays lyser op med en anden farve end `toClickColor`. Antallet af sekunder det tager brugeren at trykke på displayet med farven `toClickColor` sendes til PC'en som en score. `numRandoms` stiger med 1 for hver runde, men antager dog maksimalt værdien 19 (antallet af ikke-hjørne displays). Dette spil er bygget op omkring tre stadier - i det første stadie initialiseres variabler, og den tilfældige farve vælges. I næste stadie lyser de forskellige displays i de farver, de skal lyse i, og i det sidste stadig modtages input fra knapperne, og scoren sendes til PC'en. Når tredje stadie er gennemført, fortsætter programmet igen fra starten af stadie to - på denne måde vil programmet fortsætte med at vise tilfældige farver, og tillade spilleren at spille. Alle de forskellige spil programmeres med denne opbygning, så det sikres, at programmet altid er i stand til at stoppe et spil og starte et andet hvis der fra lærerens side er ønske derom.

Hermed afsluttes beskrivelsen af softwaren på Masteren. Masteren er således i stand til at modtage kommandoer fra PC'en samt afvikle spil. Kommandoer sendes dog til Slaverne, som skal være i stand til at modtage kommandoer, og sende et tilsvarende PWM-signal ud til RGB-controllerne.

### 8.2.2 Slave

Her følger en beskrivelse af softwaren på Slaven. Som det fremgår af blokdiagrammet på figur 8.4 (s. 56), står slaven blot for at modtage et signal fra Masteren, behandle det, og sende det videre til RGB-controlleren. Efter samme princip som Master-softwaren i sektion 8.2.1 (s. 57) opsættes en `setup()` og en `loop()`-funktion. Disse funktioner ses på kodeudsnit 8.14 (s. 63).

```

1 #define BAUD 115200      // Baud rate for all RS232-communication
2 #define MAXINP 30       // Max chars in input from Master
3
4 int I = 0;              // Used for reading serial input from the Master
5 char InpFromMaster[MAXINP]; // Used for reading input from the Master
6 void setup() {
7     Serial1.begin(BAUD); // Open the connection to the master
8
9     // Configure PWM outputs and set all displays to OFF
10    int p;
11    for (p = 2; p <= 13; p++) {
12        pinMode(p, OUTPUT); // Init the port as PWM output
13        analogWrite(p, 0);  // Turn off the display
14    }
15 }
16
17 void loop() {
18     CollectSerialInput(); // Collect input from the master
19 }

```

Code snippet 8.14: Setup() og loop()

Her ses, at slavens eneste opgave består i kald til funktionen `CollectSerialInput()`. Funktionen fungerer på samme måde som Masterens `CollectSerialInput()`, som kan ses i kodeudsnit 8.8 (s. 58). Slaven er dog kun i stand til at modtage en kommando, som skifter farven på et display - eksempelvis sætter kommandoen D01,234 PWM-udgangen forbundet med display 1 til at køre med en duty cycle

på  $\frac{234}{255}$ . Dette gøres ved hjælp af et kald til funktion `SetDisplayColor()`, som ses på kodeudsnit 8.15 (s. 64).

```

1 // Sets the color of a display
2 void SetDisplayColor(int dispNo, int color) {
3     analogWrite(dispNo+1, color); // Add one, as the pin numbers are 2-13
4 }

```

Code snippet 8.15: Skift af farve på display vha. `SetDisplayColor()`

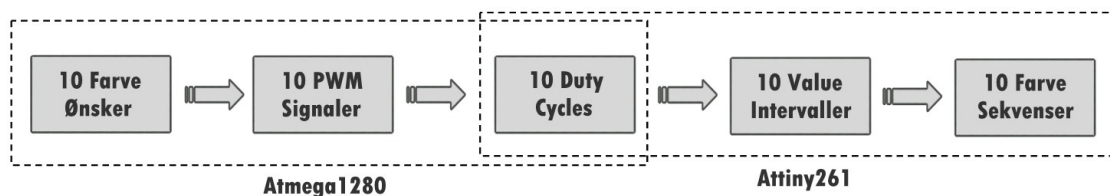
Dette afslutter beskrivelsen af arduino-softwaren - hele kildekoden kan ses i Appendiks C.

## 8.3 Displaydriver-software

Softwaren på displaydriveren er skrevet i BASIC (Beginners All-purpose Symbolic Instruction Code), og designet ved hjælp af IDE'en Bascom som beskrevet i Anvendt Teori sektion 6.5 (s. 31). Displaydriverens software tager udgangspunkt i en kommunikationsprotokollen, som er udarbejdet i forbindelse med signalerne og farveønskerne, som sendes fra slaven til driveren. Disse signaler bliver sendt som PWM signaler med faste perioder, hvor duty cyclen justeres i forhold til hvilken spænding der ønskes at imitere.

For at kunne behandle og anvende PWM-signalerne digitalt, konverteres de til en digital værdi, svarende til tiden, en puls er høj. Dette betyder således, at RGB-controlleren måler hvor lang tid der går fra det modtagne signal går højt, til det bliver lavt igen.

Da denne tid varierer i forhold til det afsendte PWM-signals duty cycle, gør det RGB-controlleren i stand til at udlede flere forskellige værdier fra den enkelte forbindelse til slaven og på denne måde modtage forskellige kommandoer. 10 PWM-signaler med forskellige duty cycles sendes/modtages. Disse duty cycles og de afledte værdier tilknyttes derefter forskellige kodeblokke. For et mere udførligt overblik henvises til figur 8.6 (s. 64) som beskriver hvordan de 10 valg kommunikeres fra slaven til RGB-controlleren.



Figur 8.6: Signalbehandling og omregning af farveønsker fra slaverne til RGB-controllerne

På figur 8.6 (s. 64) ses, hvordan softwaren på slaven omdanner de 10 valg til 10 forskellige PWM-signaler, som repræsenterer 10 forskellige duty cycles. Disse duty cycles modtages, aflæses og konverteres af RGB-controlleren til en værdi, som bestemmer hvilken kodeblok der skal eksekveres, og derved hvilken farve, der skal vises.

Vha. en praktisk test er en duty cycle på 99.6% blevet målt til 79 tidsenheder, hvilket betyder at dette er max-værdien som PWM-signalet kan antage. Når denne information er fastlagt kan de forskellige

PWM-signaler inddeles i 10 intervaller afhængig af deres duty cycle.

Som eksempel kan det beregnes, hvilken værdi farveønsket rød vil antage fra afsendelse (slaven) til modtagelse (RGB-controlleren). På figur 8.7 (s. 65) kan det ses, at rød er tilknyttet et PWM-signal på 25 til 50 som svarer til en duty cycle på 10%-20%. Dette betyder, at RGB-controlleren modtager et signal som ligger imellem disse 2 grænser. Grænserne omregnes til en værdi på følgende måde:

$$Dutycycle \cdot 79 = Val \Rightarrow$$

$$10\% \cdot 79 = 7,9 \approx 8 \wedge 20\% \cdot 79 = 15,8 \approx 16$$

Hvis ovenstående udregning udføres på de 10 valg tilhørende duty cycles kan kommunikationsprotokollen udvides, og digitale værdier, som svarer til hver farve kan fastsættes.

| Farvesignaler som sendes fra Atmega1280 |                 |     |        |            |     |
|-----------------------------------------|-----------------|-----|--------|------------|-----|
| Farve                                   | 8 bit PWM værdi |     |        | Dutycycle  |     |
|                                         | Fra og med      | Til | Median | Fra og med | Til |
| Ingen                                   | 1               | 25  | 13     | 0%         | 10% |
| Rød                                     | 25              | 50  | 38     | 10%        | 20% |
| Grøn                                    | 50              | 75  | 63     | 20%        | 29% |
| Blå                                     | 75              | 100 | 88     | 29%        | 39% |
| Gul                                     | 100             | 125 | 113    | 39%        | 49% |
| Lilla                                   | 125             | 150 | 138    | 49%        | 59% |
| Turkis                                  | 150             | 175 | 163    | 59%        | 69% |
| Hvid'ish                                | 175             | 200 | 188    | 69%        | 78% |
| Puls alle                               | 200             | 225 | 213    | 78%        | 88% |
| Ekstra                                  | 225             | 250 | 238    | 88%        | 98% |
| 8bit PWM max = 255                      |                 |     |        |            |     |

Figur 8.7: Konvertering og omregning af valg til PWM signaler og duty cycles

| Farvesignaler som modtages af Attiny261 |     |            |     |           |
|-----------------------------------------|-----|------------|-----|-----------|
| Dutycycle                               |     | Value      |     | Farve     |
| Fra og med                              | Til | Fra og med | Til |           |
| 0%                                      | 10% | 0          | 8   | Ingen     |
| 10%                                     | 20% | 8          | 15  | Rød       |
| 20%                                     | 29% | 15         | 23  | Grøn      |
| 29%                                     | 39% | 23         | 31  | Blå       |
| 39%                                     | 49% | 31         | 39  | Gul       |
| 49%                                     | 59% | 39         | 46  | Lilla     |
| 59%                                     | 69% | 46         | 54  | Turkis    |
| 69%                                     | 78% | 54         | 62  | Hvid'ish  |
| 78%                                     | 88% | 62         | 70  | Puls alle |
| 88%                                     | 98% | 70         | 77  | Ekstra    |
| Value max = 79                          |     |            |     |           |

Figur 8.8: Konvertering og omregning af duty cycles til værdier og valg

Når kommunikationsprotokollen er fastlagt kan programmeringen af RGB-controlleren påbegyndes.

| I/O oversigt over ATTiny261 (RGB-controller) |          |         |           |                     |
|----------------------------------------------|----------|---------|-----------|---------------------|
| Indgang                                      | Port/Bit | Ben nr. | Udgang    | Bemærkning          |
| Fra slave mega                               | A0       | 20      |           | Signal om farvevalg |
|                                              |          | 2       | RGB diode | Rød                 |
|                                              |          | 4       | RGB diode | Grøn                |
|                                              |          | 8       | RGB diode | Blå                 |

Tabel 8.4: Input/Output oversigt over forbindelser til og fra ATTiny261

På tabel 8.4 (s. 66) ses en I/O oversigt over microcontrollerens ben - denne anvendes som reference under udviklingen af softwaren.

I starten af programmet skal compileren have en række direktiver for at fastsætte hvilke porte, der er tilgængelige på microcontrolleren, og controllerens clock skal defineres. Derefter konfigureres microcontrollerens forskellige porte og timers til input/output, forskellige variabler defineres og aliases oprettes. Dette ses i kodeudsnit 8.16 (s. 66) med forklarende kommentarer.

```

1  ' *****
2  ' COMPILER DIRECTIVES
3  ' *****
4  $regfile "attiny261.DAT"          'Load the ATTiny261 definition
5  $crystal = 8000000                'Set the clock frequency to 8MHz
6
7  ' *****
8  ' CONFIGURATION
9  ' *****
10 Config PORTB = Output             'Configure all bits on port B as output
11 Config PORTA = Input              'Configure all bits on port A as input
12 Config Timer1 = Pwm , Prescale = 32 'Set Timer1 to PWM mode
13 Declare Sub Getval                'Declare that subroutine "Getval" exists
14
15 ' ** SETTING UP PWM **
16 TCCR1A = &B11110011              'TCCR1A bits 7:6 = 11 sets PWM1A to Clear down
17                                  'TCCR1A bits 5:4 = 11 sets PWM1B to Clear down
18                                  'TCCR1A bits 1:0 = 11 enables PWM1A and PWM1B
19
20 TCCR1D.1 = 0                      'TCCR1D bits 1:0 = 01 sets Waveform Generation Mode
21 TCCR1D.0 = 1                      ' to Phase & Frequency Correct PWM
22
23 TCNT1 = &B00000000                'Timer1 count value is cleared, starting the timer
24
25 TCCR1B = &B00000110                'TCCR1B bits 3:0 = 0110 sets Timer1's prescale
26                                  ' to 32
27
28 TCCR1C.0 = 1                      'TCCR1C bit 0 = 1 enables PWM1D
29 TCCR1C.2 = 1                      'TCCR1C bits 3:2 = 11 sets PWM1D to Clear down
30 TCCR1C.3 = 1
31
32 Red Alias OCR1A                   'Equal ocr1a with the color it controls
33 Green Alias OCR1B                 'Equal ocr1b with the color it controls
34 Blue Alias OCR1D                  'Equal ocr1d with the color it controls
35

```

```

36 ' ** SETTING UP VARIABLES **
37 Dim Value As Word           'Define a variable named Value as Word
38                             ' a word is an unsigned int (two bytes)
39 Value = 0

```

Code snippet 8.16: Compilerdirektiver, portopsætning og variabel erklæring

Efter opsætningen startes et "main loop" hvor selve koden til hovedprogrammet køres. Det er denne kode som styrer microcontrollerens in- og output. Loopet kontrollerer konstant værdien af den globale variabel **Value**, og afhængigt af værdien sættes de tre PWM-udganges duty cycles til forskellige, predefinerede værdier. Disse værdier svarer således til en række forskellige farver.

For at opdatere værdien af **Value** kaldes subroutinen **Getval** - alt dette ses i kodeudsnit 8.17 (s. 67).

```

1  ' *****
2  '      MAIN LOOP
3  ' *****
4  Do
5      Call Getval           'Calls the subroutine "Getval", updating Value
6      Beginning:           'Creates label "Beginning"
7      If Value < 8 Then     'If Value is less than 8,
8          Red = 0           'Turn off red
9          Green = 0         'Turn off green
10         Blue = 0          'Turn off blue
11     End If
12
13     If Value >= 8 And Value < 15 Then 'If value is 8-15,
14         Red = 255         'Turn on red 100%
15         Green = 0         'Turn off green
16         Blue = 0          'Turn off blue
17     End If
18
19     If Value >= 15 And Value < 23 Then
20         Red = 0
21         Green = 255
22         Blue = 0
23     End If
24     ....
25     If Value >= 54 And Value < 62 Then
26         Red = 255
27         Green = 255
28         Blue = 255
29     End If
30 Loop                       'End of the main program loop
31
32 Sub Getval                 'Start of the "Getval" subroutine
33     Value = 0              'Set Value to 0
34     Do                    'Wait until Pin A0 is 0
35     Loop Until Pina.0 = 0
36     Do                    'Wait until Pin A0 is 1
37     Loop Until Pina.0 = 1
38     Do                    'When Pin A0 is 1 then:
39         Value = Value + 1  'Increase value by 1 as fast as the clock allows
40     Loop Until Pina.0 = 0  'Keep increasing value until Pin A0 is 0
41 End Sub                    'End subroutine

```

Code snippet 8.17: Compiler directives

Dette var et udsnit af de forskellige funktioner som programmeringen til RGB-controlleren består af. For at se hele koden se Appendiks E 11.5.

## 8.4 Opsummering af software

I forhold til alle relevante krav i kapitel 3, er softwaren som er blevet udviklet i forrige afsnit i stand til at efterkomme dem alle. Dette betyder at PC softwaren:

- Indeholder en brugerinterface så brugeren ikke skal kende til kodesprog e.lign.
- Kan modtage og behandle data fra elevinputtet og udskrive resultater med grafer.
- Kan vælge og starte et spil.

Samtidig betyder dette også at Arduino programmeringen kan:

- Modtage og gemme input på baggrund af en spillers præstationer.
- Sendte informationer videre til PC og modtage spilvalg fra PC via. USB.
- Afvikle spil og sende signaler til RGB-controllerer.
- Sendte informationer til RGB-controllerer via impulser i en enkelt forbindelse.

Tilsidst betyder dette også at RGB-Controller programmeringen kan:

- Styre 1 display individuelt men 24 displays samlet.
- Modtage signaler vha. en enkelt forbindelse.
- Styre displays til at vise mere end 7 forskellige farver.

Dette betyder altså som sagt at softwaren er i stand til at efterkomme alle krav som har været stillet. Det skal dog nævnes at softwaren ikke er udformet som først forventet, resultater under test af systemet viste at hver af Seeeduinoerne har 2 PWM udgange med dobbelt så høj frekvens som resten. Dette betød at i alt 4 RGB-controllere måtte udvikles til at modtage PWM signaler med dobbelt så høj frekvens men stadig udføre de samme kodeblokke som resten. Oprindeligt var det også ment at RGB-controlleren skulle behandle PWM signalet vha. sin ADC men da dette viste sig problematisk, primært pga. Seeediunoens lav-frekvente PWM signal, blev metoden som beskrevet i udviklingsafsnittet benyttet i stedet.

## Udførelse af accepttest

Der kan udføres en række tests for at konstatere om produktet lever op til kravspecifikationen. Disse tests udføres ud fra kravspecifikationen og accepttestspecifikationen i kapitel 5 (s. 23). Der kan desuden udføres en række tests for at analysere anvendeligheden af produktet specificeret i krav 1-4 i sektion 5.1 (s. 23). Den korrekte udførelse af accepttesten ville være med forskellige brugere og testledere, men denne test er begrænset til én testperson.

Den valgte testperson er en normalt fungerende pige på 4 år med normalt syn og hørelse. Det vurderes, at denne testperson vil være passende til accepttesten, da personens niveau ikke vil overstige CDH-elevernes. Dette niveau er beskrevet i problemanalysen i sektion 2.2 (s. 7).

De tekniske krav 5-10 i sektion 5.1 (s. 23) er blevet testet inden kravene 1-4. Prototypen er dimensioneret til at kunne opskaleres ved, at overvejelserne omkring opbygning er lavet ud fra det færdige produkt. Test af diodernes mulighed for at vise 7 farver blev gjort med et program som skifter mellem farverne. PWM signalet er testet ved at sende signaler mellem slave og display driveren. RS232 signalet er testet ved at sende et signal fra PC til masteren. Trykknapperne er testet med et program hvor dioden lyser når der trykkes på knappen. Disse krav opfyldes tilfredsstillende.

Resultaterne hvor både spil og brugerfladen på en PC er testet, opfylder kravene. Det blev dog bemærket at det kan være svært at skelne alle 7 farver fra hinanden. Blålige farver (herunder blå, violet, turkis og hvid) er svære at skelne fra hinanden. Spil hvor kun 3-4 farver benyttes virker efter hensigten. Herunder ses de yderligere tests og resultaterne deraf.

|                    |                                                                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Krav               | En elev eller person skal kunne "spille" et spil på produktet med det samme efter reglerne er blevet forklaret.                                                                                                                                    |
| Test beskrivelse   | Testpersonen får reglerne omkring et givet spil forklaret hvorefter det startes og testen udføres.                                                                                                                                                 |
| Forventet resultat | Testpersonen danner hurtigt overblik over felterne og deres tilhørende knapper hvilket gør at testpersonen kan spille uden problemer.                                                                                                              |
| Resultat           | Testpersonen kan uden forklaring uddybende forklaring forstå opbygningen af spillet og sammenhængen mellem knap og diode. Testpersonen kunne med enkelte anvisninger forstå simple spil, og efter kort forklaring forstå et mere kompliceret spil. |
| Konklusion         | Produktet er i stand til at opfylde kravet i tilfredsstillende grad.                                                                                                                                                                               |

Tabel 9.1: Accepttest af krav nr. 1

|                    |                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Krav               | En almindeligt fornuftig person skal kunne vælge og igangsætte spil fra computeren via USB efter at have set et eksempel med kort instruks.                                                                    |
| Test beskrivelse   | En ekstern person inddrages og bliver præsenteret for et eksempel hvor et spil igangsættes, derefter påbegyndes testen hvor testpersonen er på egen hånd.                                                      |
| Forventet resultat | Den eksterne person kan vælge og påbegynde spil uden problemer.                                                                                                                                                |
| Resultat           | Den eksterne person var efter gennemlæsning af manualen og korte instrukser i stand til at igangsætte spil. Efter få minutter var personen desuden i stand til at plotte en graf over spillerens præstationer. |
| Konklusion         | Produktet er i stand til at opfylde kravet i tilfredsstillende grad.                                                                                                                                           |

Tabel 9.2: Accepttest af krav nr. 2

|                    |                                                                                                                    |
|--------------------|--------------------------------------------------------------------------------------------------------------------|
| Krav               | Produktet skal kunne måle, sende og lagre data omkring spilleren til et stykke software på PC'en via USB.          |
| Test beskrivelse   | Et spil gennemføres af testpersonen hvorefter det tjekkes om softwaren har lagret korrekte informationer på PC'en. |
| Forventet resultat | PC'en har modtaget og lagret de korrekte informationer.                                                            |
| Resultat           | Efter endt spil er information om spillerens præstationer korrekt lagret på PC'en.                                 |
| Konklusion         | Produktet er i stand til at opfylde kravet i tilfredsstillende grad.                                               |

Tabel 9.3: Accepttest af krav nr. 3

|                    |                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Krav               | Produktet skal kunne behandle spillerdata og vise/udskrive grafer som en almindeligt fornuftig person kan forstå uden forklaringer. |
| Test beskrivelse   | En graf forsøges udskrevet fra foregående test hvorefter den præsenteres for en ekstern person.                                     |
| Forventet resultat | Den eksterne person kan forstå og forklare grafen uden hjælp.                                                                       |
| Resultat           | Den eksterne person var i stand til at forstå grafen uden yderligere forklaringer.                                                  |
| Konklusion         | Produktet er i stand til at opfylde kravet i tilfredsstillende grad.                                                                |

Tabel 9.4: Accepttest af krav nr. 4

## Konklusion

Der er med udgangspunkt i problemstillingen udviklet en teknisk løsning som kan hjælpe med at stimulere de synshæmmede børn. Denne løsning er designet som en prototype med 24 felter som vil kunne opskaleres til den ønskede ramme fra CDH, hvis der bygges trykfølsomme farvede displays i fuld størrelse. Prototypen har følgende egenskaber:

- Den kan vise 7 forskellige farver i 24 uafhængige displays
- Den kan modtage inputs fra 24 uafhængige trykknapper
- Den kan afvikle forskellige spil på baggrund af brugervalg
- Den kan måle og behandle data på baggrund af spillerens præstationer
- Den kan illustrere spillernes præstationer over tid i en graf

Egenskaberne opfylder kravene som blev stillet i krav til produktet, kapitel 3 (s. 9).

I produktudviklingen er der samtidig blevet gennemført en accepttest af produktet på baggrund af kravspecifikationen kapitel 5 (s. 23). Denne test omhandlede primært produktets brugervenlighed og viste at systemet var brugervenligt da:

- Det er nemt for læreren at vælge og igangsætte spil efter at have læst Quick Start Guiden (Se appendiks 11.6)
- Spillet er nemt at benytte for eleven efter at simple instrukser er givet
- Produktet genererer grafer og resultater for at vise en spillers præstation
- Det er nemt at forstå resultater og grafer som produktet danner.

Hvis produktet senere skal videreudvikles vil ovenstående test blive udført i samarbejde med CDH således produktet kan tilpasses mere nøjagtigt i forhold til målgruppen.

Det færdige system kan ses på det overordnede systemdiagram, figur 7.1 (s. 41). Prototypen fungerer således ved at programmet, som afvikler spillet på masteren sender kommandoer til slaverne, vha. RS232, når et display skal tænde. Slaverne sender derefter signal til den respektive RGB-controller om hvilken farve det valgte display skal vise vha. PWM signaler. Når spilleren vha. trykknappen sender et signal tilbage registrerer masteren dette på sine digitale inputs. Afhængigt af det valgte spil registreres respons og responstid der lagres som data. Disse data bliver efterfølgende behandlet til resultater. Til sidst kan brugeren vha. PC-softwaren vælge at vise grafer eller resultater som evt. kan udskrives.

På trods af, at de blålige farver, som prototypens dioder fremstiller, ikke kan skelnes tydeligt fra hinanden, har prototypen i tilfredsstillende grad opfyldt de opstillede krav og bestået accepttesten. Det færdige produkt er således realistisk at fremstille ud fra princippet, som illustreres vha. prototypen.

## Appendiks oversigt

Alle apendiks findes på vedlagte CD i mappen appendiks.

**11.1 Appendiks A: Overordnet systemdiagram**

**11.2 Appendiks B: PC-software kildekode**

**11.3 Appendiks C: Arduino-kildekoden**

**11.4 Appendiks D: IC printudlæg**

**11.5 Appendiks E: Display-driver Software**

**11.6 Appendiks F: Manualer**

## Bilag oversigt

Alle bilag er placeret på vedlagte CD i mappen bilag.

### **12.1 Projektkatalog**

### **12.2 Produktforslag**

### **12.3 Interview med CDH**

### **12.4 Datablad til ATTiny261**

### **12.5 Datablad til anvendt MOSFET**

### **12.6 Datablad til RGBB diode**

# Referencelister

## Figurer

|      |                                                                                                                                                                       |    |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1  | CDH-Væggen fra Projektkataloget . . . . .                                                                                                                             | 4  |
| 2.1  | Blokdiagram med udgangspunkt i det tekniske hjælpemiddel . . . . .                                                                                                    | 6  |
| 3.1  | Kommunikation mellem de enkelte enheder i systemet. Bemærk at eleven modtager et output fra display og kan trykke på det dertilhørende input og give respons. . . . . | 9  |
| 4.1  | Ramme fra CDH . . . . .                                                                                                                                               | 15 |
| 4.2  | 8x8 matrix LED display - købes som færdigt produkt . . . . .                                                                                                          | 18 |
| 4.3  | Illustration af de to forskellige 7-segment displays . . . . .                                                                                                        | 19 |
| 4.4  | Illustration af felter m. 7-segment displays (a) og forskellige farvefelter (b) . . . . .                                                                             | 19 |
| 4.5  | Muligheder for visning af forskellige tal og farver . . . . .                                                                                                         | 20 |
| 6.1  | Diagram over system . . . . .                                                                                                                                         | 24 |
| 6.2  | Blokdiagram over ATmega1280 [4] . . . . .                                                                                                                             | 25 |
| 6.3  | Eksempler på to forskellige duty cycles med PWM . . . . .                                                                                                             | 27 |
| 6.4  | 8-bit timer og PWM-udgang v. sammenligningsværdi på 100 . . . . .                                                                                                     | 27 |
| 6.5  | Eksempel på RS232-kommunikation hvor et G sendes . . . . .                                                                                                            | 29 |
| 6.6  | Eksempler på forskellige bitantal . . . . .                                                                                                                           | 30 |
| 6.7  | Successive Approximation ADC . . . . .                                                                                                                                | 31 |
| 6.8  | Fra kode til program . . . . .                                                                                                                                        | 32 |
| 6.9  | Eksempel på opsætning af simpel Windows Forms applikation . . . . .                                                                                                   | 35 |
| 6.10 | Eksempel-programmet efter tryk på knap 1 . . . . .                                                                                                                    | 36 |
| 6.11 | Forskellige transistortyper . . . . .                                                                                                                                 | 38 |
| 6.12 | Symboler for P-MOSFET og N-MOSFET transistorer [15] . . . . .                                                                                                         | 38 |
| 6.13 | Mosfets som tændt eller slukket . . . . .                                                                                                                             | 38 |
| 6.14 | Diodesymboler . . . . .                                                                                                                                               | 39 |
| 7.1  | Detaljeret systemdiagram . . . . .                                                                                                                                    | 41 |
| 7.2  | 3D tegning af vores testopstilling . . . . .                                                                                                                          | 42 |
| 7.3  | Simpelt diagram over RGBdiode . . . . .                                                                                                                               | 42 |
| 7.4  | Diagram af et enkelt felts forbindelser . . . . .                                                                                                                     | 43 |
| 7.5  | Udsnit af RGB dioders datablad . . . . .                                                                                                                              | 43 |
| 7.6  | Udsnit af RGB dioders datablad . . . . .                                                                                                                              | 43 |
| 7.7  | Diagram over funktionen af RGB-controllerne . . . . .                                                                                                                 | 44 |

|     |                                                                                        |    |
|-----|----------------------------------------------------------------------------------------|----|
| 7.8 | Diagram over ideen med RGB-controlleren . . . . .                                      | 45 |
| 7.9 | fotografi af displaydriver . . . . .                                                   | 45 |
| 8.1 | Klassediagram, som viser relationen mellem players og scores . . . . .                 | 48 |
| 8.2 | Illustration af PC-softwarens kommunikation med de ydre elementer . . . . .            | 49 |
| 8.3 | PC-softwarens GUI med angivelse af elementnavne . . . . .                              | 51 |
| 8.4 | Blokdiagram over arduino-softwaren . . . . .                                           | 56 |
| 8.5 | Flowchart over Game2() . . . . .                                                       | 62 |
| 8.6 | Signalbehandling og omregning af farveønsker fra slaverne til RGB-controllerne . . . . | 64 |
| 8.7 | Konvertering og omregning af valg til PWM signaler og duty cycles . . . . .            | 65 |
| 8.8 | Konvertering og omregning af duty cycles til værdier og valg . . . . .                 | 65 |

## Tabeller

|     |                                                                                        |    |
|-----|----------------------------------------------------------------------------------------|----|
| 4.1 | Eksempel på en SQL-tabel . . . . .                                                     | 16 |
| 6.1 | Forskellige udtryk, der ofte anvendes indenfor software til microprocessorer . . . . . | 32 |
| 6.2 | Eksempel på opbygningen af en tabel, der viser elever . . . . .                        | 37 |
| 8.1 | Tabellen Players . . . . .                                                             | 48 |
| 8.2 | Tabellen Scores . . . . .                                                              | 48 |
| 8.3 | Elementer i GUI'et . . . . .                                                           | 50 |
| 8.4 | Input/Output oversigt over forbindelser til og fra ATTiny261 . . . . .                 | 66 |
| 9.1 | Accepttest af krav nr. 1 . . . . .                                                     | 69 |
| 9.2 | Accepttest af krav nr. 2 . . . . .                                                     | 70 |
| 9.3 | Accepttest af krav nr. 3 . . . . .                                                     | 70 |
| 9.4 | Accepttest af krav nr. 4 . . . . .                                                     | 70 |

## Litteratur

- [1] Foreningen af Danske DøvBlinde (FDDB). Handicap - døvblindhed. december 2010.  
<http://www.handicap.dk/dokumenter/publikationer/dh-vejviser/kapitel15>.
- [2] Mike Chapple. Sql fundamentals. The Internet, November 2010.  
<http://databases.about.com/od/sql/a/sqlfundamentals.htm>.
- [3] B. Collis. An introduction to practical electronics, microcontrollers and software design. The Internet, September 2010. [www.techideas.co.nz](http://www.techideas.co.nz).
- [4] Atmel Corporation. Atmega640/1280/1281/2560/2561 datasheet, September 2010.
- [5] et al. David A. Mellis. The arduino language reference. The Internet, Oktober 2010.  
<http://arduino.cc/en/Reference/HomePage>.
- [6] MCS Electronics. Bascom avr help reference, November 2010. ver. 1.11.9.7 document build 26.

- [7] Videnscentret for Døvblindblevne, Juni 2010. <http://www.servicestyrelsen.dk/dovblindfodt>.
- [8] Videnscentret for Døvblindblevne. Usher syndrom - en kort beskrivelse, Januar 2010. [http://www.dbcent.dk/Usher\\_syndrom](http://www.dbcent.dk/Usher_syndrom).
- [9] Center for Døvblindhed og Høretab. Definition af døvblindhed. The Internet, Juni 2008. <http://www.dbc.rn.dk/Doevblindhed/DefinitionAfDoevblindhed/>.
- [10] Center for Ligebehandling af Handicappede. Hørehandicap, Juni 2010. <http://www.handicapmainstreaming.dk/vaerktoejskasse/kommunikation-med-mennesker-med-funktionsnedsaettelse/hoerehandicap/>.
- [11] Det Centrale Handicapråd. Handicap og ligebehandling - et refleksionspapir. The Internet, Juli 2001. <http://www.dch.dk/publ/handicapogligebehandling/doc2.html>.
- [12] John Hewes. Electronic components reference. The Internet, November 2010. <http://www.kpsec.freeuk.com/components/tran.htm>.
- [13] Jon Jagger John Sharp. *Microsoft Visual C# .NET Step by Step-Version 2003*. 2003. ISBN: 0-7356-1909-3.
- [14] Anders Martin Rundh & Mads Kopperholt. Lærer - skoleafdeling for børn med høretab, 2010. Center for Døvblind og Høretab Kollegievej 1 9000 Aalborg [www.aalborgskolen.rn.dk](http://www.aalborgskolen.rn.dk).
- [15] Gary Madsen. Hvad er fet'en for en fætter?, Februar 2008.
- [16] Ole E. Mortensen. Op mod 15.000 ældre over 80 år med kombineret høre- og synsnedsættelse i danmark? kan det passe?? *Møde i Det Faglige Netværk*, 2006. <http://www.dbcent.dk/vcfdbb/subpage151.aspx#op>.
- [17] Dr. Rod Nave. Hyperphysics - successive approximation adc, december 2010. <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/adc.html>.
- [18] Dana Nourie. Getting started with an integrated development environment (ide). The Internet, Marts 2005. <http://java.sun.com/developer/technicalArticles/tools/intro.html>.
- [19] Døvblindecentret og Aalborgskolen. Portræt af døvblindecentret, 2009. [http://www.cdh.rn.dk/NR/rdonlyres/C35E1DFA-F0BB-44F4-AD6A-C13865B27B31/0/DBC\\_tidsskrift\\_2009web.pdf](http://www.cdh.rn.dk/NR/rdonlyres/C35E1DFA-F0BB-44F4-AD6A-C13865B27B31/0/DBC_tidsskrift_2009web.pdf).
- [20] et al. Paolo Pavan. *Flash Memory Cells - An Overview*. August 1997.
- [21] Craig Peacock. Interfacing the serial/rs232 port, Oktober 2010.
- [22] Egon Rasmussen. Analog teknik, 2001. ISBN:87-600-0354-5.
- [23] Steven Campbell Hamid A. Toliyat. *DSP-Based Electromechanical Motion Control*. CRC Press, 2004. ISBN: 978-0-203-48633-7.
- [24] Martin H. Weik. A third survey of domestic electronic digital computing systems, 1961. <http://ed-thelen.org/comp-hist/BRL61.html>.