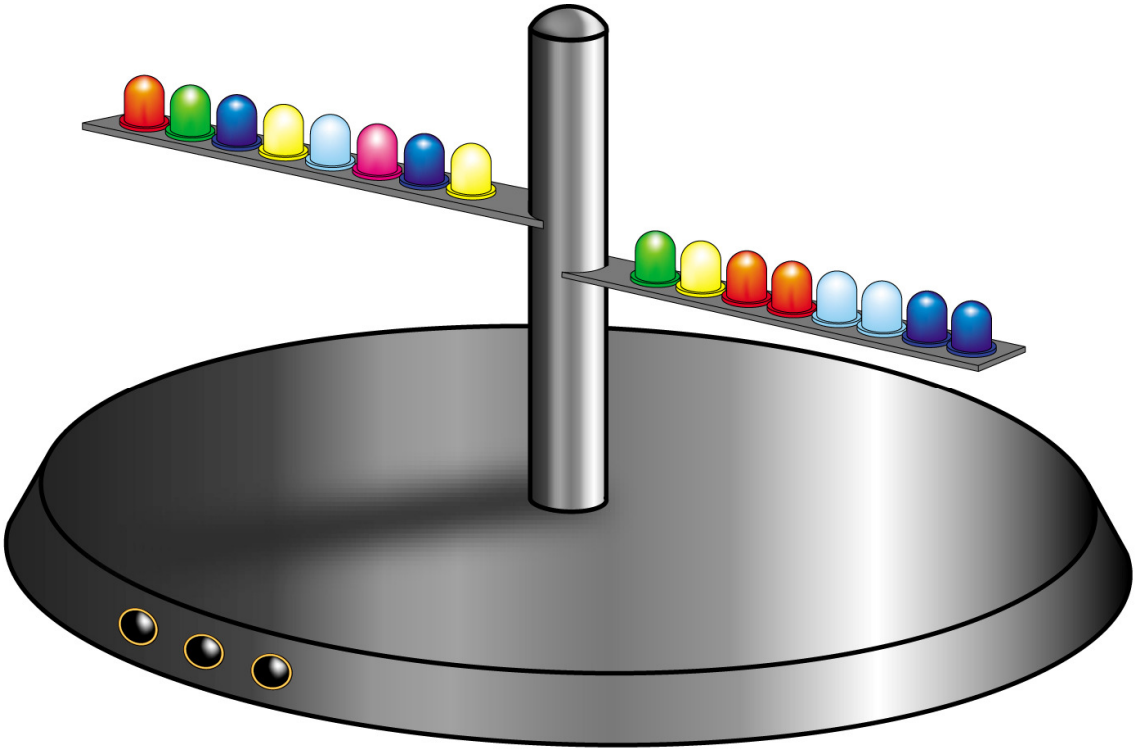
3D LED-display

Concepten en implementatie van roterend gedeelte



Technische Universiteit Delft
Juni 2007
Delft

F.C. van Dam, 1159720
M.R. van der Leije, 1222740

Begeleider:
A.J. van Genderen

White Rabbit Co.

Inhoudsopgave

Voorwoord

Dit rapport is het bachelor afstudeerverslag van F.C. van Dam en M.R. van der Leije. Het bachelor afstuderen in het derde jaar van de bacheloropleiding Elektrotechniek, aan de Technische Universiteit te Delft, bestaat uit het schrijven van een businessplan met zes studenten en het implementeren van het product dat hierbij hoort. Ons product, het 3D LED-display, wordt opgesplitst in drie subonderdelen die door drie verschillende tweetallen worden gemaakt. Dit rapport beschrijft het ontwerptraject, de keuzes en de uiteindelijke implementatie van het roterende gedeelte van het display. Tijdens het hele traject zijn wij vakkundig begeleid door dr. ir. A.J. van Genderen.

6/6/2007
F.C. van Dam
M.R. van der Leije

Samenvatting

De toekomst van beeldscherm is 3D. White Rabbit Co speelt hierop in en is volop bezig met de ontwikkeling van hun 3D LED-display. Het display maakt gebruik van roterende LED's waardoor het mogelijk is om een afbeelding van alle kanten te bekijken.

De aanleiding van dit rapport is dan ook het documenteren van de ontwikkeling van het 3D LED-display door White Rabbit Co. Het rapport bevat alle stappen tot de implementatie van het roterend gedeelte, waarbij nadruk op een heldere en stabiele weergave van verschillende kleuren ligt.

Er zijn een tweetal keuzes gemaakt om op het roterende gedeelte een mooie 3D-kleurenafbeelding te laten zien. Er is gekozen om het LED-display op te bouwen uit horizontale balkjes van acht RGB (Rood, Groen, Blauw) LED's, omdat dit een stabiele weergave oplevert. Deze RGB LED's worden parallel aangestuurd, omdat dit voor een heldere weergave zorgt.

Om te zorgen dat het roterende gedeelte goed werkt, moesten er naast de gekozen concepten ook een viertal subonderdelen afgesproken en gemaakt worden.

- Een lichtsluis die gemaakt is om het display adaptief aan de snelheid te maken.
- Een sleepcontact om voeding en datacommunicatie van het vaste naar het roterende deel te brengen.
- LIN(Local Interconnect Network)--hardware met een zelfontwikkelt protocol om een foutloze dataoverdracht te realiseren.
- Een plaatje is opgebouwd uit een aantal cirkels boven elkaar, die elk weer in 64 stukken van drie bytes (RGB van 8 LED's) zijn opgedeeld.

Aanbevolen wordt om een prototype van metaal en plastic te maken om wrijvingen te minimaliseren. Ook het ontwikkelen van PCB's (Printed Circuit Boards) voor de hardware zullen voor significante verbeteringen zorgen. De microcontroller, waarmee de LED's aangestuurd worden, kan uitgebreid worden met extra geheugen om grotere plaatjes op te slaan. Als laatste kan de resolutie van het display verhoogd worden door de LED's sneller aan te sturen.

1 Inleiding

Al geruime tijd zijn er al ontwikkelingen gaande op het gebied van 3D-displays. Een aantal bedrijven, zoals Philips, zijn in volle gang met de ontwikkeling hun eigen “3D-displays”. Zo wil ook White Rabbit Co. een 3D-display op basis van roterende Light Emitting Diodes (LED’s). Anders dan bij de concurrenten is het met deze roterende techniek wel mogelijk om rond de 3D-afbeelding te lopen.

Om een 3D-afbeelding te laten zien is het cruciaal dat de individuele LED’s op het juiste moment aangestuurd worden. Naast de software die hiervoor ontwikkeld dient te worden, is het ook noodzakelijk dat er foutloos data op het roterende deel ontvangen kan worden.

Het doel van het project is het implementeren van het aansturen van de roterende LED-matrix en het verwerken van ontvangen data. Hierbij wordt de nadruk gelegd op een heldere en stabiele weergave van verschillende kleuren. Voorlopig wordt nog niet gekeken naar de uiteindelijke mechanische uitwerking en behuizing van het product.

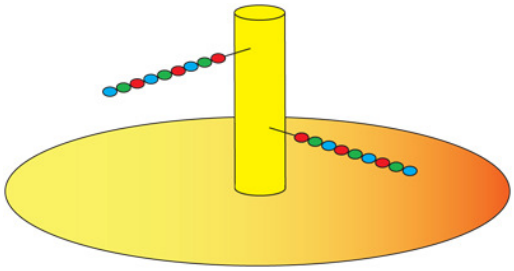
De opbouw van dit rapport is als volgt. Hoofdstuk 2 beschrijft de ontwerpopdracht. Vervolgens staan in hoofdstuk 3 eisen met betrekking tot de ontwerpopdracht. In hoofdstuk 4 worden de verschillende concepten die voortvloeien uit de eisen uitgelegd. Vervolgens wordt in hoofdstuk 5 het definitieve ontwerp afgewogen en gekozen uit de verschillende concepten. In hoofdstuk 6 wordt het implementatietraject van het gekozen ontwerp beschreven. Tot slot komen in hoofdstuk 7 de conclusies en aanbevelingen aan bod.

2 Ontwerpopdracht: aansturen van de LED's voor het 3D LED-display

In dit hoofdstuk wordt de ontwerpopdracht, het aansturen van de LED's voor het 3D LED-display, beschreven. In paragraaf 2.1 wordt het product omschreven. Vervolgens wordt het te ontwerpen deelsysteem, het aansturen van de LED's, afgebakend en toegelicht (§2.2).

2.1 Het 3D LED-display

In figuur 2.1 is te zien hoe het 3D LED-display opgebouwd zal worden: een roterende plaat met daarop een cilinder waaraan staafjes met LED's bevestigd zijn. Door voldoende staafjes te gebruiken en de plaat sneller te laten draaien dan het oog kan waarnemen, kan er een realistisch 3D beeld gerealiseerd worden.

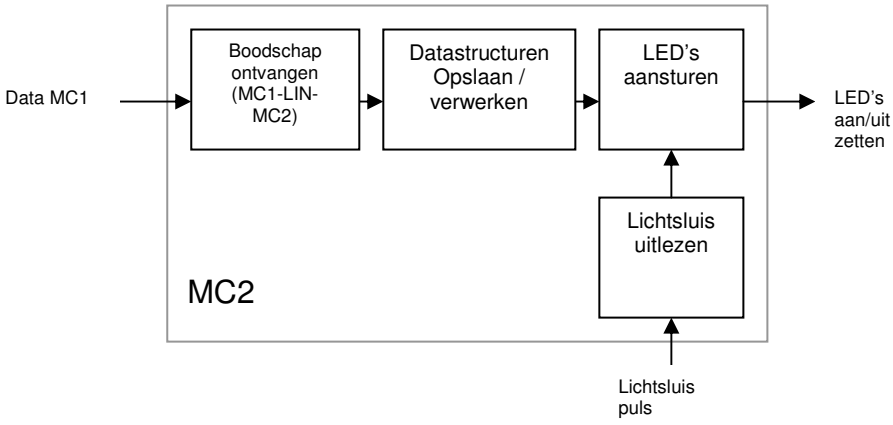


Figuur 2.1 - Schematische weergave LED Display

Het 3D LED-display moet een plaatje vanaf de PC weer kunnen geven op de draaiende LED's. Om dit te kunnen doen wordt er een gebruikersinterface gemaakt die via USB met een stilstaande microcontroller (MC1) communiceert. Vervolgens speelt deze de ontvangen data door naar de draaiende microcontroller (MC2), die dit weergeeft op de LED's. Ook kan het display worden aangestuurd door middel van knoppen.

2.2 Deelsysteem: aansturen van de LED's

Het draaiende gedeelte krijgt data van MC1 via het Local Interconnect Network (LIN) protocol. Deze data dient geanalyseerd te worden om te bepalen wat er mee gedaan moet worden. Zo kunnen er bijvoorbeeld plaatjes opgeslagen worden. Nadat de ontvangen data is verwerkt kunnen de LED's weer aangestuurd worden. Om de LED's op tijd te schakelen geeft de lichtsluis een puls wanneer de plaat een rondje gedraaid heeft (zie figuur 2.2). Het volledige overzicht van het blokschema staat in bijlage 1.



Figuur 2.1 – Blokschema: aansturen van de LED's
(volledig blokschema: zie bijlage 1)

3 Programma van eisen

Dit hoofdstuk behandelt het programma van eisen. Dit bestaat uit twee onderdelen, namelijk de eisen met oog op het display (§ 3.1) en de randvoorwaarden die opgelegd worden door het product (§ 3.2).

3.1 Eisen aan het display

Voor het draaiende gedeelte van het 3D LED-display zijn er eisen die voor een goede weergave van de 3D afbeelding zorgen. Deze zijn hieronder onderverdeeld.

Eisen voor de LED's
- Minimaal 8 verschillende kleuren
- Minimaal 64 keer per omwenteling aan en uit op minimaal 25 Hz
- Goede weergave bij weinig of geen omgevingslicht

Eisen voor de aansturing
- Zo stabiel mogelijke weergave behalen
- Zo min mogelijk aansluitingen van de microcontroller gebruiken

Eisen voor de opstelling van de LED's
- Zo stabiel mogelijk ronddraaien
- Zo goed mogelijk zijaanzicht
- Zo goed mogelijk bovenaanzicht
- Minimaal 32 LED's

3.2 Randvoorwaarden die voortvloeien uit het product

Voor het product zijn er verschillende ontwerpkeuzes gemaakt. Deze keuzes geven randvoorwaarden, waar het deelsysteem aan moet voldoen.

De volgende mechanisch componenten en hardware dienen in het product gebruikt te worden:
- PIC18F4550 microcontroller
- Sleepcontacten
- Local Interconnect Network (hardware)
- Motor

4 De concepten van de subonderdelen

In dit hoofdstuk worden de verschillende concepten voor het roterende gedeelte behandeld. Allereerst worden in § 4.1 de verschillende LED opstellingen besproken. Vervolgens worden in § 4.2 de mogelijkheden om verschillende kleuren te maken behandeld. En als laatste worden in § 4.3 methoden voorgelegd om de LED's aan te sturen.

4.1 De opstelling van het LED display

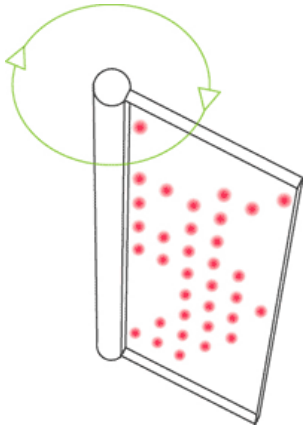
Er zijn verschillende LED-opstellingen mogelijk. Er zijn drie verschillende opties bekeken, namelijk:

- Vlakke rechtopstaande plaat van LED's (§ 4.1.1)
- Horizontale balkjes van LED's op verschillende hoogtes (§ 4.1.2)
- Verticale balkjes van LED's op verschillende afstanden van de middenas (§ 4.1.3)

4.1.1 Vlakke plaat

De vlakke plaat is een plaat met LED's aan één of twee kanten aangebracht. Er zijn twee mogelijkheden: (1) de plaat loopt van de rand van de cirkel tot de middenas, of (2) de plaat is even lang als de diameter van de cirkel en loopt dus van de ene rand naar de andere door de middenas.

Mogelijkheid 1 heeft als nadeel dat er tijdens het draaien instabiliteit kan onstaan. De oorzaak hiervan is dat het gewicht aan een kant geconcentreerd is. Diis echter eenvoudig te verhelpen, omdat het gewicht aan één kant zit. De gewichtsverdeling is veel egaler bij mogelijkheid 2. Deze mogelijkheid zal dus stabiel zijn.



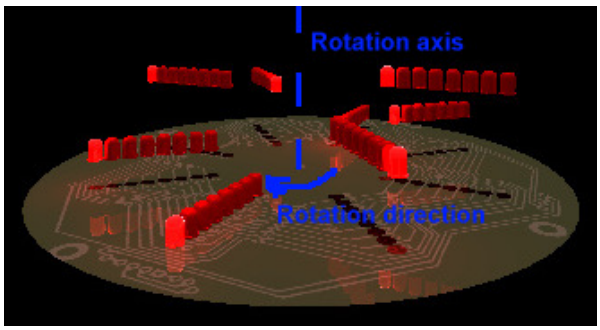
Figuur 4.1 – LED opstelling: vlakke plaat

Een vlakke plaat heeft in het algemeen als nadeel dat het zijaanzicht niet optimaal is, omdat er dan recht op de zijkant van de plaat gekeken wordt. Daar zijn namelijk alleen de buitenste LED's te zien. Recht van voren ontstaat er dan een soort dode hoek, waar geen diepte te zien is.

Het bovenaanzicht is slechter dan het zijaanzicht, want van boven is alleen de bovenste rij LED's te zien. Dit geeft dus slechts een 2D beeld.

4.1.2 Horizontale balkjes

De horizontale balkjes worden opgebouwd uit diverse horizontale balkjes met LED's (zie figuur 4.2). Deze balkjes zijn één LED hoog en zo breed als de straal van de cirkel. Ze worden niet recht boven elkaar geplaatst, maar steeds iets verschoven, waardoor een soort spiraalvorm ontstaat. De LED's op de balkjes zijn naar boven gericht, waardoor er van alle kanten op de balkjes gekeken wordt.



Figuur 4.2 – LED opstelling: horizontale balkjes
Ondraszek (2007)

Het voordeel van deze opstelling is dat hij erg stabiel is. Dit komt omdat het gewicht van de balkjes over de gehele cirkel verdeeld is. Alleen doordat de balkjes op verschillende hoogten geplaatst worden ontstaat er enige onstabiliteit.

Bij de plaatsing van de LED's moet opgelet worden dat de balkjes op de goede hoogte komen, zodat ze tijdens het roteren geen gat laten ontstaan. Ook mogen de balkjes van LED's in het beeld niet overlappen.

Het bovenaanzicht van deze opstelling is erg goed en het zijaanzicht is ook redelijk. Wanneer er van boven op wordt gekeken zijn alle LED's duidelijk zichtbaar. Als van de zijkant wordt gekeken is alleen de buitenste LED van één balkje nog zichtbaar, terwijl de rest van de balkjes goed zichtbaar blijft. Dit komt omdat de balkjes op verschillende plaatsen en hoogten op de cirkel geplaatst zijn.

4.1.3 Verticale balkjes

De verticale balkjes worden opgebouwd uit diverse verticale balkjes met LED's (zie figuur 4.3). Deze balkjes worden op verschillende afstanden tot de middenas geplaatst. Hierdoor ontstaat de diepte van het scherm.

Figuur 4.3 – LED opstelling: verticale balkjes
Digital Light Sculpture Studio
Foto: van der Leije

Deze opstelling kan problemen in de stabiliteit krijgen. Dit komt doordat de balkjes met LED's op verschillende afstanden van de middenas worden geplaatst. De gewichtsverdeling is daardoor ongelijk over de cirkel. Dit probleem kan echter opgelost worden door onder de cirkel contragewichten te plaatsen.

Er moet, net als bij de verticale balkjes, opgelet worden dat de balkjes op de goede diepte geplaatst worden, zodat deze tijdens het roteren geen gat laten ontstaan, maar ook in het beeld niet overlappen.

Het bovenaanzicht van deze opstelling is redelijk en het zijaanzicht is goed. Als van de bovenkant gekeken wordt, zullen alleen de bovenste LED's zichtbaar zijn (Dit komt omdat de LED's naar buiten gericht zijn). Vanuit het zijaanzicht dekt geen enkele LED de andere af, wat een optimaal beeld oplevert.

4.2 Kleuren in het display

Er zijn twee mogelijke implementaties om verschillende kleuren te krijgen in het 3D LED-display. De eerste mogelijkheid is de meerkleurige LED (RGB LED), waarbij de verschillende kleuren in één behuizing zitten, en de tweede mogelijkheid is het gebruik van meerdere eenkleurige LED's.

Voor de eenkleurige LED's zijn meerdere identieke LED-opstellingen nodig, omdat elk afzonderlijk gekleurde LED op dezelfde plaats in de ruimte moet kunnen komen. Dit levert zichtbeperkingen op. Ook is het moeilijk om meerdere LED's op exact dezelfde plek aan en uit te schakelen, waardoor een heldere weergave lastig wordt.

De RGB LED heeft één LED-opstelling nodig; welke nauwelijks zichtbeperkingen oplevert. Ook wordt een optimaal helder beeld geleverd, omdat de verschillende kleuren op exact dezelfde lokatie aan- en uitgezet worden.

4.3 Aansturing LED's

Er zijn verschillende manieren om LED's aan te sturen. De meest gebruikelijke manier is om ze in een matrix te plaatsen. Met deze methode kan er bijvoorbeeld een 8 bij 4 matrix, wat dus 32 LED's zijn, met 12 (8 + 4) bits geschakeld worden. Zo kunnen dus veel LED's met weinig aansluitingen gestuurd worden. Deze methode heeft als nadeel dat er maar één rij met LED's tegelijk aangestuurd kan worden. Normaal gaat dit snel genoeg om het oog te bedriegen, maar omdat de LED's met een grote snelheid ronddraaien, moeten de LED's erg snel gaan schakelen om een stabiele weergave te maken.

Een tweede manier is om de LED's parallel aan te sturen. Dit betekent voor elke kleur van elke LED dat er één bit nodig is om deze aan te sturen. Dit kost dus veel uitgangen op de microcontroller (MC2). Maar deze manier zorgt er wel voor dat alle LED's op hetzelfde tijdstip schakelen, wat een stabiele weergave tot gevolg heeft.

5 Afweging van de verschillende concepten

De verschillende afwegingen van alle deelconcepten worden in dit hoofdstuk gewogen en gekozen. In § 5.1 wordt als eerste de beste opstelling voor het LED-display gekozen. Daarna wordt in § 5.2 bepaald hoe het beste kleuren in het display geïmplementeerd kunnen worden. En tot slot wordt in § 5.3 beschreven wat de beste oplossing is voor het aansturen van de LED's.

5.1 LED display opstelling

Er zijn drie verschillende opstellingen (zie § 4.2) die afgewogen zijn. De criteria, aan de hand waarvan een keuze gemaakt is, komen voort uit het programma van eisen, namelijk stabiliteit, goed zijaanzicht en een goed bovenaanzicht. De eis dat de opstelling ruimte moet hebben voor 32 LED's, wordt niet meegenomen in de afweging, omdat alle opstellingen hier aan voldoen.

De criteria goed zijaanzicht en bovenaanzicht zijn belangrijker dan het criterium stabiliteit. Dit komt omdat deze niet meer te veranderen zijn, terwijl de stabiliteit eenvoudig gecorrigeerd kan worden door de opstelling uit te balanceren.

Zoals in tabel 5.1 te zien is, scoort de vlakke plaat het beste in stabiliteit, omdat de plaat natuurlijk precies gebalanceerd is over het midden van de as. De horizontale balkjes scoren net iets minder, omdat de balkjes op verschillende hoogten aan de as vast zitten, maar ze zijn wel goed over de oppervlakte van de draaiende cirkel verdeeld. Voor de verticale balkjes geldt dit laatste niet, daarom scoren deze onvoldoende.

De winnaar voor het beste zijaanzicht is de verticale balkjes, omdat geen enkele LED de andere LED's in de weg zit, wanneer er van de zijkant gekeken wordt. Dit geldt een beetje voor de horizontale balkjes, want als er recht op de zijkant van een balkje wordt gekeken, is alleen de voorste LED van dat balkje zichtbaar. Het zijaanzicht van de vlakke plaat is het minst, omdat daar in het midden een kleine dode hoek zit.

Bij het bovenaanzicht zijn de horizontale balkjes het best. De score voor de verticale balkjes en horizontale balkjes wisselen van plaats ten opzichte van het zijaanzicht om precies dezelfde redenering als in de vorige alinea. De vlakke plaat verdient op dit criterium een onvoldoende, omdat alleen de bovenste LED's nog zichtbaar zijn.

Tabel 5.1 – Afweging van de display opstelling

Criterium	Weegfactor	Opstelling		
		Horizontale blakjes	Verticale balkjes	Vlakke plaat
Stabiliteit	1	8	5	**9**
Zijaanzicht	2	8	**9**	7
Bovenaanzicht	2	**9**	8	4
Totaal		**42**	39	31

Legenda : 1 staat voor zeer slecht, 10 voor uitmuntend

Uiteindelijk zijn de horizontale balkjes gekozen, ondanks dat verticale balkjes bijna hetzelfde zicht geven als de horizontale balkjes. Dit komt doordat de stabiliteit van de verticale balkjes veel slechter is.

5.2 Kleuren in het display

De criteria om te kiezen of de RGB LED of meerdere eenkleurige LED's het meest geschikt zijn om kleuren in het display weer te geven, zijn een heldere weergave en het zichtveld van de kijker. Met heldere weergave wordt er bedoeld dat de kleuren niet met andere punten van het raster samen mengen en met het zichtveld wordt bedoeld dat de kleuren weergave geen extra zichtbeperkingen opleveren. De eisen, dat de opstelling 8 verschillende kleuren moet kunnen weergeven en goed zichtbaar zijn bij weinig of geen omgevingslicht, voldoen bij beide methoden. Ook zijn beide methoden snel genoeg om 64 keer per omwenteling aan en uit te schakelen.

De heldere weergave heeft de zwaarste weegfactor, zoals weergegeven in tabel 5.2. Dit is zo omdat er geen kleurmengingen mogen voorkomen en er bij het zichtveld toch altijd zichtbeperkingen zijn, maar dit maakt het niet gelijk onbelangrijk.

De RGB LED wint zowel op de heldere weergave als op het zicht. Dit komt omdat er bij meerdere eenkleurige LED's het moeilijk is om de LED's op exact dezelfde plaats aan of uit te zetten. Ook leveren meer LED's zichtbeperkingen op. De RGB LED is dan ook de grote winnaar.

Tabel 5.2 – Afweging van de kleuren in het display

Criterium	Weegfactor	Opstelling	
		RGB LED	Meerdere eenkleurige LED's
Heldere weergave	3	**8**	5
Zichtveld	2	**8**	4
Totaal		**40**	23

Legenda : 1 staat voor zeer slecht, 10 voor uitmuntend

5.3 Aansturing LED's

Het meest belangrijke criterium bij het aansturen van de LED's is een stabiele weergave. Ook belangrijk zijn het aantal aansluitingen dat nodig is om de LED's aan te sturen. Een slechte stabiele weergave is niet eenvoudig op te lossen, het aantal aansluitingen juist wel. Daarom heeft de stabiele weergave een hogere weegfactor, zoals te zien is in tabel 5.3.

Voor een heldere weergave is de LED's parallel aan te sturen de meest geschikte optie, omdat alle LED's dan tegelijk worden aangestuurd. Als de LED's in een matrix aangestuurd worden is dit niet het geval, omdat de LED's dan alleen rij voor rij op aangestuurd kunnen worden, wat leidt tot een slechte weergave.

Voor het aantal aansluiting is de matrixaansturing beter dan de parallelaansturing. Dit komt omdat er bij de parallelaansturing voor elke kleur van elke LED één bit nodig is.

Uiteindelijk is het dus beter om de LED's parallel aan te sturen. Dit is vooral te danken aan de stabiele weergave, die de parallelaansturing garandeert.

Tabel 5.3 – afweging van de aansturing van de LED's

Criterium	Weegfactor	Aansturing	
		Matrix	Parallel
Stabiele weergave	3	5	**9**
Aansluitingen	1	**8**	4
Totaal		23	**31**

Legenda : 1 staat voor zeer slecht, 10 voor uitmuntend

5.4 Conclusies

Het beste concept voor het draaiende gedeelte van het 3D LED display zijn de roterende horizontale balkjes met RGB LED's, die parallel aangestuurd worden. Dit totale concept biedt een goed zicht voor gebruiker, omdat er met alle inkijkhoeken rekening is gehouden en er een goede methode bedacht is om de verschillende kleuren helder en scherp weer te geven.

Het enige nadeel van dit concept is dat er meerdere uitgangen van de microcontroller nodig zijn om de LED's parallel aan te sturen. Dit is echter geen onoverkomelijk probleem.

6 Implementatietraject

De implementatie van het draaiende gedeelte van de microcontroller wordt in dit hoofdstuk behandeld. Voordat er echt begonnen wordt de implementatie uit te leggen, worden eerst in § 6.1 de algemene afspraken behandeld. Achtereenvolgens worden de deelopdrachten: lichtsluis uitlezen (§ 6.2), LED's aansturen (§ 6.3), Local Interconnect Network (LIN) protocol (§ 6.4) en het verwerken van de ontvangen data (§ 6.5), behandeld. In deze paragrafen wordt woordelijk besproken wat er in de programmacode gebeurt (voor programmacode zie bijlage 2). En als laatste wordt in § 6.6 uitgelegd hoe het mechanisch in elkaar zit.

6.1 Algemene afspraken

Omdat het deelsysteem van dit verslag onderdeel is van het 3D display, zijn er algemene afspraken gemaakt. De afspraken die relevant zijn voor dit verslag betreffen:

- Het datamodel (§ 6.1.1)
- De communicatie (§ 6.1.2)
- De commando's (§ 6.1.3)

6.1.1 Datamodel

Om de LED's op de juiste plaats de goede kleur te laten branden moet er een datamodel afgesproken worden. Er wordt hier gebruik gemaakt van een draadmodel, wat inhoud dat het weer te geven plaatje wordt omgezet in een raster van kleuren. In dit geval beschrijft het raster een aantal cirkels boven elkaar, omdat er horizontale rijen LED's rond gedraaid worden.

Om te beginnen is er gekozen voor:

- 8 kleuren
- Rijen van 8 LED's
- Cirkel van 64 stukken

De 8 verschillende kleuren worden gemaakt door de verschillende kleuren, namelijk rood, groen en blauw (RGB), uit of aan te zetten (zie tabel 6.1). Omdat de rijen uit 8 LED's bestaan, wordt één kleur van een rij LED's doormiddel van 8 bits (1 byte) weergegeven. Waarbij de buitenste LED de laagste bit is en 'aan' een logische '1' is en 'uit' een logische '0'. Elke rij LED's bestaat dan ook uit 3 bytes, waarbij de volgorde rood, groen, blauw (RGB) is.

In het draadmodel worden van onder naar boven de cirkels gezet, waarbij het opbouwen van de cirkel bovenin begint en vervolgens met de klok mee gaat. Op deze manier zijn er 64 stukken van 3 bytes (RGB) nodig om één cirkel te beschrijven. Het totale datapakket bestaat dan ook uit 192 bytes per cirkel.

Tabel 6.1 – kleuren van het display

rood	groen	blauw	kleur
aan	aan	aan	wit
aan	aan	uit	geel
aan	uit	aan	magenta
aan	uit	uit	rood
uit	aan	aan	cyan
uit	aan	uit	groen
uit	uit	aan	blauw
uit	uit	uit	zwart

6.1.2 Communicatie

De data die van de stilstaande microcontroller ontvangen wordt kan twee verschillende dingen zijn, namelijk een plaatje of een commando. Een commando is één afgesproken byte om bepaalde functies, zoals 'ga naar het volgende plaatje', te implementeren. En een plaatje is 192 bytes per cirkel groot (zie §6.1.1). Het plaatje wordt in pakketten van 64 bytes verstuurd, omdat dit handiger in de microcontroller te implementeren is.

De overdracht van data moet gecontroleerd worden om zo niet verkeerde plaatjes en/of commando's te ontvangen. Als er een commando uitgevoerd moet worden, wordt het commando twee keer overgezonden. Dus worden er 2 bytes ontvangen die hetzelfde en een bestaand commando moeten zijn. Als dit allemaal correct is wordt er een 'ok' byte (0xF0) en anders een 'fout' byte (0x0F) teruggezonden.

Bij het ontvangen van een datapakket van een plaatje zijn er meerdere bytes die allemaal correct ontvangen moeten worden. Dit wordt gedaan door alle afzonderlijke bytes bij elkaar op te tellen. Van het opgetelde getal worden alleen de laagste byte als controlebyte gebruikt. Deze controlebyte wordt aan het einde van het datapakket geplakt. Vóór het datapakket moeten twee commando bytes geplaatst worden, die aangeven dat het om een bepaald datapakket gaat. Ook als dit datapakket goed door de controles heen komt, wordt er een 'ok' byte (0xF0) en anders een 'fout' byte (0x0F) teruggezonden.

6.1.3 Commando overzicht

Om het 3D LED display echt te laten werken zijn er ook functies nodig om het display te besturen. Zo moet er naar een ander plaatje gebladerd kunnen worden en moeten er oude plaatjes gewist en nieuwe plaatjes geprogrammeerd kunnen worden. Voor een volledig overzicht van de commando's die deze functies implementeren zie hieronder tabel 6.2.

Tabel 6.2 – Commando overzicht

naam	decimale waarde	omschrijving
stop	1	Laat niks (zwart) zien (het display blijft wel ronddraaien)
start	2	Laat het huidige plaatje zien
volgende	4	Laat het volgende geprogrammeerde plaatje zien
vorige	8	Laat het vorige geprogrammeerde plaatje zien
wissen	16	Wist het huidige plaatje
programmeer	32	Geeft aan dat het huidige plaatje geprogrammeerd gaat worden.
ganaar	64 + X	Laat plaatje X zien (Als X gelijk is aan 0 of op plaats X geen plaatje geprogrammeerd staat, wordt het logo weergegeven)
data	128 + X	Datapakket X voor het te programmeren plaatje

6.2 Lichtsluis uitlezen

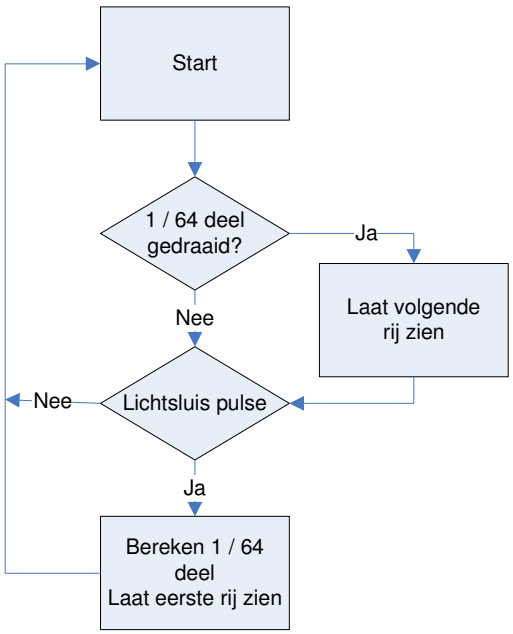
De lichtsluis geeft iedere omwenteling een puls, zodat het systeem kan anticiperen op wisselende snelheden van het draaiende display. De lichtsluis geeft een logische '0', wanneer de zender (infrarood-LED) recht boven de ontvanger (foto transistor) is en anders een logische '1'.

Het probleem bij het uitlezen van de lichtsluis is dat deze spikes[1] vertoont. Om dit op te lossen is er een ontdendering[2] in de software van de microcontroller gebouwd. De ontdendering wordt simpelweg gerealiseerd door, nadat er een puls is ontvangen, even te wachten met opnieuw uitlezen van de lichtsluis.

[1] Een spike is een ongeplande verandering in het signaal
[2] Ontdendering is het uitfilteren van spikes

6.3 LED's aansturen

Het grootste probleem bij het parallel aansturen van de LED's (zie §5.3) is dat er eigenlijk voor elke kleur van elke LED één pin nodig is op de microcontroller. Om dit op te lossen wordt er gebruik gemaakt van '8-bit shift registers' [3]. Zo worden de bits één voor één in de shift registers geladen en daarna worden alle shift registers weergegeven op het display via één pin.



Figuur 6.1 – flowchart van het aansturen van de LED's

Het systeem moet kunnen anticiperen op verschillende draaisnelheden. Daarom wordt elk rondje opnieuw berekend wanneer de LED's aan- en uitgezet moeten worden (zie figuur 6.1). Nadat het display één 64ste deel gedraaid heeft, worden de LED's opnieuw met de juiste waarden van plaatje bijgewerkt. Dit blijft zich herhalen tot er dan een puls van de lichtsluis gedetecteerd wordt. Als dit gebeurt wordt opnieuw het 64ste deel berekend en de eerste rij van het plaatje op de LED's weergegeven.

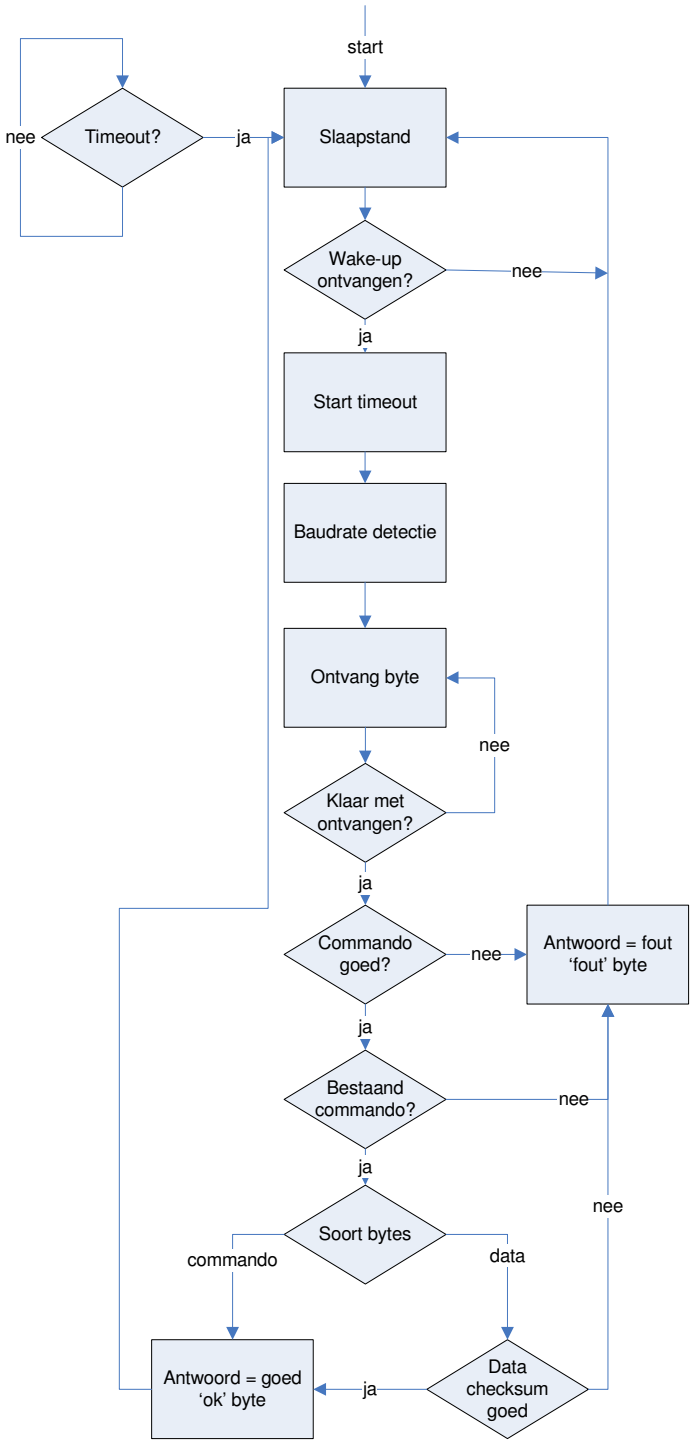
[3] 8-bit shift register is een seriëel-parallelomzetter van 1 bit naar 8 bit

6.4 Local Interconnect Network (LIN)

De draaiende microcontroller communiceert met de stilstaande microcontroller via de Local Interconnect Network (LIN) hardware. Dit protocol staat bekend om zijn robuuste communicatie over bijvoorbeeld sleepcontacten (die ook in het product gebruikt worden, zie §6.6.3). De LIN-software maakt gebruik van de UART (universal asynchronous receiver / transmitter)–module op de microcontroller (Microchip 2007: 237).

Om de communicatie tussen de microcontrollers tot een goed einde te brengen, wordt er iedere keer een serie stappen doorlopen. De eerste stap is het ontvangen van het wake-up signaal(wakker worden). Hiermee gaat de draaiende microcontroller (MC2) uit zijn slaapstand. Vervolgens wordt er een time-out[4] detectie aangezet om eventuele fouten in de communicatie te onderscheppen. Het eerst volgende wat hierna gebeurt is het detecteren de baudrate[5], die verkregen wordt van de stilstaande microcontroller (MC1). Zodra deze detectie is voltooid kunnen er bytes worden ontvangen.

Na het ontvangen van de bytes worden er een twee controles uitgevoerd. Eerst wordt er gecontroleerd of het commando goed is aangekomen, gevolgd door een check of het commando überhaupt bestaat. Mocht het bij één van deze controles fout gaan dan wordt er 'fout' byte gestuurd naar MC1.



Figuur 6.2 - flowchart van de communicatie tussen de microcontrollers

[4] Een time-out is een fout die optreedt zodra er te lang niks wordt ontvangen
[5] De baudrate staat gelijk aan het aantal symbolen dat per seconde verzonden kunnen worden

Als deze controles voltooid zijn, wordt er bepaald wat voor bytes er ontvangen zijn. Is er een commando ontvangen, dan wordt de 'ok' byte verstuurd en gaat de UART-module in slaapstand. Is er echter data ontvangen, dan wordt eerst gecontroleerd of de controlebyte klopt.

Vanaf het starten met het ontvangen van de bytes tot het terugsturen van een antwoord, wordt er constant gecontroleerd of er een time-out optreedt. Mocht dit gebeuren dan wordt de UART-module weer in de slaapstand gezet.

6.5 Verwerken ontvangen data

De ontvangen bytes dienen elk op een andere manier door de microcontroller verwerkt te worden. Er zijn de commando bytes: start, stop, volgende, vorige, wissen, programmeer, ganaar en data. Deze bedienen diverse functies(zie §6.1.3) op de microcontroller. Ook zijn er nog de databytes, deze bevatten informatie over de opbouw van een plaatje op het display.

Het start- en stop-commando

Het start- en stopcommando zorgen ervoor dat MC2 respectievelijk wel en niet in de display module[6] komen. Het start-commando zorgt ervoor dat MC2 in zijn display module terecht komt. Het stop-commando daarentegen zorgt er juist voor dat MC2 niet in deze module komt.

Het volgende- en vorige-commando

Om het volgende- en vorige-commando te implementeren is het nodig dat het aantal geprogrammeerde plaatjes, maar ook het nummer van het huidige plaatje, in het geheugen staan. De geheugentabel voor deze registratie bestaat momenteel uit 11 bytes. De eerste byte houdt bij welk plaatje wordt weergegeven, terwijl de overige 10 bytes staan voor de aanwezigheid van plaatjes 1 tot 10.

De twee commando's lezen dit geheugen en voeren hier twee controles op uit zodra ze worden aangeroepen. Als eerste wordt er gecontroleerd of het volgende(of vorige) plaatje wel bestaat. Mocht dit niet het geval zijn, dan wordt naar het eerstvolgende plaatje gezocht. Een voorbeeld hiervan: stel het display laat plaatje 8 zien, zodra op vorige wordt gedrukt wil het display plaatje 7 laten zien. In het geval dat dit plaatje er niet is gaat hij door naar plaatje nummer 6, daarna naar nummer 5, enzovoorts. De tweede controle kijkt of er wel naar het volgende of vorige plaatje gesprongen kan worden. Als het display bijvoorbeeld op plaatje 10 staat en het het volgende-commando wordt gebruikt, dan dient hij naar plaatje 1 te springen. Voor het vorige-commando is dit het geval als er van 1 naar 10 gegaan wordt. Mocht er helemaal geen plaatje in het geheugen staan, dan wordt het 3D LED-display logo weergegeven. Dit logo staat op een gereserveerde locatie in het geheugen om overschrijven te voorkomen.

[6] In de display module wordt het geselecteerde plaatje op het display weergegeven

Het ganaar-commando

Het ganaar-commando zorgt ervoor dat een specifiek plaatje getoond wordt, door direct de eerste byte van de geheugentabel (zie ‘Het volgende- en vorige-commando’) te veranderen. Zodra het commando gegeven is, wordt het nummer van het gewenste plaatje in de eerste byte van de geheugentabel weggeschreven. Als op de gewenste positie geen plaatje staat zal het 3D LED-display logo worden weergegeven.

Het data-commando en databytes

Het data commando kondigt de komst van een datapakket(data bytes) aan en treft voorbereidingen om dit op te slaan in het geheugen. Voordat een datapakket kan worden opgeslagen, moet de gewenste locatie eerst gewist worden(dit is vereist voor het flash geheugen). Hierna wordt het datapakket, op de locatie die wordt aangewezen, opgeslagen. Als dit niet het eerste datapakket is, dan wordt deze een stuk verder(64 bytes) in het geheugen opgeslagen. Ook wordt de geheugentabel bijgewerkt, zodat deze weet dat er een nieuw plaatje is weggeschreven.

Overige commando’s

Het wissen-commando wist het huidige plaatje dat op het display wordt weergegeven. Ook wordt de geheugentabel (zie ‘Het volgende- en vorige-commando’) gewijzigd, zodat het plaatje niet meer kan worden weergegeven.

6.6 Mechanische behuizing

Het huidige prototype, dat gebruikt wordt om software en hardware te testen, bestaat uit 5 hoofdonderdelen, de bevestiging van de motor (§6.6.1), de behuizing en montage van de microcontroller en de LIN-hardware (§6.6.2), het sleep contact (§6.6.3), stabilisatie van de as (§6.6.4) en de behuizing van de LED's (§6.6.5).

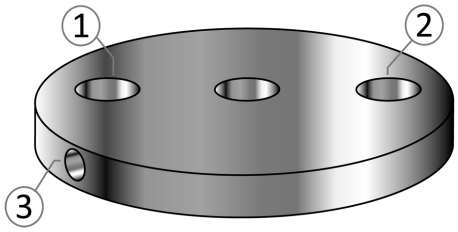
6.6.1 De bevestiging van de motor

Om de motor voldoende stevig te bevestigen is voor een combinatie behuizing van staal, hout en plastic gekozen. De motor is bevestigd aan een stalen vakwerk constructie (gerealiseerd met Mecano™), die op zijn beurt op een houten plaat is gemonteerd. Deze houten plaat is, met oog op veiligheid, op zijn beurt gemonteerd in een plastic bak.

6.6.2 De behuizing en montage van de microcontroller en de LIN-hardware

De microcontroller en de LIN-hardware zijn met behulp van een Lego™-constructie en een metalen bus op de as van de motor bevestigd.

De metalen bus bestaat uit een cirkelvormig stuk staal met vier gaten erin. In figuur 6.3 is dit schematisch weergegeven. De gaten gemarkeerd 1 en 2 dienen voor het bevestigen van de Lego behuizing. Het gat met nummer 3 is aanwezig om de metalen bus op de as van de motor vast te schroeven.



Figuur 6.3 - metalen bus voor montage Lego

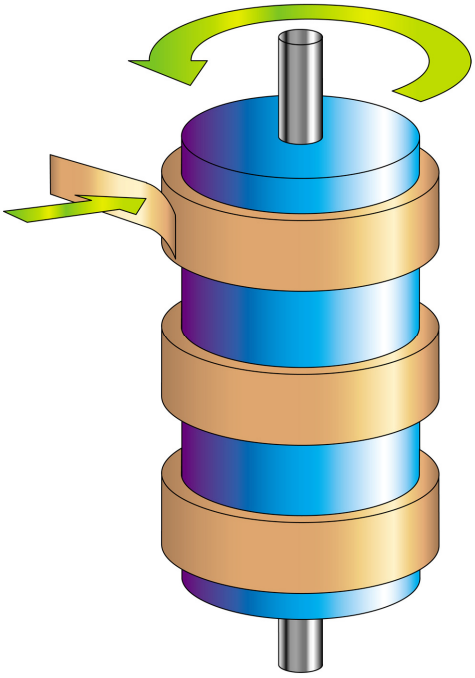
De Lego constructie is zo ontworpen dat het microcontrollerbordje en de LIN-hardware stevig, maar voldoende compact, naast elkaar liggen. Bij de plaatsing van deze 2 onderdelen is er ook voor gezorgd dat het zwaartepunt van de Lego constructie precies in het midden ligt.

6.6.3 Het sleepcontact

Het sleepcontact (zie figuur 6.4) is gerealiseerd door koperen ringen en verende contacten om zo een stabiele overdracht van voeding en data naar het roterende deel te garanderen.

De koperen ringen, in totaal drie stuks (twee voor de voeding en één voor de data), worden om een doorboorde kurk geschoven. Deze kurk wordt dan met een as aan de overige delen bevestigd.

De verende contacten bestaan uit een hefboom, die met elastiekjes op de draaiende koperen ringen wordt geduwd.



Figuur 6.4 - sleepcontact

6.6.4 Stabilisatie van de as

Door de diverse onderdelen die aan de as hangen, is er een extra constructie nodig om deze te stabiliseren. Aan de bovenkant van de plastic bak zijn kruiselings 2 houten balken bevestigd. In het midden, waar de balken elkaar kruisen, is een gat voor de as geboord. Op deze balken wordt uiteindelijk nog een kooiconstructie van Lego geplaatst, om de as extra te stabiliseren.

6.6.5 De behuizing van de LED's

De behuizing van de LED's bestaat uit een Lego-constructie waar de as doorheen gaat. Om de veiligheid van het display te kunnen garanderen, zijn minimale omvang en voldoende stevigheid bij deze constructie een vereiste.

7 Conclusies en aanbevelingen

De implementatie van het roterende gedeelte van het 3D LED-display is op vier punten tot stand gekomen:

- Het LED-display is opgebouwd uit horizontaal geplaatste balkjes van 8 LED's. De LED's kunnen acht verschillende kleuren weergeven door rood, groen of blauw aan of uit te zetten (RGB LED's). Deze LED's zijn parallel aangestuurd door middel van shift registers.
- De dataoverdracht is gerealiseerd met LIN(Local Interconnect Network)-hardware en vindt plaats via een sleepcontact over één draad. Bij deze LIN-hardware is een eigen protocol met errordetectie gemaakt.
- Een plaatje is opgebouwd uit een raster van kleuren, waarbij elke laag in 64 stukken is verdeeld. Voor elk stuk zijn drie bytes (RGB) nodig.
- Een lichtsluis die gemaakt is om het display adaptief aan de snelheid te maken.

De belangrijkste aanbeveling voor toekomstige ontwikkelaars is het maken van betere mechanische constructie. Een betere mechanische constructie geeft namelijk meer garantie voor een stabiele overdracht en stabiele weergave. Met de gemaakte opstelling is het namelijk niet mogelijk om de opstelling met 25 Hz goed te laten functioneren, doordat de sleepcontacten te vaak los komen. Ook de weergave van het 3D beeld is niet optimaal, doordat de mechanische constructie de 25 Hz niet haalt (om goed te werken).

Een andere aanbeveling is het ontwikkelen van PCB's (Printed Circuit Boards) voor alle elektronische hardware. Door de mechanische trillingen die optreden is het hierbij verstandig om voor de diverse IC's geen voetjes te gebruiken.

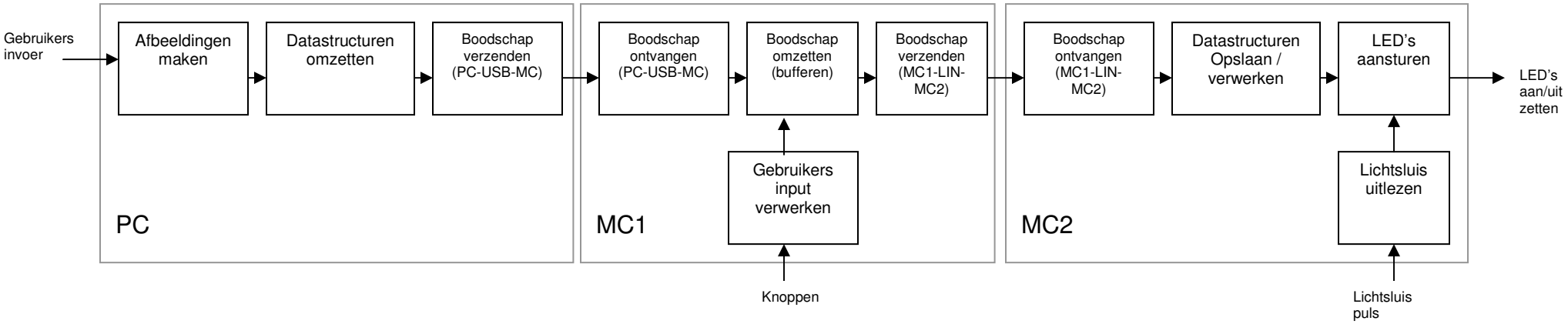
De laatste aanbevelingen zijn de microcontroller uitbreiden met meer geheugen en een plaatje uit meer dan 64 stukken op te bouwen om zo een hogere resolutie te krijgen.

Literatuurlijst

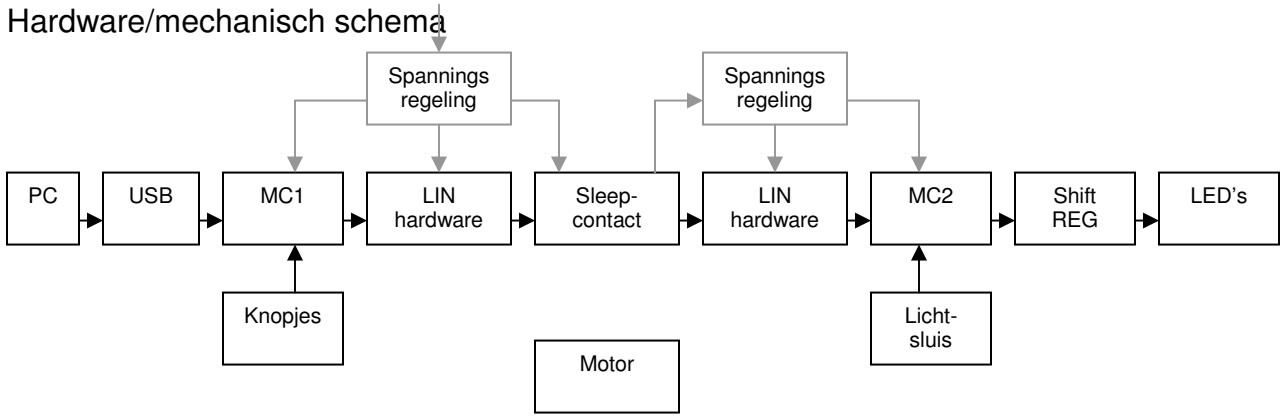
Ondraszek (2007)
http://www.ondraszek.ds.polsl.gliwice.pl/~looser/dyplom/kl01.png
Geraadpleegd 11 mei 2007

Microchip Technology Inc. (2007). *PIC18F2455/2550/4455/4550 Data Sheet*

Bijlage 1: Functioneel Blokschema



Hardware/mechanisch schema



Bijlage 2: Programmacode

```
#include <p18f4550.h> //Laden van de benodigde bibliotheken
#include <delays.h> //Bib. met standard delay-functies

#define stop          1
#define start         2
#define volgende      4
#define vorige        8
#define wissen        16
#define programmeer   32
#define ganaar        64      //ganaar_X = 128+X, waarbij X het plaatjenummer is
#define data          128 //data_X = 64+X, waarbij X het datanummer is
#define       padress     0x005000
#define       ladress     0x007E40
#define p_aanwadress 0x007FC0

extern void _startup (void);
void mcp201(void);
void pim(void);
void rgb(char, char, char, char, char, char);
void startupPIM(void);
void starttimeout(void);
void timeout(void);
void stoptimeout(void);
void zendantwoord(void);
void paanw_write(char[11]);
unsigned char flash_readbyte(long);
void flash_erasebuffer(long);
void flash_writebuffer(char[64], long);

void functies(void);
void programmeren(void);
void LEDs(void);
void Logo(void);

#pragma code _RESET_INTERRUPT_VECTOR = 0x000800
void _reset (void)
{
     _asm goto startupPIM _endasm;   //ga naar startupPIM zodra je reset krijgt
}
#pragma code
#pragma code _HIGH_INTERRUPT_VECTOR = 0x000808
void _high_ISR (void)
{
     _asm goto pim _endasm;               //interrupt wanneer byte onvtangen wordt
}
#pragma code
#pragma code _LOW_INTERRUPT_VECTOR = 0x000818
void _low_ISR (void)
{
     _asm goto timeout _endasm;           //interrupt wanneer er een timeout optreed
}
#pragma code

void startupPIM(void)
{
     rgb((char) 0,(char) 0,(char) 0xAA, (char) 0,(char) 0,(char) 0xAA);
     _asm goto _startup _endasm;
}

// 3D LED LOGO
```

```
rom char lbuf1[64] =
{0x00,0x00,0x40,0x00,0x00,0x40,0x00,0x00,0x40,0x00,0x00,0x40,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x49,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x41,0x00,0x00,0x00,0x41,0x00,0x00,0x00,0x63,0x00,0x00,0x3E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x09,0x00,0x00,0x09};
rom char lbuf2[64] =
{0x00,0x00,0x79,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00,0x00,0x78,0x00,0x00,0x00,0x48,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00,0x00,0x4F,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x79,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00};
rom char lbuf3[64] =
{0x00,0x3E,0x00,0x00,0x00,0x41,0x00,0x00,0x00,0x41,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x49,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x41,0x00,0x00,0x00,0x41,0x00,0x00,0x00,0x63,0x00,0x00,0x00,0x3E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,0x00,0x08,0x00,0x00,0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F};

char teller = 1;
char tellerbuf = 0;
char buf[64];                   //initialisatie variabelen
char antwoord;
char functie = 0;
char ontvangen = 0;
char checksumdata = 0;
int gr_pl = 2*192;
char bezig = 1;
char p_aanw_oud = 0;
int byte = 0;
int byte2 = 0;
long adressw = 0;
long adress = 0;
char p_aanw[11];
long tellerL = 0;
long deeltellerL = 0;
long deelL = 0;

void main(void)
{
    int i = 0;
    INTCON = 0b00000000;
    for (i=0; i<11; i++)                //Initialisatie van p_aanw
    {
        p_aanw[i] = flash_readbyte(p_aanwadress+i);
    }

    if (p_aanw[0]!=(char) 255)
    {
        adress = ((p_aanw[0]-1)*gr_pl)+padress; //begin bij het goede plaatje
    }
    else
    {
        adress = ladress;               //geen plaatjes = logo laten zien
    }

    mcp201();                                   //LIN hardware init

    TRISA=0;
    PORTAbits.RA0 = 0;                          //zet lampje uit

    TRISC=0b10000011;//RX en TX poorten aan.
    RCSTA=0b10010000;// Serial port enabled Enable Reciever
    TXSTA = 0b00100110;//Transmit Enable, Hoge Baudrate select ,Transmit Shift Register Status is leeg
    BAUDCON=0b00001000;             //16-bit baudrate(BRG16)


    RCONbits.IPEN = 1;              //Enable priority levels on interrupts
```

```
    RCONbits.SBOREN = 0;    //Disable BOR

    IPR1 = 0b00100000;          //Ontvangen is high interrupt(RCIP)
    PIE1 = 0b00100001;      //Ontvang interrupt is aan(RCIE)

    T1CON = 0b11110000; //16-bit,interne clock Fosc/4, 2-bits prescaler 1:1,
                        //oscillator on, niet synchroniseren, interne clock, aan

    SPBRGH = 0x02;              //baudrate register
    SPBRG = 0x70;

    Delay10KTCYx(200); //Wacht 2000x10000 clockcycles

    BAUDCONbits.WUE = 1;    //sleep mode comm.


    Logo();                     //programeer logo

    while(1)
    {
        INTCON = 0b00000000;    //interrupts uit

        functies();                 //voert ontvangen commando's uit
        LEDs();                     //stuurt LEDs aan
        programmeren();             //programmeert plaatje

        INTCON = 0b11000000;    //interrupts aan
    }
}

void pim(void)
{
    char in;
    in = RCREG;                         //ontvangbyte

    rgb((char) teller,(char) 0,(char) 0, (char) teller,(char) 0,(char) 0);
    //laat aantal ontvangen bytes zien

    buf[tellerbuf] = in;        //schrijf byte in buffer

    starttimeout();             //zet timeout aan
    if (teller==1)              //initialiseer communicatie
    {
        tellerbuf = 0;
        PORTAbits.RA0 = 0;
        while(BAUDCONbits.WUE == 1) {}; //wacht tot MC wakker is
        while (BAUDCONbits.ABDEN == 1 && (!(PIR1bits.TMR1IF)))
        //wacht tot de auto baud rate detectie klaar is
        {}
    }
    else
    {
        if (tellerbuf == 64)    //65ste byte is een checksum
        {
            checksumdata = in;
        }

        tellerbuf++;

        if (teller == 5)        //header pakketje is ontvangen
        {
            if      (buf[2]==buf[3]) //check of het een commando is
            {
```

```
                    functie = buf[3];
                    if (functie == stop || functie == start || functie == volgende || functie == vorige
|| functie == programmeer || functie == wissen || (functie & ganaar) || (functie & data))
//als een goedde functie is aangekomen
                    {
                        antwoord = 0xF0;       //antwoord = ok
                        ontvangen = 1;         //voer functie uit
                    }
                    else
                    {
                        antwoord = 0x0F;       //antwoord = fout
                        ontvangen = 0;         //voer niks uit
                        functie = 0;
                    }
                }
                else
                {
                    antwoord = 0x0F;           //antwoord = fout
                    ontvangen = 0;             //voer niks uit
                    functie = 0;
                }

                if (functie & data)            //als de functie data is
                {
                    ontvangen = 0;         //voer niks uit en wacht op verdere data
                    tellerbuf = 0;
                }
                else
                {
                    rgb((char) functie,(char) 0,(char) antwoord, (char) functie,(char) 0,(char)
antwoord);  //laat functie en antwoord zien

                    zendantwoord();            //stuur antwoord
                    teller = 0;
                    BAUDCONbits.WUE = 1;    // ga slapen
                    stoptimeout();             //stop timeout
                }
            }
        }

        teller++;                          //tel je ontv. bytes

        INTCONbits.GIE = 1;                //zet global interrupts aan
}


void rgb(char r, char g, char b, char r2, char g2, char b2)
//functie om 2 balkje RGB LEDjes aan te sturen.
{
    int i;
    TRISD=0x00;     //Maak alle pinnen van poort D output
    for (i=0; i<8 ;i++)
    {
        PORTD = 0;
        //schuif 1 bit in het shift register (rood)
        PORTDbits.RD5 = (r>>i) & 0x01;
        PORTDbits.RD3 = (g>>i) & 0x01;
        PORTDbits.RD7 = (b>>i) & 0x01;
        //schuif 1 bit in het shift register (rood rij 2)
        PORTDbits.RD1 = (r2>>i) & 0x01;
        PORTDbits.RD6 = (g2>>i) & 0x01;
        PORTDbits.RD0 = (b2>>i) & 0x01;
```

```
            PORTDbits.RD4 = 1;
      }
      PORTDbits.RD2 = 1;            //laat de ledjes branden
}

void mcp201(void)
{
      unsigned int mydelay = 20000;
      TRISC=0b10000011;       //init serial pins - TX and control
      PORTCbits.RC2=0;    //chip naar goeie status
      while(mydelay--);
      mydelay = 20000;
      PORTCbits.RC2=1;       //chip naar goeie status
      while(mydelay--);
}

void timeout(void)
{
      PORTAbits.RA0 = 1;
      rgb((char) 0, (char) 255, (char) 0, (char) 0, (char) 255, (char) 0);

      T1CONbits.TMR1ON = 0;   //timer uit
      PIR1bits.TMR1IF = 0;    //reset interrupt falg /overflow

      tellerbuf = 0;                //begin opnieuw met tellen data
      teller = 1;
      BAUDCONbits.WUE = 1;    //sleep mode
}

void starttimeout(void)
{
      T1CONbits.TMR1ON = 0;         //timer uit
      TMR1H = 0x6D;                    //timer instellen 25 ms
      TMR1L = 0x84;
      INTCONbits.PEIE = 1;          //interrupts aan
      T1CONbits.TMR1ON = 1;         //timer aan
}

void stoptimeout(void)
{
      T1CONbits.TMR1ON = 0;         //timer uit
}

void zendantwoord(void)
{
      RCSTAbits.CREN = 0;                 //recieve uit
      TXSTAbits.TXEN = 1;                 //transmit aan

      TXREG = antwoord;                   //stuur antwoord
      while(!TXSTAbits.TRMT);       //wacht tot antwoord verstuurt is

      TXSTAbits.TXEN = 0;                 //transmit uit
      RCSTAbits.CREN = 1;                 //recieve aan
}

void paanw_write(char p_aanw[11])      //schrijft PAANW
{
      int i;
      char buf[64];
      for (i=11; i<64; i++)
      {
            buf[i] = 0xFF;
      }
```

```
    for (i=0; i<11; i++)
    {
        buf[i] = p_aanw[i];
    }

    flash_writebuffer(buf, p_aanwadress);
}

void functies(void)          //voer de functies uit
{
    char ganr = 0;

    if (ontvangen==1)
    {
        switch (functie)
        {
            case stop:          //leds uit
                bezig =0;
                break;
            case start:         //leds aan
                bezig =1;
                break;
            case volgende:              //ga naar volgende plaatjes
                p_aanw_oud = p_aanw[0];
                if (p_aanw[0]!=(char) 255)     //als er een plaatje in staat
                {
                    p_aanw[0]++;                //ga naar volgende
                    if (p_aanw[0]>(char) 10)//bij plaatje 10 ga naar 1
                    {
                        p_aanw[0]=1;
                    }
                    while (p_aanw[p_aanw[0]]==(char) 255 && p_aanw_oud != p_aanw[0])
                    //sla lege plekken over
                    {
                        p_aanw[0]++;
                        if (p_aanw[0]==(char) 11)
                        {
                            p_aanw[0]=1;
                        }
                    }

                    if (p_aanw[p_aanw[0]]!=(char) 255)     //laat plaatje zien
                    {
                        adress = ((p_aanw[0]-1)*gr_pl)+padress; //bereken adres plaatje
                        paanw_write(p_aanw);                    //schrijf nieuw paanw weg
                    }
                    else
                    {
                        adress = ladress;           // geen plaatje = logo
                    }

                }
                else
                {
                    adress = ladress;       // geen plaatje = logo
                }
                break;
            case vorige:          //laat vorig plaatje zien  werkt zelfde als volgende
                p_aanw_oud = p_aanw[0];
                if (p_aanw[0]!=(char) 255)
                {
                    if (p_aanw[0]<(char) 2)
```

```
            {
                  p_aanw[0]=11;
            }
            p_aanw[0]--;

            while (p_aanw[p_aanw[0]]==(char) 255 && p_aanw_oud != p_aanw[0])
            {
                  if (p_aanw[0]==(char) 1)
                  {
                        p_aanw[0]=11;
                  }
                  p_aanw[0]--;
            }

            if (p_aanw[p_aanw[0]]!=(char) 255)
            {
                  adress = ((p_aanw[0]-1)*gr_pl)+padress;
                  paanw_write(p_aanw);
            }
            else
            {
                  adress = ladress;
            }
      }
      else
      {
            adress = ladress;
      }
      break;
case wissen:                              //wis geheugen locatie
      adress = (p_aanw[0]-1)*gr_pl+padress;   //bereken adres

      flash_erasebuffer(adress);
      flash_erasebuffer(adress+64);
      flash_erasebuffer(adress+128);
      flash_erasebuffer(adress+192);
      flash_erasebuffer(adress+256);
      flash_erasebuffer(adress+320);

      p_aanw[p_aanw[0]]=0xFF;        //wis plaatje uit paanw

      paanw_write(p_aanw);

      adress = ladress;              //laat logo zien

      break;
default:
      if (functie & ganaar)   //als ganaar commando is gegeven
      {
            ganr = functie & 0b00111111;//bereken offset

            if (ganr != 0)          //als het geen logo is
            {
                  p_aanw[0] = ganr;
                  paanw_write(p_aanw);
            }

            if (p_aanw[ganr]!=(char) 255 && ganr != 0)
            //laat plaatje zien
            {
                  adress = ((p_aanw[0]-1)*gr_pl)+padress;
            }
            else
            {
                  adress = ladress;
```

```
                        }
                    }
                }

                ontvangen = 0;          //klaar met uitvoeren functie
            }
        }

        void programmeren(void)
        {
            char checksum = 0;
            char ganr=0;
            int i;

            if(tellerbuf==65)
            //als je 64 byte data en 1 byte checksum hebt ontvangen
            {
                tellerbuf++;
                checksum = 0;
                for(i = 0; i <64; i++)                  //bereken checksum
                {
                    checksum = checksum+buf[i];
                }

                if (checksum == checksumdata)           //Bij goede data
                {
                    ganr = functie & 0b01111111;    //bereken adres
                    adress = (p_aanw[0]-1)*gr_pl+padress;
                    adress = adress+((int) ganr-1)*64;
                    flash_writebuffer(buf, adress); //schrijf het weg
                    antwoord = 0xF0;                //antwoord = ok
                }
                else
                {
                    antwoord = 0x0F;        //antwoord = fout
                }

                rgb((char) 0,(char) 0, (char) antwoord, (char) 0,(char) 0,(char) antwoord);

                stoptimeout();

                zendantwoord();         //verzend antwoord

                if (ganr == 6 && checksum == checksumdata)
                //zodra 6 pakketen zijn ontvange = 1 plaatje en cehcksum is goed
                {
                    p_aanw[p_aanw[0]]=0x00;         //update paanw
                    paanw_write(p_aanw);
                }

                teller = 1;                     //wacht op nieuw bericht
                BAUDCONbits.WUE = 1;    //sleep
            }
        }

        void LEDs(void)
        {
            char r = 0;                 //init kleuren
            char g = 255;
            char b = 255;
            char r2 = 0;
            char g2 = 255;
            char b2 = 255;
```

```
    if (bezig==1)   //laat plaatje zien
    {
        if (tellerL>deeltellerL)        //als je 1/64ste hebt gedraaid
        {
            deeltellerL = deelL+deeltellerL;        //bereken volgende 1/64ste deel
            if (byte>=0)                                    //blijf eigen plaatje
            {
                r = flash_readbyte(adressw+byte);       //lees plaatje
                g = flash_readbyte(adressw+byte+1);
                b = flash_readbyte(adressw+byte+2);

                r2 = flash_readbyte(adressw+byte2);
                g2 = flash_readbyte(adressw+byte2+1);
                b2 = flash_readbyte(adressw+byte2+2);

                rgb(r, g, b, r2, g2, b2);//stuurt rij ledjes aan
                byte = byte - 3;              //bereken volgende bytes om te laten zien
                byte2 = byte2 - 3;

                if(byte2==189)
                //2e rij leds is 180 graden gedraaid ten opzichte van 1ste rij
                {
                    byte2=381;
                }
            }
        }

        if (PORTBbits.RB0 == 0 && tellerL > 1000)       // rondje gedraaid & anti spike  lichtsluis
        {
            adressw =adress;
            deelL = tellerL>>6;           //bereken 1/64ste deel
            deeltellerL = 0;
            byte = 189;                 //reset byte tellers
            byte2 = 192+93;
            tellerL = 0;
        }
        tellerL++;            //telt rondje

    }
}

void Logo(void)             //schrijf logo weg op ladress
{
    int i;

    for (i=0; i<64; i++)
    {
        buf[i] = 0;
    }
    flash_writebuffer(buf, ladress);
    flash_writebuffer(buf, ladress+64);
    flash_writebuffer(buf, ladress+128);


    for (i=0; i<64; i++)
    {
        buf[i] = lbuf1[i];
    }
    flash_writebuffer(buf, ladress+192);

    for (i=0; i<64; i++)
    {
        buf[i] = lbuf2[i];
    }
```

```
        flash_writebuffer(buf, ladress+256);

        for (i=0; i<64; i++)
        {
                buf[i] = lbuf3[i];
        }
        flash_writebuffer(buf, ladress+320);
}
```