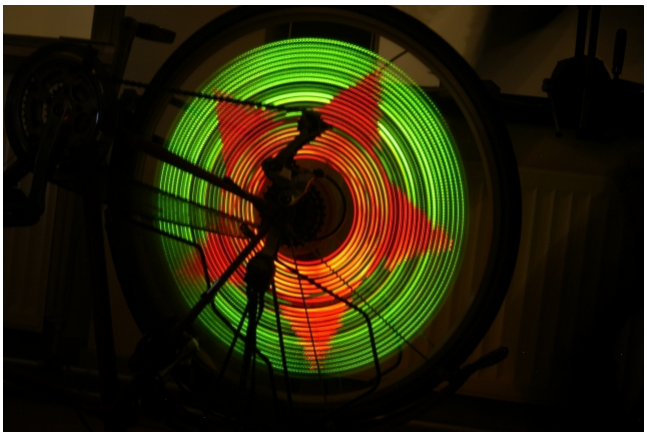
Profielwerkstuk ‘PimpMyBike’

Door Jan Paul Posma, klas 6b, Willem Lodewijk Gymnasium te Groningen, 2007/2008

Inleiding

Idee

Wat is precies het doel van mijn profielwerkstuk? Ik probeer een beeld binnenin een fietswiel te creëren. Dat zou je op een aantal manieren kunnen doen. Ten eerste zou je van buitenaf door middel van projectie een beeld op een stuk karton of iets dergelijks in het fietswiel kunnen vormen. Dit is vrij simpel: je plakt een rond stuk karton tussen de spaken en je richt er een beamer op. Helaas kan dit alleen als de fiets stilstaat, tenzij je met een beamer wil rondfietsen. Ook is dit nogal duur.
Beter is om gebruik te maken van een trucje dat onze hersenen uithalen. Wanneer we losse beelden maar snel genoeg zien, combineren we deze beelden tot een geheel. In het Engels heet dit ‘persistence of view’ (POV, niet te verwarren met ‘point of view’). Je kunt het vergelijken met een lange sluitertijd gebruiken op een fototoestel. Ook de televisie maakt gebruik van een soortgelijk effect: door snel beelden te tonen versmelten ze in elkaar en lijkt het alsof je beweging ziet.
Op dit principe baseer ik ook mijn profielwerkstuk. Als je binnenin het fietswiel lampjes plaatst die je heel snel van kleur en intensiteit laat afwisselen, en daarbij het fietswiel maar snel genoeg ronddraait, zullen de hersenen van een waarnemer deze losse lampjes, die in een rij zitten, van binnen naar buiten, zien als een beeld dat niet verandert. Je hoeft dus niet het hele fietswiel volplakken met lampjes, immers slechts een rij is voldoende omdat de hersenen deze lampjes combineren tot een beeld, wanneer het wiel maar snel genoeg draait.
Om de lampjes zo te laten ‘knipperen’ dat er daadwerkelijk een beeld ontstaat is er wel vrij geavanceerde elektronica nodig. Een microcontroller voor de aansturing is bijvoorbeeld noodzakelijk. Ook moet er op een of andere manier geheugen aanwezig zijn om het te weergeven beeld in op te slaan. Bovendien moet er rekening mee worden gehouden dat de snelheid van het wiel niet constant is. De snelheid waarmee de lampjes moeten knipperen is hiervan wel afhankelijk, dus moet de snelheid van het wiel gemeten worden door de microcontroller. Voor de hand ligt om een magneetsensor te gebruiken, waarbij een magneet op het frame van de fiets wordt geplaatst.



Uitvoering

In dit profielwerkstuk beschrijf ik hoe ik dit project heb gerealiseerd. Ik heb hierbij enige ondersteuning gehad door mijn natuurkundeleraar, dhr. Falkena, en een medewerker van de Jonge Onderzoekers Groningen, namelijk dhr. Schreuder. Vrijwel alles was echter op eigen kracht te doen. Met name bij de Java-software om een plaatje om te zetten in een voor de microcontroller te lezen formaat ben ik ondersteund door dhr. Schreuder. Mijn dank gaat bovendien uit naar dhr. Woelders, die vanuit China zeer goedkoop onderdelen aanbiedt op www.samenkopen.net en een voordelige printplaat-service biedt op www.makepcb.com. Daarbij heb ik ook veel onderdelen bij www.tme.pl besteld, een Poolse webshop met een zeer groot assortiment en ook lage prijzen.

Ik heb dit project ‘PimpMyBike’ genoemd. Alles wat ik in die profielwerkstuk beschrijf is ‘Prototype 1’ van PimpMyBike. Wellicht breng ik in de toekomst verbeteringen aan. Dit profielwerkstuk heeft daar uiteraard geen betrekking op. Alles in dit profielwerkstuk mag worden gebruikt volgens de GNU Public License versie 3 of nieuwer, behalve de PCB layout. Deze is ook slechts in lage resolutie bijgevoegd. Dit is vanwege eventuele toekomstige verkoop van dit product. Zie bijlage A voor de volledige licentieovereenkomst.

Gebruikte hardware

Selectie van onderdelen

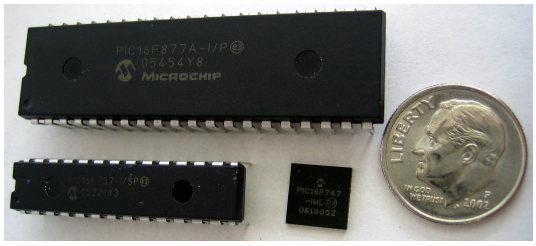
Voor de microcontroller heb ik gekozen voor de PIC18F4550 van Microchip. Ik heb vaker microcontrollers van deze serie gebruikt, omdat bij Microchip gratis proefmodellen (samples) zijn aan te vragen. De PIC18F4550 is bovendien een erg mooi model omdat er ingebouwd USB op zit en met een vrij hoge klokfrequentie (48 Mhz) kan werken. Ook heeft dit model maar liefst 32 bruikbare invoer-uitvoer aansluitingen. Dit is de datasheet: http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf

De voedingsschakeling heb ik gebaseerd op de TC105 van Microchip. Ik had voorheen geen ervaring hiermee, en dit leek wel een aardige keuze omdat deze chip een grote variëteit aan input-spanningen kan omzetten (van 2,2 tot 10 volt) en ook een vrij groot vermogen kan leveren (1 A). De datasheet is hier te vinden: http://ww1.microchip.com/downloads/cn/DeviceDoc/cn_21349b.pdf

De magneetsensor en de led's heb ik besteld bij dhr. Woelders. De magneetsensor in een standaard reedcontact waarvan het niet de bedoeling is dat er hoge stromen doorheen gaan lopen. Er is ook geen datasheet bijgeleverd. De led's zijn RGB-led's. RGB staat voor 'rood-groen-blauw'. Deze drie kleurelementen zijn een RGB-led aanwezig, en hiermee is elke gewenste kleur te verkrijgen door zogenaamde additieve kleurmenging. Zo'n led heeft vier aansluitpinnen. Drie anodes voor elk van de kleuren, en een gezamelijke kathode. Dit type RGB-led heet dan ook 'common kathode' (gezamelijke kathode). Een datasheet is te vinden op http://www.samenkopen.net/action_product/208611/137382.

Basisprincipes PIC microcontroller

Een microcontroller (of kortweg µC) is een geïntegreerd circuit (IC) met een geheugenruimte voor uitvoerbare codes, die door de gebruiker zelf in te stellen is. Oudere microcontrollers waren vaak maar éénmalig te programmeren (te vullen met uitvoerbare codes), en duurdere exemplaren waren te wissen met UV-licht, waarna ze opnieuw te gebruiken waren. Moderne microcontrollers zijn echter zeer vaak elektronisch te wissen en opnieuw te gebruiken, en kosten vrijwel niks.

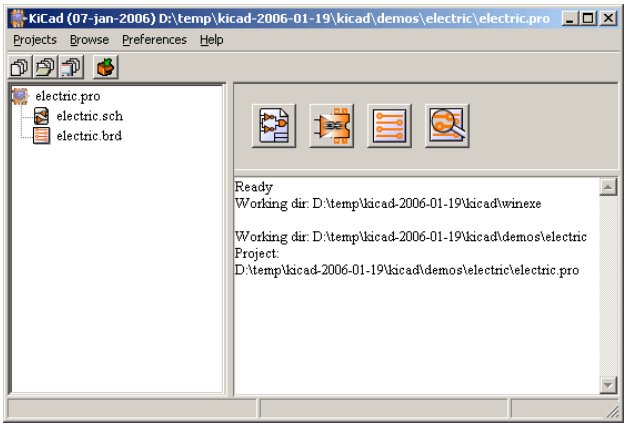


Een microcontroller bestaat uit een aantal onderdelen: een CPU (Central Processing Unit), die de programmacodes en rekenbewerkingen kan uitvoeren; een ROM of EEPROM (Electrically Erasible Programmable Read-Only Memory) geheugen waarin de programmacodes staan en een RAM (Random Access Memory) geheugen, die het programma kan gebruiken om gegevens in op te slaan. De gegevens in het RAM geheugen gaan verloren wanneer de spanning wegvalt en is dus slechts voor tijdelijke opslag. Gegevens die langer bewaard moeten worden kan het programma in het ROM geheugen

opslaan. Daarnaast zitten in geavanceerdere microcontrollers allerlei onderdelen ingebouwd die het gebruik vergemakkelijken, bijvoorbeeld een analoog-digitaal omzetter om analoge signalen te meten en timers die op de microseconde nauwkeurig kunnen tellen.
Een programma kan via de CPU allerlei zogenaamde registers instellen. Dit zijn - net als het RAM - tijdelijke opslagruimtes. Registers zijn echter hardwarematig aan de verschillende onderdelen in de microcontroller verbonden. Zo zijn er logische spanningen op de pinnen van de microcontroller in te stellen door het programma, en kunnen de analoog-digitaal omzetters en timers en andere ingebouwde functies bestuurd worden.
De PIC is te programmeren door middel van 3 pinnen, die bijvoorbeeld op de seriële poort van een computer kunnen worden aangesloten. Ook zijn er programmeerschakelingen die gebruik maken van USB.

Gebruikte software

Tekenprogramma: KiCad

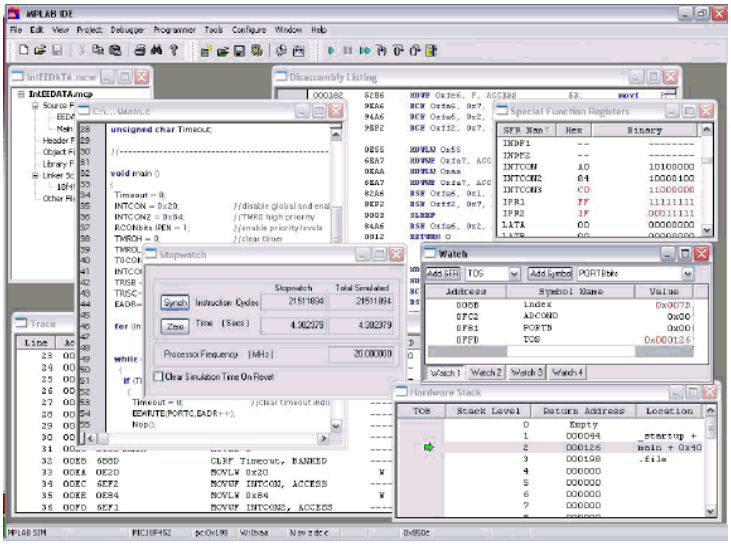


Om een elektronicaschema en printplaat te ontwerpen gebruik ik vaak het tekenprogramma KiCad. Dit is een open-source ontwerpprogramma dat op verschillende besturingssystemen te gebruiken is. Het is gratis te downloaden. Het enige nadeel is dat er geen mooie simulaties in te maken zijn. Het programma is te downloaden op http://www.lis.inpg.fr/realise_au_lis/kicad/.

Ontwikkelomgeving microcontroller programma: MPLAB IDE

Een ontwikkelomgeving of IDE (Integrated Development Environment) is een programma waarin programmacode overzichtelijk wordt gerangschikt en waarin één of meerdere programmeertalen zijn geïntegreerd. Ik gebruik de gratis te downloaden MPLAB IDE, wat de officiële ontwikkelomgeving is van Microchip. Te downloaden op http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

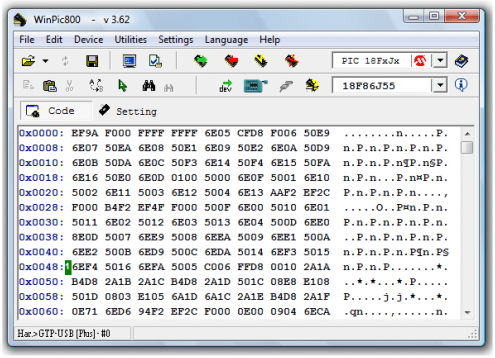
Programmeertaal microcontroller programma: C



Je kunt natuurlijk zelf alle programmacodes stuk voor stuk bedenken en achter elkaar zetten. Dit is wel een hels karwei en lastig. De oplossing hiervoor is programmeren in assembleertaal. Hierbij wordt elke code een naam gegeven, en kun je bovendien geheugenruimtes abstract maken: je geeft stukken geheugenruimte een naam, waarna het programma die het omzet in programmeercodes (de assembleerder) automatisch deze namen omzet in de geheugenruimtes die ze representeren. Zo wordt een programma een stuk overzichtelijker terwijl je wel directe controle over de programmacodes hebt. Op deze manier dingen abstract maken om leesbaarheid ten goede te komen, noemt men 'hoger' programmeren. Je staat immers verder af van het eindresultaat (de daadwerkelijke programmacodes die in de microcontroller terecht komen).

Je kunt echter nog verder afstand nemen. De programmeertaal C is een voorbeeld hiervan. Het is een universele programmeertaal die voor allerlei soorten processors gebruikt kan worden, waaronder microcontrollers maar ook bijvoorbeeld PCs. Complexere functies zijn zo gemakkelijker te maken. Optellen en aftrekken is in assembleertaal vrij gemakkelijk (de PIC kent hier standaard commando's voor). Machtsfuncties zijn echter nauwelijks te doen. Als je in C programmeert hoef je je hier geen zorgen om te maken, de omzetter naar assembleertaal (de compiler) zorgt er zelf voor dat dit efficiënt gebeurt.
Een van de grootste nadelen van een zodanig hoge programmeertaal als C is dat je als programmeur niet precies meer weet wat voor programmacodes nou daadwerkelijk worden uitgevoerd. Sommige functies kunnen in assembleertaal vele malen efficiënter gemaakt worden, dan dat een compiler dat kan doen. De meeste compilers staan daarom ook toe om assembleercodes te combineren met C codes. Zo kunnen kritieke delen van een programma die bijvoorbeeld zeer snel moeten worden uitgevoerd 'met de hand' zo efficiënt mogelijk worden gemaakt, terwijl andere delen in C codes geschreven kunnen worden, wat de leesbaarheid en onderhoudbaarheid van de code ten goede komt. Goede onderhoudbaarheid wil zeggen dat de code makkelijk veranderd kan worden zonder de hele structuur en tientallen (of honderden!) assembleercodes te veranderen.
Ik maak bij het schrijven van de code gebruik van de Microchip C18 Compiler (te vinden op http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014).

Programmer: WinPIC800

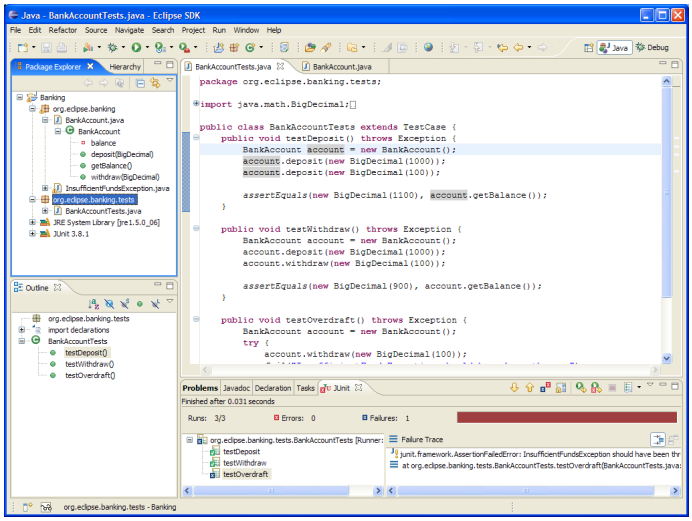


Om de eerste keer de Bootloader, waar ik het later uitgebreider over zal hebben, te kunnen programmeren, is programmer-software nodig. Hiervoor gebruik ik het programma WinPIC800 dat te downloaden is op http://www.winpic800.com.

Bootloader: PICDEM USB Demo Tool

Dit programma gebruik ik om via USB plaatjes en het programma in te laden zodra de bootloader geïnstalleerd is. Hierover later meer.

Ontwikkelomgeving PC programma: Eclipse



Dit is de ontwikkelomgeving die ik gebruik om het PC programma te schrijven, waarmee je een plaatje omzet in een bestand dat leesbaar is voor de microcontroller. Te downloaden op http://www.eclipse.org/ .

Programmeertaal PC programma: Java

Ik schrijf het PC programma in de taal Java. Dit is best een mooie taal, die als voordeel heeft dat er erg veel voorbeeldcode voor te vinden is. Ook kunnen programma's die hiermee gemaakt worden cross-platform draaien, dat wil zeggen zowel op Windows als op Mac, Linux en andere besturingssystemen.

Afbeeldingen: Dia

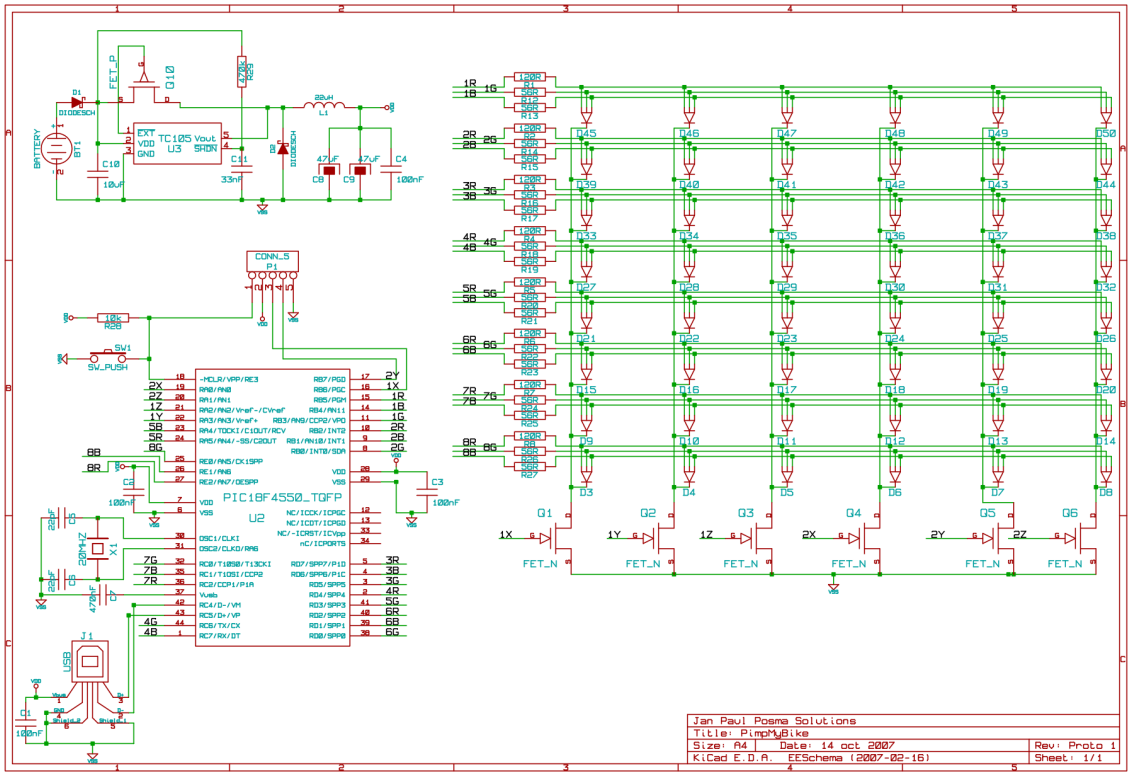
Voor enkele afbeeldingen in die profielwerkstuk maak ik gebruik van het open-source programma Dia. Dia is gratis te downloaden op http://www.gnome.org/projects/dia/ .

Elektronicaschema

Inleiding

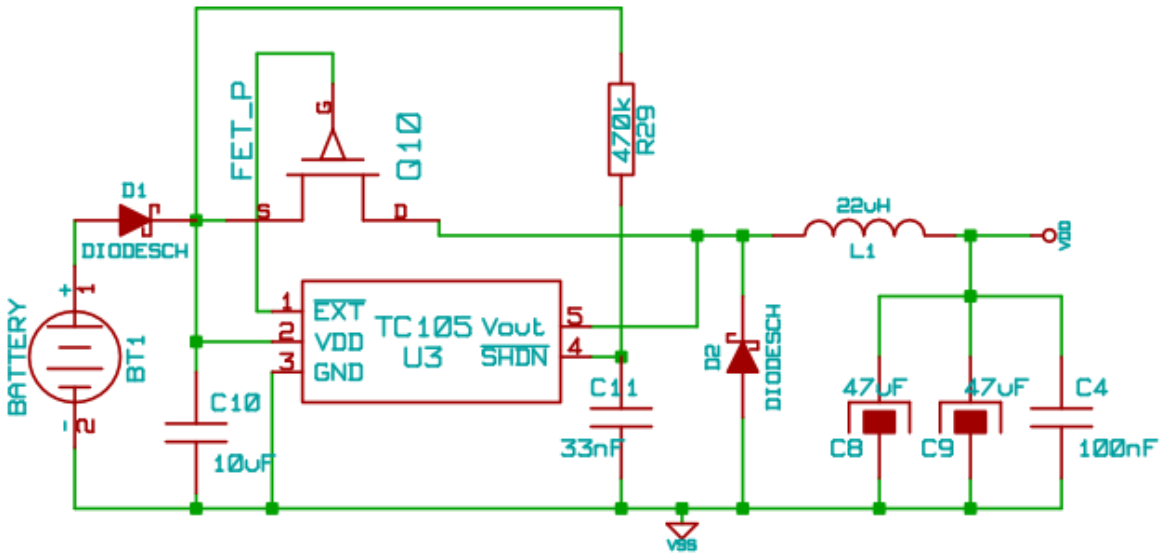
Ik zal verschillende onderdelen van het schema bespreken. Ook allerlei technische functies komen aan bod. Dit hoofdstuk heeft een zekere mate van overlap met het hoofdstuk 'Programmawerking'. Het is daarom aanbevolen eerst dit hoofdstuk te lezen, omdat ik in het hoofdstuk Programmawerking hierop verder bouw.

Overzicht



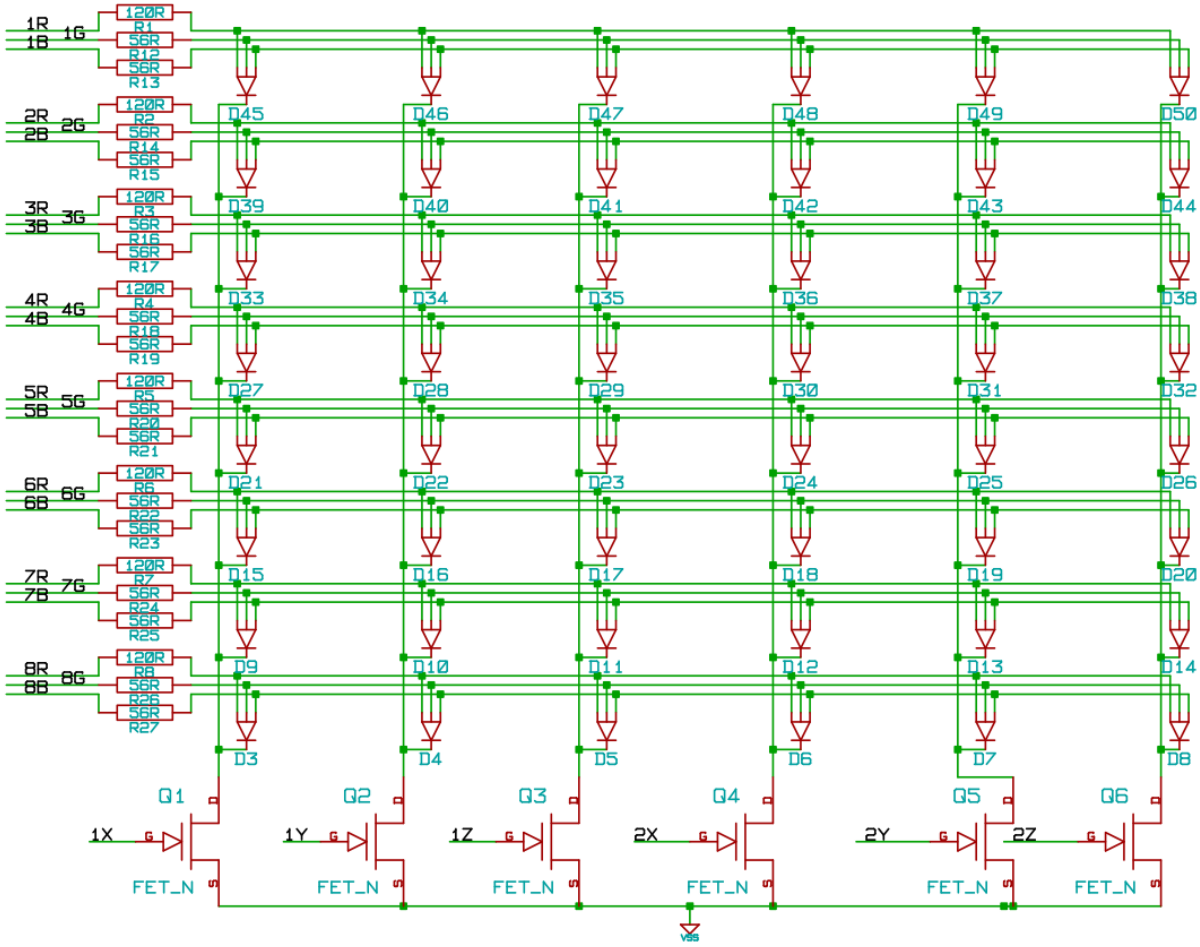
De figuur hierboven is een verkleinde weergave van het schema. Het volledige schema is bijgevoegd als Bijlage B. In de passages hieronder beschrijf ik elk onderdeel in detail. Van belang is dat verschillende delen van het schema door middel van labels aan elkaar worden gekoppeld. VSS is de gezamenlijke min-draad en VDD is +5V ten opzichte van VSS. Ook zijn de led's door middel van labels gekoppeld aan de microcontroller, omdat het anders een onoverzichtelijke wirwar van lijnen zou worden.

Voeding



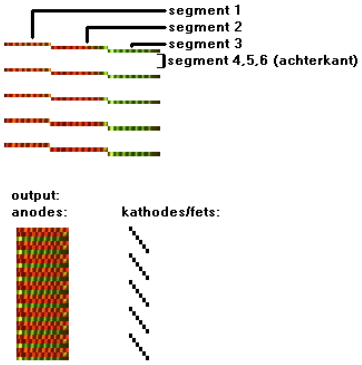
Dit is het voedingselement. Deze zet een gelijkspanning afkomst van een batterij om in 5 volt. Helemaal links is de batterij (BT1) aangegeven. Ik gebruik hiervoor zelf een 9-volt blokje maar elke spanning tussen ongeveer 6 en 16 volt zou moeten werken. Achter de batterij is een Schmitt-diode (D1) geschakeld. Dit is om het circuit te beschermen tegen verkeerd om aansluiten. Een Schmitt-diode is niet duurder dan een normale diode en heeft een zeer lage spanningsval waardoor de minimale spanning van de batterij zo laag mogelijk wordt gehouden. De schakeling erna is namelijk een standaard voedingsschakeling die van elke spanning hoger dan 5 volt, een stabiele 5 volt maakt. Deze schakeling is op basis van de TC105 (U3) van Microchip en maakt gebruik van een P-type FET (Q10), een spoel (L1), een Schmitt-diode (D2) en enkele condensatoren (C10 en C11). Om spanningsfluctuaties die kunnen optreden door bijvoorbeeld het schakelen van de FET of de oscillator, die later beschreven wordt, op te vangen, heb ik enkele condensators en elco's geplaatst in het schema, die fysiek (d.w.z. op de printplaat) dicht in de buurt van deze onderdelen zitten.

Led-Matrix

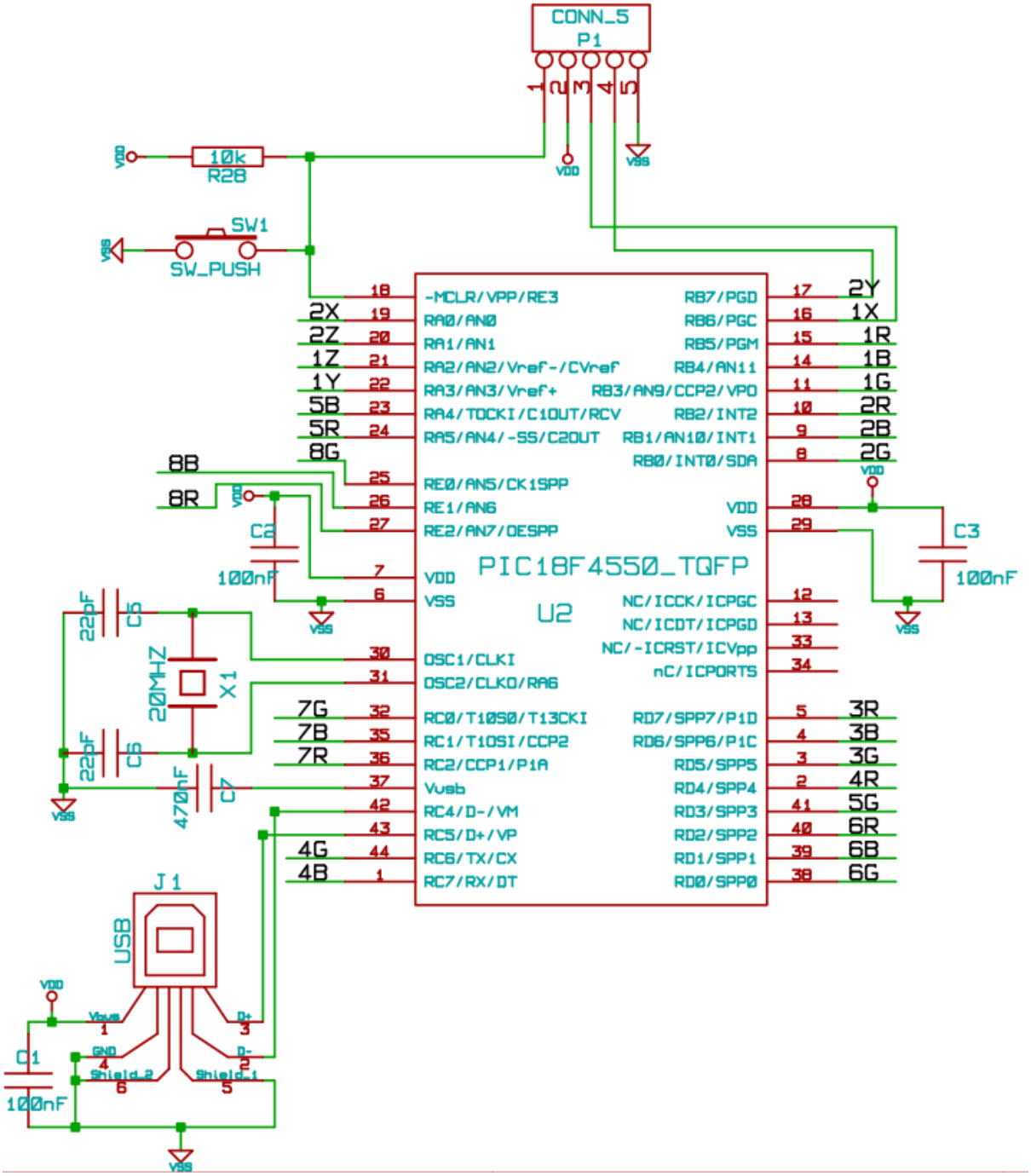


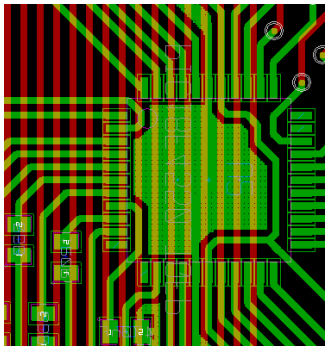
Dit is de Led-Matrix. Alle led's hebben zoals eerder opgemerkt 4 pinnen. De kathode is gezamenlijk. Het zou te veel aansluitingen kosten om alle 48 led's direct op de microprocessor aan te sluiten. Dat zou 48 * 3 = 144 aansluitingen kosten. In plaats daarvan is het veel handiger om een techniek genaamd ‘multiplexing’ toe te passen, zoals ik hierboven ook gedaan heb. Bij multiplexing maak je groepen van led's, die je afwisselend inschakelt. Dat schakelen gebeurt door middel van N-type FETs. FETs zijn in dit geval handiger dan transistors omdat het schakelen met zeer hoge frequentie moet plaatsvinden. In een kolom zijn alle kathodes van de led's op elkaar aangesloten en op één FET aangesloten. Als de gate van Q1 nu hoog wordt dan zullen alle kathodes in de eerste kolom via de FET op VSS worden aangesloten. Door de anodes van de led's kan nu naar gelieven een stroom worden gestuurd, afhankelijk van de gewenste intensiteiten. Vervolgens wordt de gate op Q1 weer laag gemaakt en de gate op Q2 hoog. Nu kunnen deze led's aangestuurd worden.

De anodes van de led's in verschillende kolommen zijn op elkaar aangesloten, waardoor veel aansluitingen worden bespaard. In dit geval zijn er 6 kolommen, die elk 8 led's bevatten. Er zijn dus voor de anodes slechts 3 * 8 = 24 aansluitingen nodig, plus 6 om de FETs aan te sluiten, wat een drastische verlaging van slechts 30 aansluitingen inhoudt tegen 144 als alle led's direct aangestuurd zouden worden. Het grote nadeel van deze manier van aansturen is dat de aansturing zeer snel moet gebeuren, wil je het niet opmerken. Daar komt nog bij, dat de aansturing van de individuele led's al zeer snel moet gebeuren, om een goed beeld in een fietswiel zelfs bij hogere snelheden van de fiets te kunnen realiseren. Desalniettemin is het door zeer efficiënte programmacode toch mogelijk multiplexing te gebruiken.



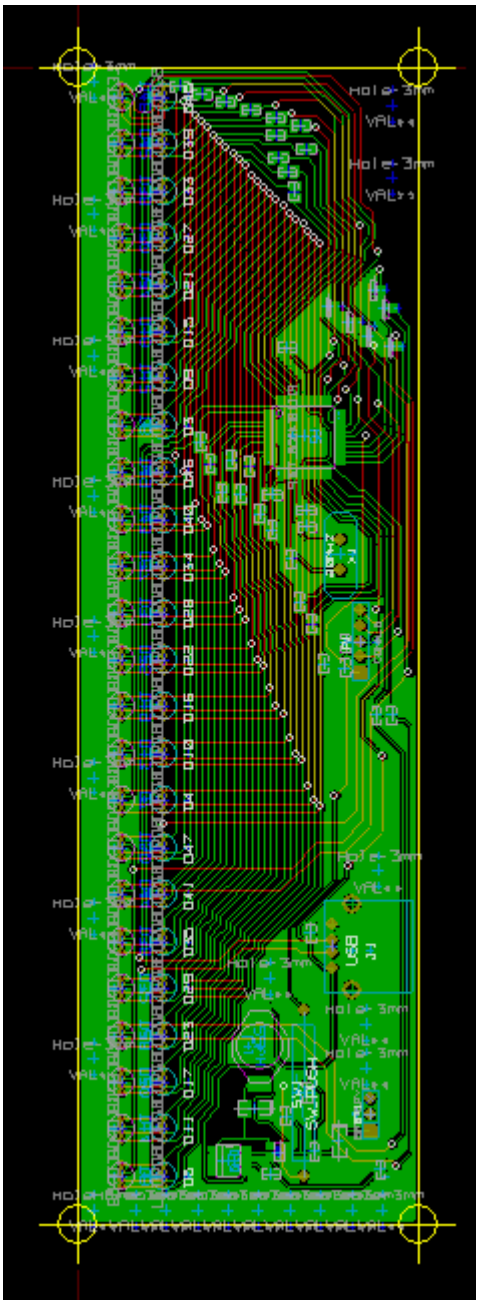
Microcontroller





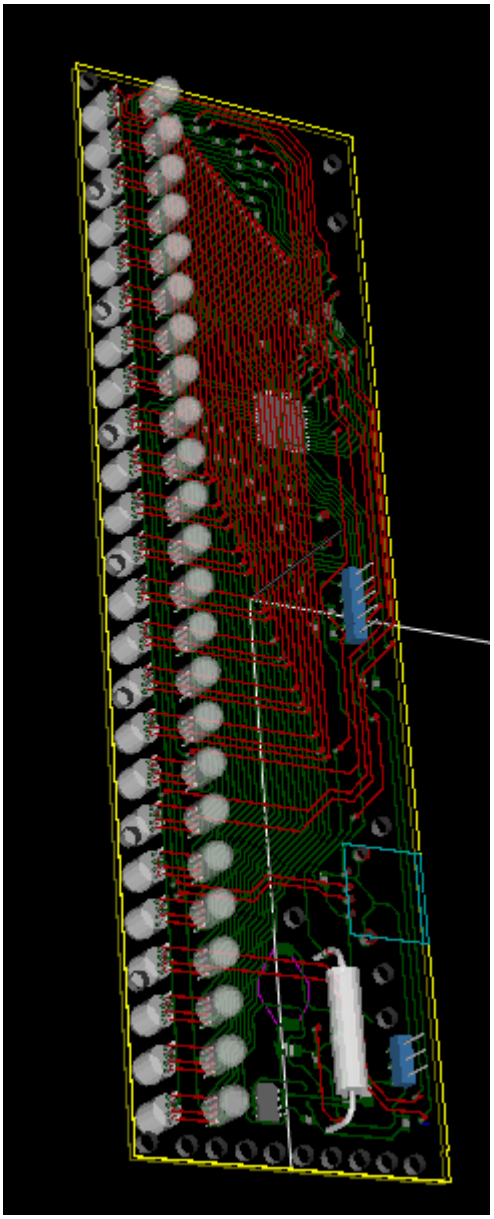
Dit is het hart van de schakeling. De microcontroller (U2) stuurt alle led's immers aan. De anodes van de led's zijn direct op de pinnen van de microcontroller aangesloten. Dit wordt in dit schema aangegeven door de verschillende cijferlettercombinaties. Dit is overzichtelijker dan het trekken van vele lijntjes. Elk cijfer staat voor een groep led's, en de letter geeft het kleurkanaal aan. Dit kan R voor rood, G voor groen of B voor blauw zijn. De volgorde waarin de verschillende groepen led's zijn verbonden met de microcontroller is niet willekeurig, maar heeft te maken met de fysieke volgorde waarin de pinnen op de microcontroller zitten. De volgorde is zo gekozen, dat er zo min mogelijk verbindingen elkaar kruisen. Op de afbeelding van de printplaat rechts is duidelijk te zien dat alle banen netjes langs elkaar lopen.
De klokfrequentie van de microcontroller wordt bepaald door een kristaloscillator. Hiervoor gebruik ik een kristal (X1) van 20 Mhz, met twee koppelcondensators (C5 en C6) van 22pF.
De microcontroller kan geprogrammeerd worden via een 5-polige connector (P1). 2 aansluitingen van deze connector zijn verbonden met de voedingslijnen, en de andere 3 met de pinnen MCLR (master clear), PGD (program data) en PGC van de microcontroller. Door op MCLR een spanning hoger dan 10 volt aan te bieden, wordt de microcontroller in programmeertoestand gebracht. Via de seriële poort van de computer kan vervolgens op PGD en PGC een signaal worden verstuurd die de microcontroller programmeert.
Dit programmeren via de seriële poort hoeft in feite maar één keer te gebeuren, omdat de PIC18F4550 zelf-programmeerbaar is. Daarvan maak ik gebruik door een USB-poort (J1) op de microcontroller aan te sluiten, waarna de microcontroller via de USB-poort programmeerbaar is. De eerste keer wordt dan een programma ingeladen die deze USB-poort inschakelt, want dat is standaard niet het geval.
Na de eerste keer programmeren via een seriële kabel is het programmeren veel gebruikvriendelijker, omdat er dan geen speciale kabel naar de seriële poort van de computer vereist is, maar slechts een normale USB-kabel. Bovendien kan de microcontroller dan tijdens het programmeren via de USB-poort gevoed worden, terwijl bij een seriële kabel de batterij aangesloten moet zijn omdat de seriële poort maar weinig stroom kan leveren.
De MCLR pin is ook als input in te stellen, zodat deze uitstekend te gebruiken is voor de magneetsensor (SW1). Deze heb ik dus óók op de MCLR aangesloten (naast de seriële programmer) in combinatie met een pull-up weerstand (R28).

Printplaat



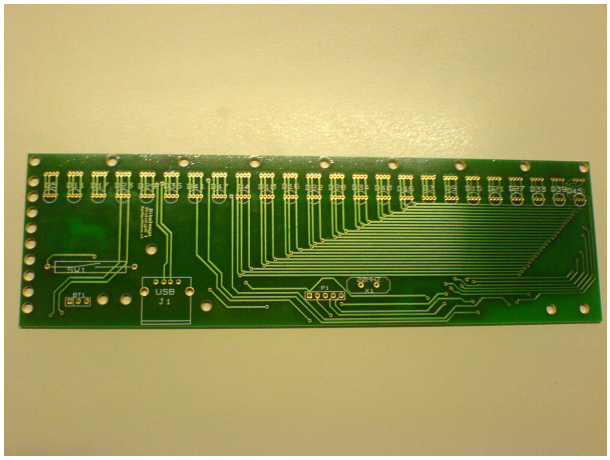
De printplaat heb ik eveneens in Kicad ontworpen. Aangezien er zeer veel ledjes aangestuurd moeten worden zijn hier veel koperbanen voor nodig. Ik heb gekozen om gebruik te maken van een dubbelzijdige printplaat. Dat wil zeggen dat aan beide kanten van de printplaat koperbanen worden aangebracht. Anders zou de printplaat te breed worden en zouden zeer veel draadbruggen nodig zijn. Omdat de printplaat behoorlijk complex is, heb ik deze professioneel laten fabriceren door makepcb.com. Om de kosten per printplaat te drukken heb ik 6 exemplaren besteld.

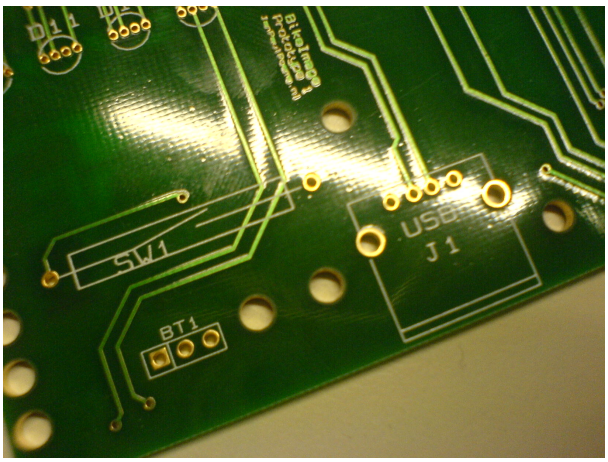
Links is het ontwerp van de printplaat te zien. Groen duidt koper aan de onderkant aan, rood is koper aan de bovenkant. Iets rechtsboven het midden is een vierkant te zien. Dit is de microprocessor. Aan de linkerkant zijn beide rijen ledjes.

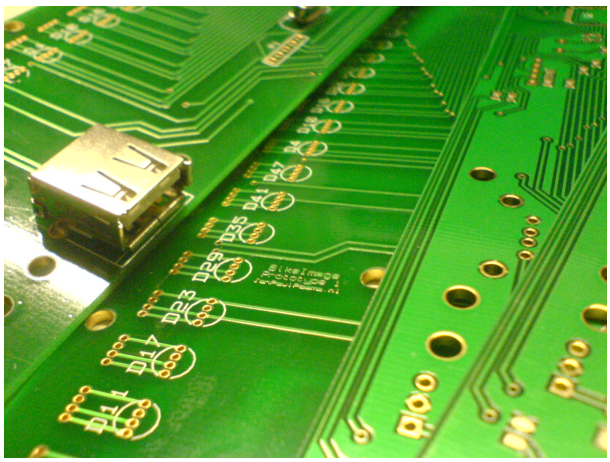


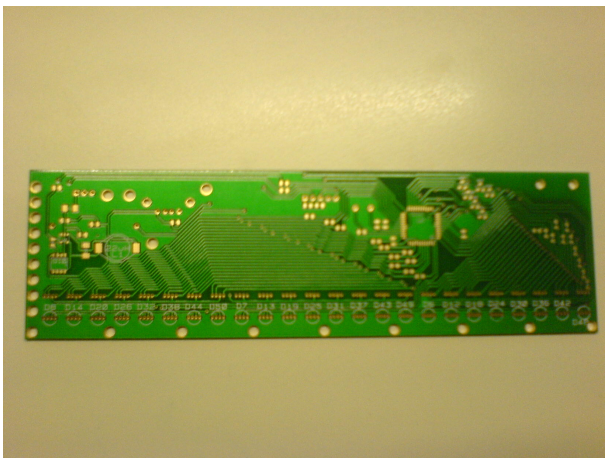
Deze afbeelding, een 3d-model, laat dit nog duidelijker zien. De ledjes zijn hier goed op te zien. De printplaat is doorzichtig afgebeeld zodat componenten aan beide kanten zichtbaar zijn. Helemaal onderaan is ook de magneetsensor goed te zien.

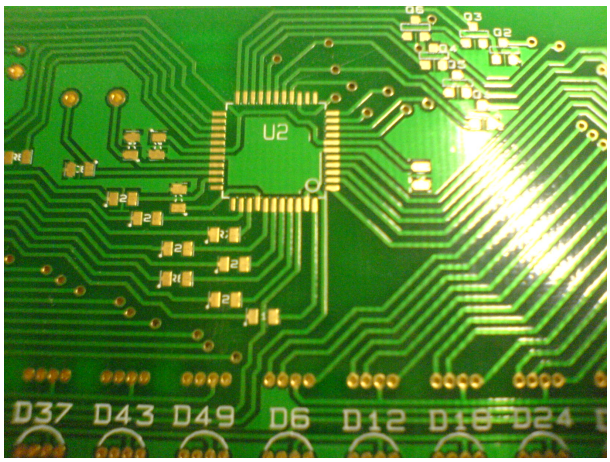
Hieronder enkele foto's van de printplaten.











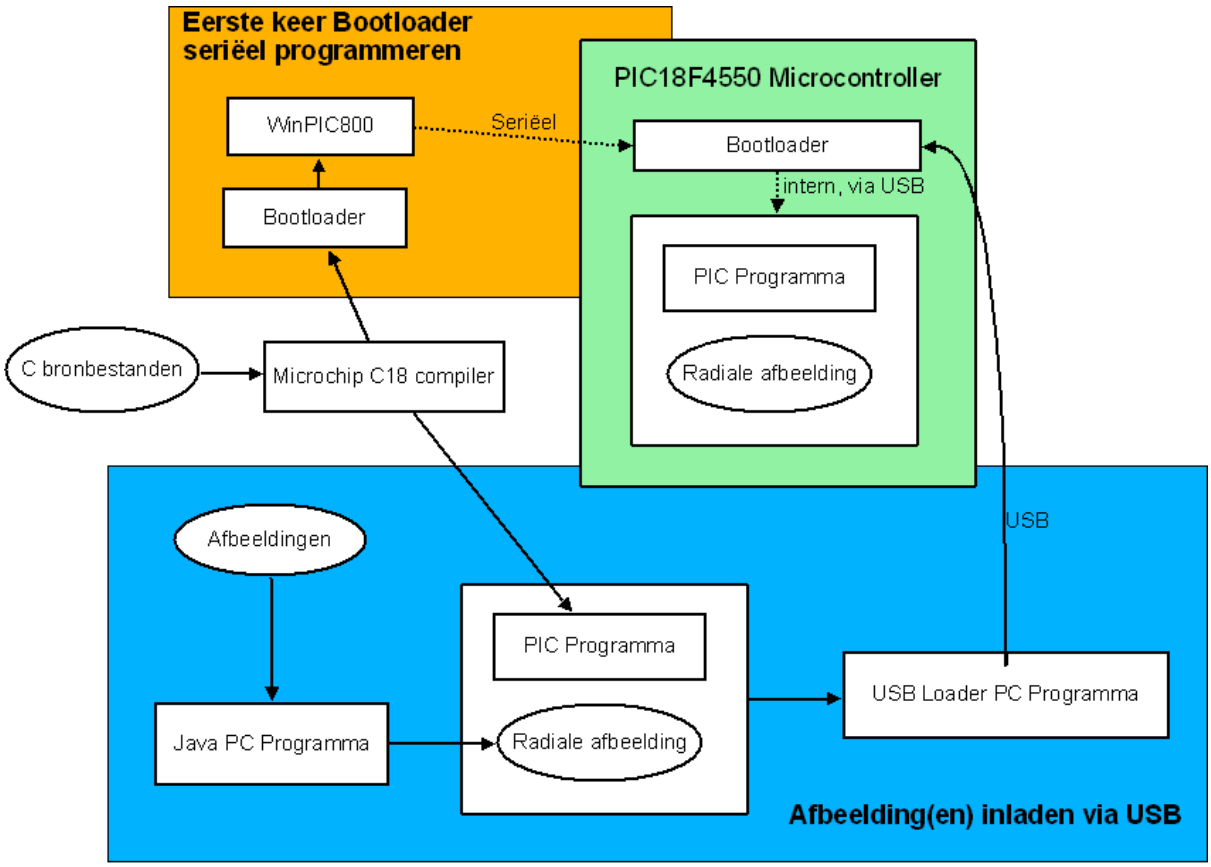
Het solderen duurt een paar uur per printplaat. Maar het resultaat mag er dan ook zijn:

Programmawerking

Inleiding

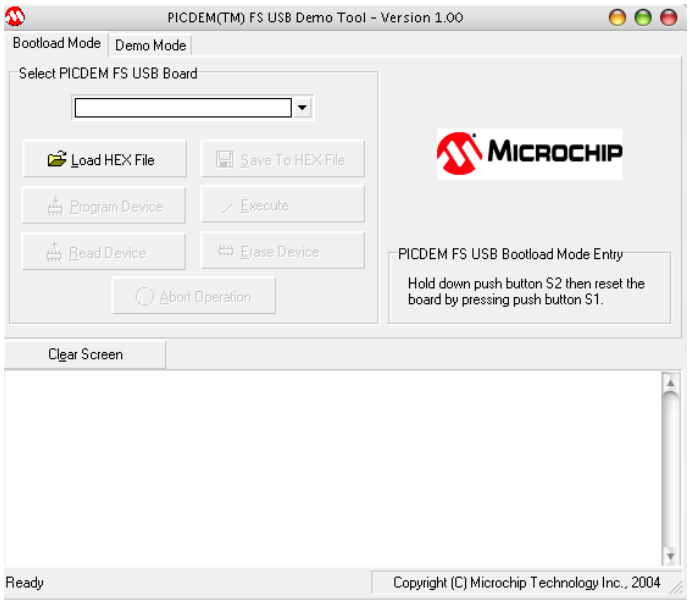
Aangezien het te omslachtig is om alle programmacode regel voor regel te bespreken zal ik slechts de belangrijkste onderdelen bespreken. De Bootloader zal ik niet bespreken, omdat dit vrijwel geheel standaardcode is waar ik slechts enkele dingen aan heb aangepast. Om dezelfde reden zal ik ook het USB Loader programma niet bespreken. Ik zal niet ingaan op de daadwerkelijke code. Deze is bijgevoegd als Bijlage C. Het is echter lastig om zonder kennis van Java, C en assembleertaal de code te begrijpen. Daarom zal ik gebruik maken van afbeeldingen om de programmastructuur duidelijk te maken.

Overzicht



De verschillende programma's en de interacties ertussen zijn in bovenstaand diagram aangegeven. Ten eerste is er een Java programma dat twee afbeeldingen (voor beide kanten van het wiel) omzet in een radiale afbeelding, die geschikt is om in de PIC te laden. Deze

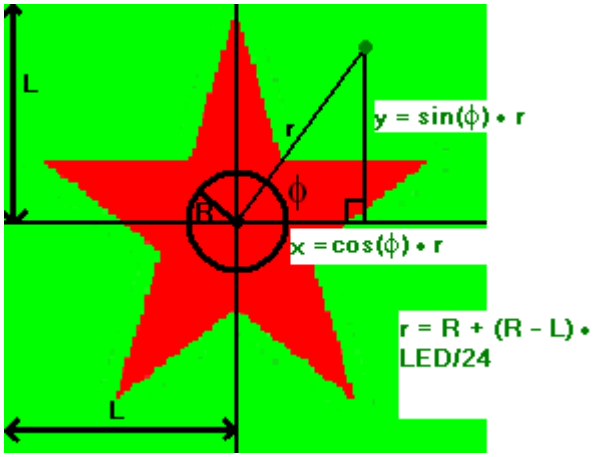
afbeelding wordt gecombineerd met het gecompileerde PIC programma die de led's doet knipperen en de magneetsensor uitleest. Door de USB Loader, een programma van Microchip, wordt deze via de Bootloader, die al in de PIC aanwezig moet zijn, geladen in de microcontroller. Hiervoor moet dus de Bootloader al aanwezig zijn, wat bij fabrieksinstellingen niet het geval is. Deze moet dus eenmalig geprogrammeerd worden, wat met het programma WinPIC800 gebeurt. De Bootloader en het PIC programma worden gecompileerd door de Microchip C18 compiler vanuit C bronbestanden. De bronbestanden van de Bootloader zijn stukken standaardcode die van de Microchip site te downloaden zijn. Ik heb ze enigszins aangepast voor dit project. Het PIC programma is naar eigen ontwerp.



De afbeelding rechts is het PICDEM USB Loader programma. Dit is een voorbeeld programma die bij de bootloader van Microchip hoort. Er is hier nog ruimte voor verbetering: dit programma zou voor een optimale ervaring van de gebruiker geïntegreerd moeten worden in het Java PC programma.

Java programma

Het Java programma is een vitaal onderdeel om het geheel werkend te krijgen. Dit programma zet namelijk een normale afbeelding (bijvoorbeeld JPG) om in een radiale afbeelding, en vervolgens in een bestand waar het PIC programma mee uit de voeten kan.

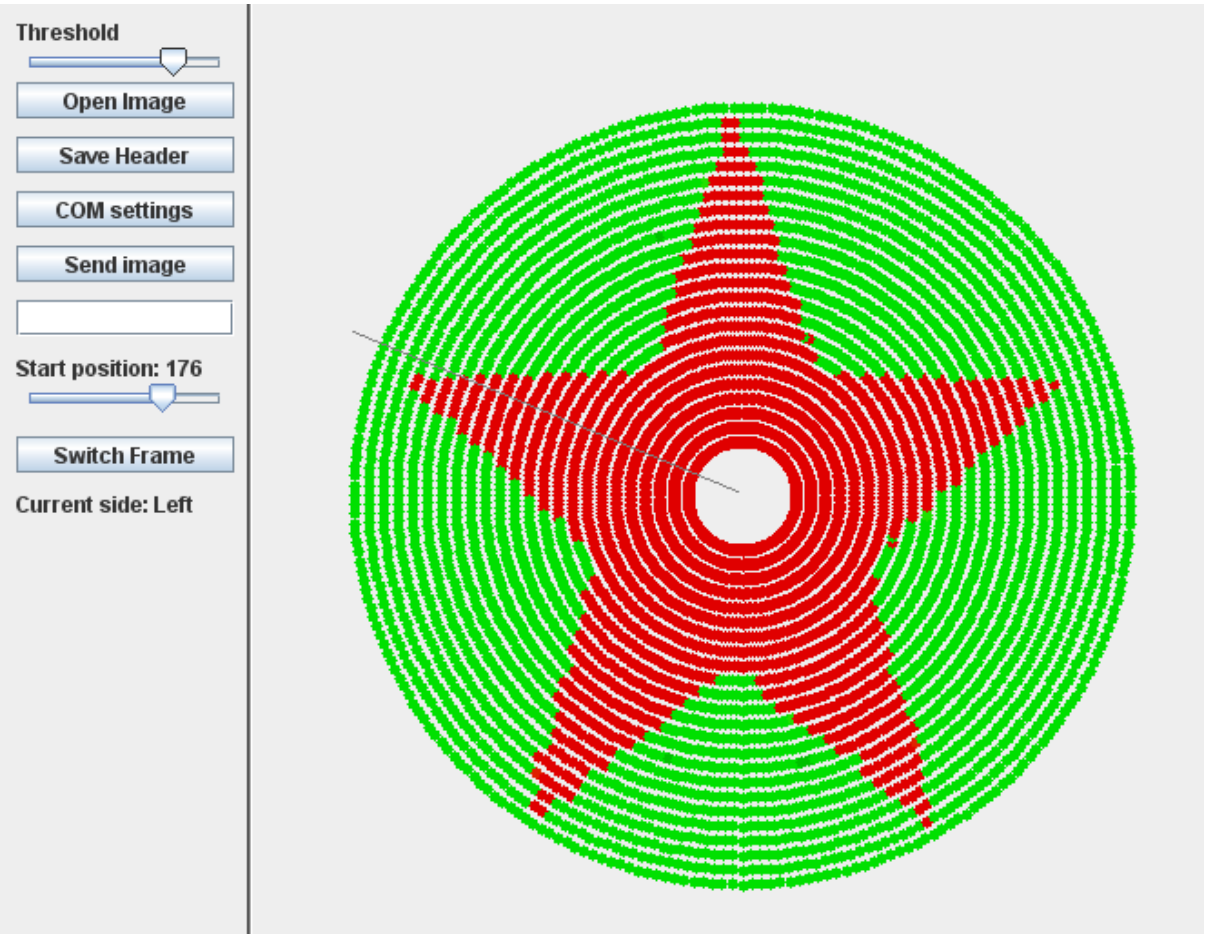


Links is de Heineken-ster te zien welke ik als voorbeeld geladen heb. Het omzetten naar een radiaal beeld gaat als volgt. Er wordt een hoek fi-0 gekozen door de gebruiker. Dit is in het programma aangegeven met een grijze lijn. Deze hoek fi-0 moet gelijk zijn met de hoek waaronder de magneetsensor wordt geplaatst ten opzichte van het middelpunt van het wiel en de weg. Deze grijze lijn kun je dus ook wel zien als een lijn die het frame van de fiets aangeeft. De magneetsensor is immers op het frame bevestigd. Voor alle leds wordt nu de eerste beeldlijn bepaald. Dat gebeurt door in het plaatje de pixels op te zoeken die onder deze hoek vallen. In de afbeelding links zijn de x en y formules hiervoor aangegeven. De oorsprong van het vlak wordt hiervoor in het middelpunt van de afbeelding geplaatst. Voor elke led worden nu deze formules ingevuld. Voor binnenste led wordt LED = 0 ingevuld, en voor de buitenste LED = 24. Er zitten immers 24 led's aan elke zijde.

Het middelste gedeelte, een circel met straal R, wordt weergegeven door led's. Hier zit immers de as van de fiets. In een toekomstige versie zou deze straal R in te stellen moeten zijn. De hoek fi wordt telkens met een stapje vergroot, en de informatie van de leds wordt weer uit het plaatje gehaald door de pixels op desbetreffende x-y-positie uit te lezen. Zo wordt een compleet beeld opgebouwd. Dit beeld bestaat uit 256 lijnen. De hoek fi dus in stapjes van ((2*pi)/256) radialen vergroot.

Dit hele proces gebeurt tweemaal. Eenmaal voor de linkerkant, en eenmaal voor de rechterkant. Het is ook mogelijk op beide kanten verschillende plaatjes te tonen.

Merk op dat in het echt de kleuren een stuk donkerder zijn. Hier is ook verbetering mogelijk: een meer realistische weergave in het programma zou duidelijker zijn.



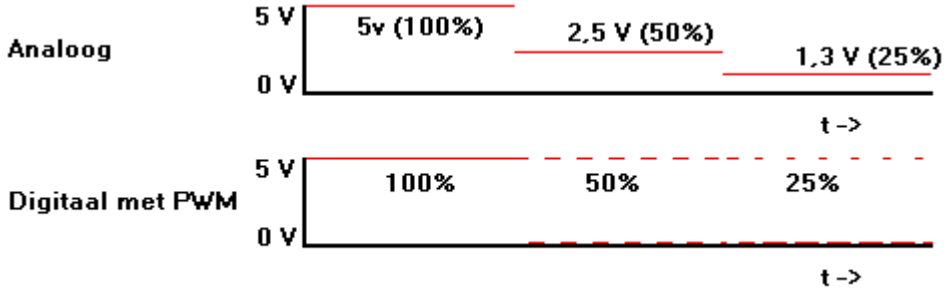
Zo ziet het programma er dan uit. Om een voorbeeld op het scherm te tekenen, wordt de informatie van de beeldlijnen gebruikt en deze op dezelfde manier weer *terug* omgezet naar een x-y-afbeelding. Op de plek van elke led wordt een kleine cirkel getekend. Hier is overigens duidelijk te zien dat in het midden van de afbeelding de resolutie een stuk hoger is, omdat de straal en dus de omtrek daar korter is terwijl er nog steeds 256 beeldpunten per ledje opgeslagen zijn.

Het programma is nog niet echt af. Het beeld kan worden opgeslagen via de knop Save Header, maar nog niet worden verstuurd via USB.

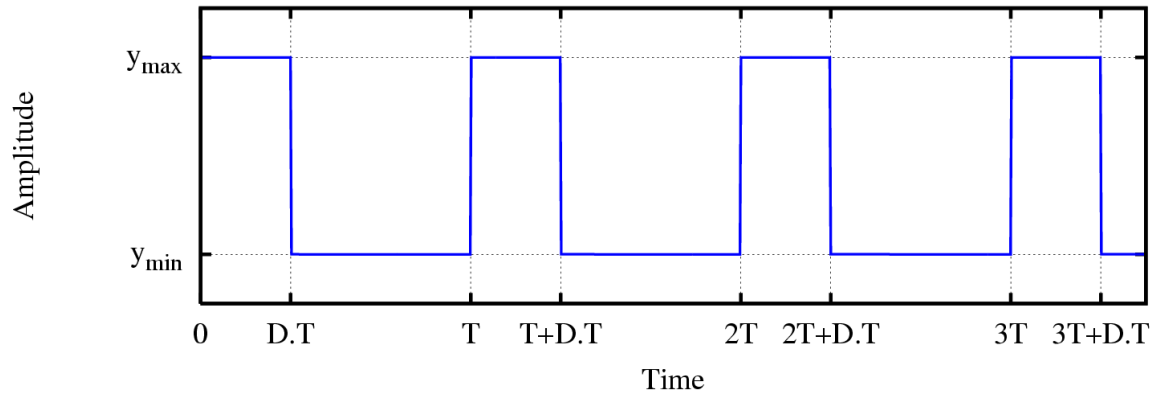
PIC Programma

Het PIC programma bestaat voornamelijk uit 3 loops die in elkaar genest zijn. Dit zijn de loops:

- Het afwisselen van de kleuren van alle led's. Dit gebeurt 265 keer per omwenteling. De informatie van de kleuren staat opgeslagen in het programmageheugen (de radiale afbeelding). De interval tussen de activeringen van de magneetsensor bepaalt de snelheid van deze loop.
- Het afwisselen van de verschillende multiplexing segmenten. Zoals eerder toegelicht bij het onderdeel Electronicaschema - Led-Matrix, zijn er 6 segmenten van elk 8 led's. Er kan maar één segment op een tijdstip aangestuurd worden. De verschillende segmenten moeten dus telkens afgewisseld worden, en snel genoeg dat het niet waarneembaar is. De snelheid van deze loop ligt vast en is maximaal. Dat wil zeggen, er is geen softwarematige beperking op de snelheid. De snelheid wordt dus bepaald door de klokfrequentie van de microcontroller.
- Het vormen van kleuren door middel van PWM (Pulse Width Modulation). Op de werking ga ik dadelijk in. De snelheid hiervan is eveneens maximaal.



PWM is een techniek om tinten te vormen terwijl alleen de logische 0 volt en 5 volt beschikbaar zijn. Normaliter zou het voor de hand liggen om het voltage af te wisselen om verschillende kleurtinten te vormen, dat wil zeggen op één kleurkanaal is dan 0 volt uit en 5 volt maximaal en alles daartussen een bepaald percentage. Zo zou je geel kunnen vormen door groen en rood beide op 5 volt te zetten, of donkergeel door beide op bijvoorbeeld 3 volt te zetten. Echter, alleen 0 en 5 volt zijn beschikbaar. Door een blokgolf te maken en de breedte van de toppen te variëren, kan het *gemiddelde* voltage bepaald worden. Als dit snel genoeg gebeurt ziet het oog daar niks van. Bovendien, hoe vaker het voltage in te stellen is, hoe nauwkeuriger het gemiddelde voltage gevarieerd kan worden.



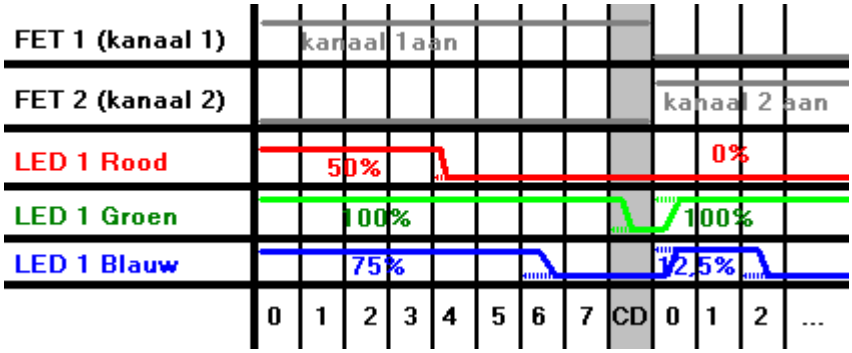
Bovenstaande figuur maakt dit nog duidelijker. T moet zo laag zijn dat het oog de knippering niet waarneemt. Door D tussen 0 en 1 te variëren, kan het gemiddelde voltage bepaald worden. Hoe vaker het voltage ingesteld kan worden, hoe nauwkeuriger D is. In mijn programma kan D op 8 niveaus ingesteld worden, dus van 0 tot 7. Elke 1/8T wordt het voltage ingesteld. Deze waarde is experimenteel bepaald. Aangezien de snelheid van de loop maximaal is, is de nauwkeurigheid van D evenredig met T. Als D twee keer zo nauwkeurig zou zijn (dus elke 1/16T het voltage zou worden ingesteld), dan zou T ook twee keer zo groot zijn. Bij een resolutie van 8 niveaus, dus 3 bits, was het knipperen niet waarneembaar voor het oog. Daarboven werd het merkbaar.

Aangezien de PWM-resolutie 3 bits is, is de benodigde informatie voor elk ledje 3 bits * 3 kleuren = 9 bits. Echter, de PIC rekent in eenheden van 8 bits. Die 9e bit zou het verwerken van de informatie behoorlijk vertragen en complexer maken. Daarom maak ik gebruik van een opslagmethode die ook bij computers gebruikelijk is: 8-bits kleuren ofwel 256 kleuren. De kleur blauw wordt dan in slechts 2 bits opgeslagen. Dat is acceptabel, aangezien ons oog minder gevoelig is voor blauw. De informatie voor de kleur van elk ledje op een gegeven tijdstip is dan zo ingedeeld: *RRRGGGBB*. Dit is snel te decoderen. Bij het decoderen wordt het blauw kanaal vermenigvuldigd met 2, zodat er wel weer gebruik gemaakt kan worden van de 3-bits PWM functie.

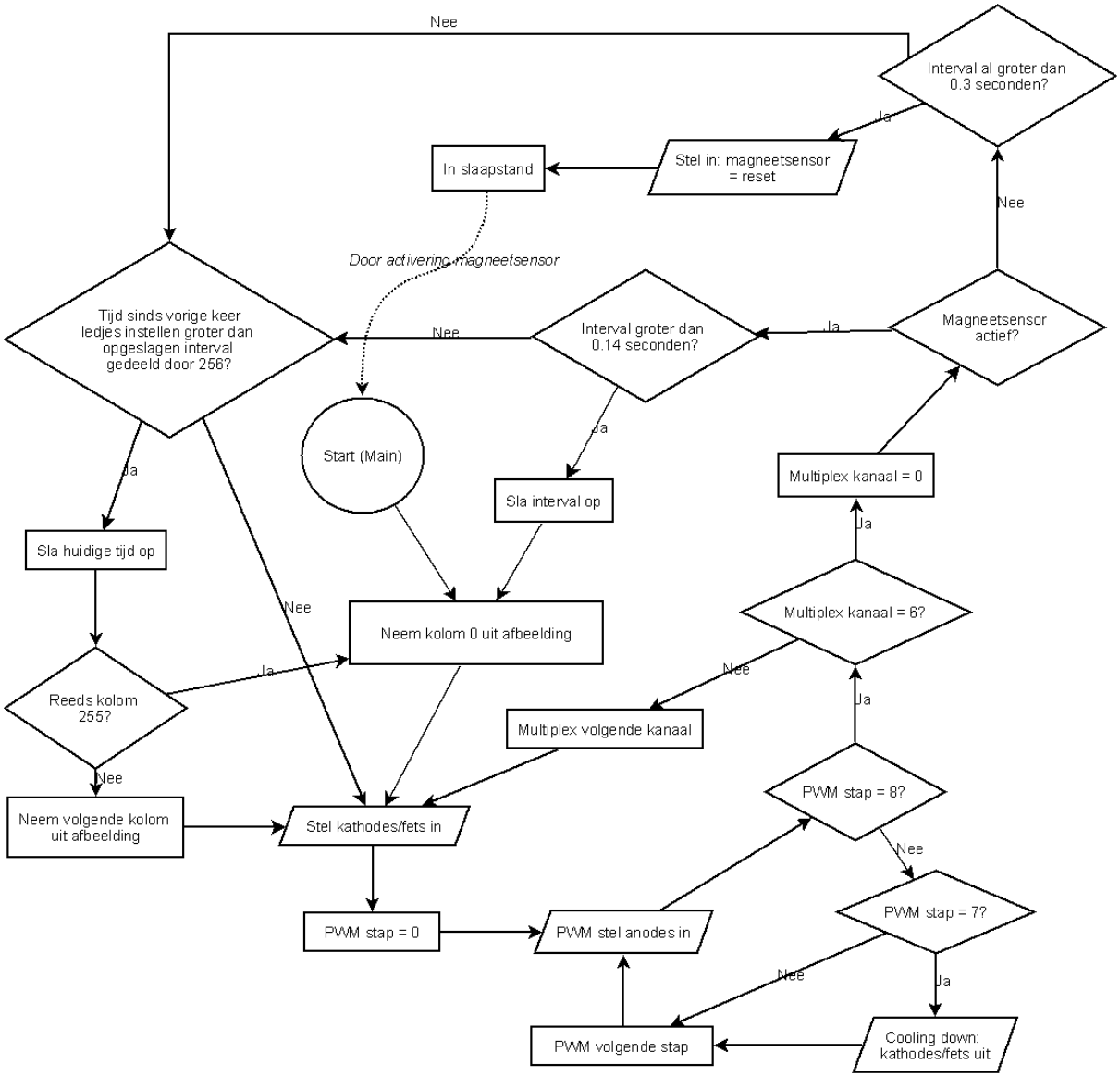
Bij het nesten van de verschillende loops heb ik een keuze moeten maken in welke volgorde ze genest zouden zitten. Het afwisselen van de kleuren staat boven de PWM functie en het multiplexing, maar of de PWM boven de multiplexer moet staan of andersom was nog onduidelijk. Misschien is dit wat lastig voor te stellen maar het zit zo: moet eerst de PWM alle kanalen hoog of laag maken naar gelang de kleuren, en vervolgens de multiplexer alle 6 kanalen aflopen (dan staat de PWM boven de multiplexer)? Of moet de multiplexer een kanaal instellen en vervolgens de PWM 8 keer bij langs gelopen worden voor dat kanaal, waarna de multiplexer het volgende kanaal kiest?

Ook dit heb ik experimenteel bepaald, hoewel het resultaat al voor de hand lag. De PWM functie is het meest tijd-intensief. Sterker nog, het neemt naar schatting 95% van alle tijd in beslag. Als de PWM boven de multiplexer zou staan, zou bij het inschakelen van het nieuwe kanaal, de toestand van het oude kanaal nog op de anodes staan. Wanneer alle led's dan weer geupdate zijn naar de nieuwe stand, wordt alweer het nieuwe kanaal door de multiplexer ingeschakeld. Er vindt dan een enorme kleurverschuiving plaats, waardoor er nog maar een wirwar van kleuren wordt waargenomen.

Daarom heb ik de multiplexer boven de PWM geplaatst. Ook nu zou kleurmenging plaats vinden, maar slechts in 1/8 van de tijd, namelijk in de eerste keer dat het nieuwe kanaal wordt ingeschakeld, en de PWM de led's voor het eerst moet updaten. Toch was die kleurmenging wel zichtbaar. Daarom heb ik een 9e periode toegevoegd, na de 8 PWM periodes, waarin alle led's worden uitgeschakeld. Dit noem ik de cooling-down periode. Wanneer dan het nieuwe kanaal wordt ingeschakeld, zijn alle led's bij voorbaat uit, en wordt er dus geen kleur gemengd. Deze cooling-down periode is veel minder goed waarneembaar dan de kleurmenging.



De MCLR pin, waar de magneetsensor op is aangesloten, heeft een paar leuke eigenschappen. Ten eerste is deze pin als input in te stellen, maar deze pin is ook in te stellen als reset. Wanneer de pin in deze modus laag wordt, zal de microcontroller zijn programma van voor af aan beginnen.
Door deze functies vernuftig te combineren kan energiebesparing gerealiseerd worden. Wanneer het programma start, wordt deze pin ingesteld als input. Dit kan omdat deze microcontroller zelf-programmeerbaar is. Nu kan de rotatiesnelheid van het wiel worden bepaald, door de tijdsinterval tussen de dalen op deze pin te meten. Wanneer het wiel te langzaam draait of stilstaat, kan de microcontroller zichzelf uitschakelen. Maar vlak voor het uitschakelen kan de MCLR pin worden ingesteld als reset. Wanneer het wiel dan weer begint te draaien, wordt de microcontroller gereset zodra de sensor langs de magneet komt. Bij het resetten wordt uiteraard ook de microcontroller weer ingeschakeld. Zo kan het systeem worden uitgeschakeld wanneer het wiel stilstaat, en weer worden ingeschakeld zodra de sensor langs de magneet komt.
Hieronder is een afbeelding die de programmastructuur weergeeft.


Nee
Interval al groter dan 0.3 seconden?
Ja
Stel in: magneetsensor = reset
In slaapstand
Door activering magneetsensor
Nee
Tijd sinds vorige keer ledjes instellen groter dan opgeslagen interval gedeeld door 256?
Nee
Interval groter dan 0.14 seconden?
Ja
Magneetsensor actief?
Start (Main)
Ja
Sla interval op
Multiplex kanaal = 0
Ja
Sla huidige tijd op
Nee
Neem kolom 0 uit afbeelding
Multiplex kanaal = 6?
Ja
Reeds kolom 255?
Nee
Multiplex volgende kanaal
Nee
Ja
PWM stap = 8?
Nee
Neem volgende kolom uit afbeelding
Stel kathodes/fets in
PWM stap = 0
PWM stel anodes in
PWM stap = 7?
Nee
Ja
PWM volgende stap
Cooling down: kathodes/fets uit

Referenties

Literatuur

- Elektronica echt niet moeilijk : deel 1, 2 en 3 - *A. Schommers, 14e druk, Elektuur B.V., ISBN 90-5381-028-5*

Websites

- www.microchip.com (onderdelen, datasheets, programma-afbeelding, software)
- www.samenkopen.net (onderdelen, datasheets)
- www.lis.inpg.fr/realise_au_lis/kicad/ (software, programma-afbeelding)
- www.makepcb.com (printplaten)
- www.tme.pl (onderdelen, datasheets)
- www.eclipse.org (software, programma-afbeelding)
- www.winpic800.com (software, programma-afbeelding)
- www.gnome.org/projects/dia/ (software, programma-afbeelding)
- www.wikipedia.org
    - http://en.wikipedia.org/wiki/Pulse-width_modulation
    - http://upload.wikimedia.org/wikipedia/commons/9/9e/Duty_cycle_general.png
    - http://en.wikipedia.org/wiki/Pic_microcontroller
    - http://upload.wikimedia.org/wikipedia/commons/4/4c/PIC_microcontrollers.jpg
    - http://en.wikipedia.org/wiki/C_%28programming_language%29
    - http://en.wikipedia.org/wiki/Assembly_language

Opmerking: Ik heb Wikipedia slechts gebruikt als verlengstuk van mijn eigen kennis, met name om duidelijke afbeeldingen te vinden. Ik ben van mening dat voor deze doeleinden Wikipedia zeer geschikt is. Lees verder over Wikipedia als onderzoekshulpmiddel op http://en.wikipedia.org/wiki/Wikipedia:Researching_with_Wikipedia.

Logboek

maandag 25 juni 2007

- Aftrap profielwerkstuk
- 2 uur gewerkt aan onderdelen selectie

dinsdag 26 juni 2007

- 3 uur gewerkt aan onderdelen selectie en prototype maken op breadboardje en testen

vrijdag 31 augustus 2007

- 4 uur gewerkt aan onderdelen selectie en tekenen van schema voor eerste echte prototype op printplaat

zaterdag 1 september 2007

- 1 uur gewerkt aan bestelling op Samenkopen.net

zondag 2 september 2007

- Bestelling geplaatst en betaald

woensdag 12 september 2007

- Bestelling binnen, 2 uur gewerkt aan tekenen van schema en footprints kiezen bij onderdelen

vrijdag 14 september 2007

- 3 uur gewerkt aan footprints maken en schema afmaken

zaterdag 15 september 2007

- 5 uur gewerkt aan printplaat tekenen en 3d modellen opzoeken

maandag 17 september 2007

- 2 uur gewerkt aan printplaat tekenen en 3d modellen opzoeken

dinsdag 18 september 2007

- 3 uur gewerkt aan printplaat tekenen en 3d modellen opzoeken

vrijdag 21 september 2007

- 2 uur gewerkt aan printplaat tekenen en 3d modellen opzoeken, is zo goed als af

maandag 24 september 2007

- 2 uur gewerkt aan zoeken van internetwinkels en prijsvergelijking

dinsdag 25 september 2007

- 1 uur gewerkt aan zoeken van internetwinkels en prijsvergelijking en een goede site gevonden: www.tme.pl

maandag 1 oktober 2007

- 1 uur gewerkt aan plaatsen van bestelling bij tme.pl (allerlei SMD onderdelen) en nog een bij samenkopen.net (reedcontacten e.d.)

zaterdag 6 oktober 2007

- 4 uur gewerkt aan opstellen en programmeren van breadboardje met RGB-led

dinsdag 9 oktober 2007

- Bestelling tme.pl binnen, uitgepakt en geordend

woensdag 10 oktober 2007

- 2 uur gewerkt aan programmeren en testen met breadboardje en simulaties draaien om snelheid te bepalen

donderdag 11 oktober 2007

- 1 uur gewerkt aan nieuwe footprints voor printplaat (nieuwe onderdelen van tme.pl)
- 3 uur geprogrammeerd

maandag 19 november 2007

- Printplaten binnen uit China, 5 uur bezig geweest met solderen

dinsdag 20 november 2007

- 3 uur gesoldeerd

woensdag 21 november 2007

- 4 uur gesoldeerd en geprogrammeerd

donderdag 22 november 2007

- 4 uur geprogrammeerd

vrijdag 23 november 2007

- 5 uur geprogrammeerd, eerste keer werkend! (zonder magneet nog)

zondag 25 november 2007

- 6 uur geprogrammeerd en getest, het werkt! Ook gefietst ermee

maandag 26 november 2007

- Gedemonstreerd op school, met Nederlandse vlag en radioactief-symbool erin geprogrammeerd
- 's Avonds gefietst door het dorp

vrijdag 30 november 2007

- Foto's gemaakt met lange sluitertijd van Heineken-logo (1 uur)

zaterdag 29 december 2007

- Introductie en over de programmeertaal geschreven (1 uur)

zaterdag 5 januari 2007

- Verschillende stukken tekst geschreven (2 uur)

zondag 6 januari 2007

- Verschillende stukken tekst geschreven (5 uur)

maandag 7 januari 2007

- Verschillende stukken tekst geschreven (6 uur)

woensdag 16 januari 2007

- Laatste delen en bijlagen toegevoegd (1 uur)

Bijlagen

Bijlage A: GNU Public License versie 3

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.
The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.
To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.
Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.
For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.
Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.
Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.
“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.
“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.
To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.
A “covered work” means either the unmodified Program or a work based on the Program.
To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.
To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.
An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.
A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.
The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.
The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.
The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.
You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.
Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.
When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.
You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.
“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.
If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).
The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.
Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.
When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.
Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.
You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.
A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.
Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.
In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.
If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.
If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.
A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific

products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.
Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.
If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.
Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

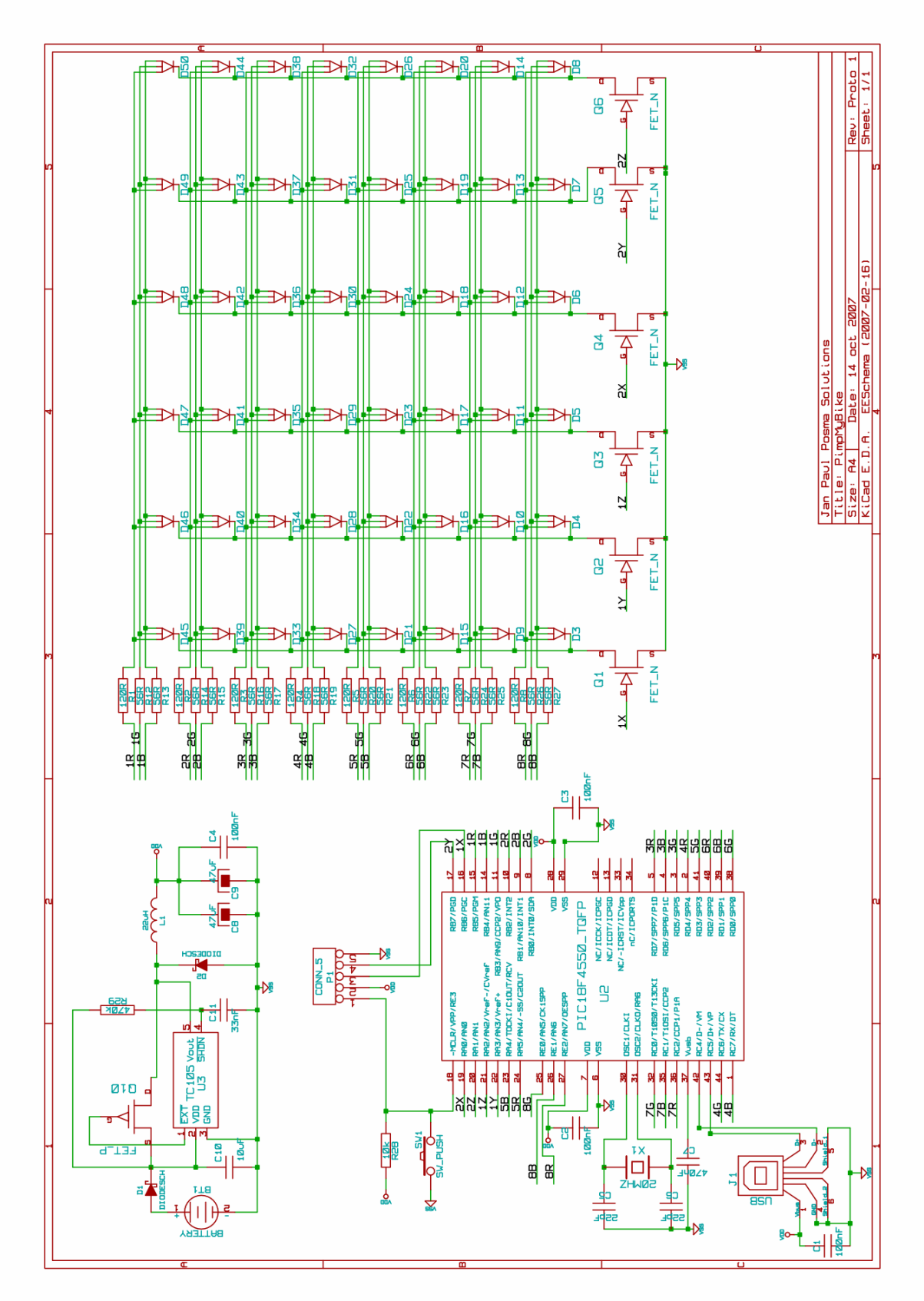
16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.
END OF TERMS AND CONDITIONS

Bijlage B: Gehele schema



Bijlage C: broncode

De volledige broncode van zowel het Java programma als het microcontroller programma is te downloaden op http://www.janpaulposma.nl/pimpmybike.