# ROBO-BITS-21-

**Jaargang 6, nummer 2, Juni 2003**

Afz. HCC Robotica, p.a. A. Vreugdenhil, Regulierenstraat 11, 2694 BA 's-Gravenzande

PTT Post
Port betaald
Port payé
Pays-Bas

**Zaterdag 5 juli, Hengelo**

**Zaterdag 2 Augusteus, Nieuwegein**

Robotica gebruikersgroep

# Inhoud



# Colofon


ROBOBITS is een uitgave van de Robotica-GG, en wordt naar alle leden van de gebruikersgroep opgestuurd.
De oplage is 600 ex.
De Robotica-GG is een onderdeel van de Hobby Computer Club.
Redactie adres: A. Vreugdenhil, Regulierenstraat 11, 2694 BA 's-Gravenzande.
E-mail: a.vreugdenhil@hccnet.nl.
Tekst aanleveren in Word of platte tekst in ascii en afbeeldingen er "los" bij in TIF of JPG formaat.


# Re(d)actie

Deze zomer maar liefst twee bijeenkomsten, zaterdag 5 Juli in Hengelo, wees erbij, en zaterdag 2 Augustus in Nieuwegein. Dit wordt zeer waarschijnlijk de locatie waar we komende tijd onze stek zullen vinden na het sluiten van de locatie in Gouda. In Nieuwegein hopen we met meer gebruikersgroepen samen te komen. We praten ondermeer met de AI-GG, NewBrain-GG en de 6500-GG. We hopen ook op deze locatie veel leden te ontmoeten om van elkaar te leren en elkaar te inspireren. Dat laatste lukt wel als we een paar artikelen lezen van mensen die naar aanleiding van bijeenkomsten en discussies zijn ontwikkeld en uitgevoerd. Zie pagina 26. Veel leesplezier en tot ziens op 5 Juli en 2 Augustus.

Abraham Vreugdenhil.

# Bestuur

Secretaris
A.J. Janssen
Galjoenstraat 65
3534 PD UTRECHT
030-2444944
lex.janssen@hccnet.nl

Voorzitter
B.T.J.A. Buiskool
Pilotenlaan 11
7943 CH MEPPEL
0522-241444

Penningmeester
A. Vreugdenhil
Regulierenstraat 11
0174-420361
2694 BA S'GRAVENZANDE
a.vreugdenhil@hccnet.nl

Lid
P. Smits
Lijtweg 302
2341 HB OEGSTGEEST
071-5156090
psmits.1@hccnet.nl

Technisch adviseur
Ing.H.M.A. van Bodegom
Stadionlaan 180
7552 VE HENGELO OV
074-2434147
ing.h.m.a.van.bodegom@hccnet.nl

Robotica gebruikersgroep

# Hengelo, here we come !
# op zaterdag 5 Juli

Clubinfo
PC Gebruikersgroep

Het bestuur van de PCGG nodigt u hierbij uit voor de:

## ROBOTICA DAG

Iedereen is van harte welkom, je hoeft geen lid van de PCGG te zijn om te komen kijken.

m.m.v. de HCC Robotica

plaats : PV-HOME
datum : zaterdag 5 juli 2003
tijd : 10:00 tot 16:00 uur

website: *http://sigweb.signaal.nl/~pcgg/* email: *pcgg@nl.thalesgroup.com*

# Zaterdag 2 Augustus, Robotica-GG bijeenkomst in Nieuwegein op nieuwe locatie.

Als robotica-GG bestuur zijn we de laatste tijd op zoek gegaan naar een vervangende locatie voor onze bijeenkomst in Gouda. Deze zaal wordt dit jaar door de afdeling Gouda afgestoten en ons rest dan niets dan een andere locatie te zoeken. Nu hebben we in Nieuwegein een buurthuis gevonden, een centrale plaats in Nederland, goed bereikbaar met het openbaar vervoer en ruime parkeerplaatsen aanwezig. Het is buurthuis 't Dok, Hoornseschans 101 te Nieuwegein Zuid. Hier willen we op zaterdag 2 Augustus een robotica-GG bijeenkomst organiseren. Deze begint om 10.00 uur en duurt tot 15.00 uur.

Buurthuis ‘t Dok ligt in Nieuwegein-Zuid afslag Fokkesteeg-N (er is ook een afslag Fokkesteeg, dat is de volgende).
Voor al het verkeer niet komende van af Den Bosch is hier alvast de route beschrijving:
A2 na Ouderijn afslag Nieuwegein. Deze blijven volgen tot je Nieuwegein-Zuid kan volgen. Dan de afslag Fokkesteeg-N. Eerste rechts, bij parkeerplaats links parkeren. Aan het einde van de parkeerplaats schuin tegenover C1000 is t’ Dok, Hoornseschans 101.
Met de sneltram vanaf Utrecht centraal, tram Nieuwegein-Zuid uitstappen bij Fokkesteeg-N. Dan ongeveer 5 minuten lopen. De tram vanaf Utrecht doet er 15-20 minuten over.

**Zie voor een routebeschrijving van de bijeenkomst in Henglo pagina 32.**

# Subsumption architecture

Een nieuw manier van robotprogrammering

Er is een nieuwe manier om robots te programmeren. Deze manier heet de opgeefmethode ("subsumption architecture") en is bij uitstek geschikt om mobiele robots te programmeren. Dit verhaal laat zien waarom de opgeefmethode beter is dan de oude manier van robotprogrammering. Het laat zien hoe de opgeefmethode werkt met een eenvoudige robot met lichtsensors en sonars a la Cybot. Verderop wordt uitgewerkt hoe je de opgeefmethode kunt gebruiken in een C programma.

**Traditionele benadering**

Traditioneel werden robots voorzien van een model van de wereld waarin geometrische details van elk voorwerp was opgenomen, zoals positie en afmetingen. Manipulatorrobots werken door de het oplossen van het pick-and-place probleem: de robot bedenkt een oplossing door te berekenen hoe een voorwerp wordt opgepakt, en waar die wordt neergezet, en berekent alle bewegingsstappen van de grijper die in een lineaire sequentie uitgevoerd moeten worden om de grijper via de meest efficiente weg van de beginpositie naar het eindpositie te bewegen. Om deze opgave te berekenen heeft de robot als invoer een wereldmodel nodig: een geometrisch beeld van de werkelijkheid. Deelprobleem die de robot mogelijk moet oplossen is hoe de grijper een in de weg staande pilaar ontwijkt. Door rekening te houden met de precieze positie en afmetingen van de pilaar berekent de robot stappen voor de omweg die de grijper moet gaan om langs de pilaar heen te gaan en niet er tegenaan te botsen (planning).

Deze benadering van model en planning heeft veel voordelen. Een robot kan op deze manier garanderen dat een beweging kan worden uitgevoerd, en binnen een bepaalde tijd of geeft aan dat de beweging niet kan. Een mobiele robot die op deze manier zo werkt, zal nooit van de trap vallen, in een doodlopende steeg rijden of tegen een voorwerp botsen, omdat het weet heeft van alle voorwerpen in zijn wereld.

Enkele belangrijke nadelen zijn dat er een krachtige computer nodig is om alle geometrische details van een grijperbeweging door te berekenen. Hoe meer obstakels er in het wereldmodel zijn, hoe meer rekenkracht het kost om het bewegings-

# Subsumption architecture

pad te berekenen. Voor vaste robots is het nog mogelijk om een wereldmodel op te bouwen in het geheugen, omdat zijn wereld begrensd is, maar voor mobiele robots die continu in een andere omgeving bewegen is dat haast niet mogelijk. De robot is afhankelijk van de programmeur voor het leveren van de gegevens over de omgeving en dat levert een statisch beeld op. Hedendaagse sensors leveren niet een gedetailleerd genoeg beeld op van de omgeving om een robot zelf een geometrisch precies model van de omgeving op te laten bouwen. Sensors kunnen elkaar bovendien tegenspreken: de sonar geeft misschien een andere afstand aan dan een infraroodsensor. Traditioneel geprogrammeerde robots moeten dan zelf veel berekeningen uitvoeren om te bepalen welke sensors het bij het rechte eind hebben en welke sensorwaarden worden gebruikt ("sensor fusion"). En als laatste nadeel: een robot volgens deze benadering heeft veel moeite om aan te passen aan een veranderende omgeving. Het plannen van een bewegingspad is een sequentieel proces en kost tijd. De robot neemt een beeld van de omgeving, berekent een plan en voert het uit. Als in de tussentijd de omgeving is veranderd, dan mislukt het plan.

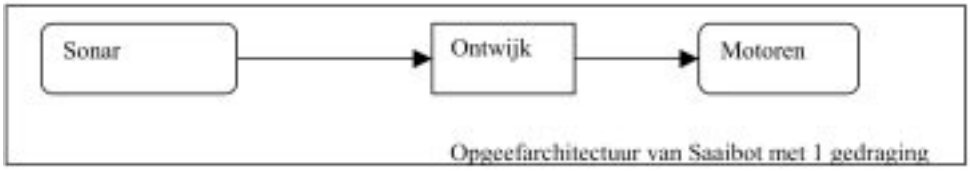
**De opgeefmethode**

Rodney Brooks van het MIT Artificial Intelligence Laboratory heeft een andere manier bedacht: de subsumption architecture. Ik vertaal het maar als 'de opgeefmethode'. Deze methode ondervangt een aantal nadelen van de traditionele benadering. Het idee is dat er een aantal gedragingen zijn opgebouwd uit real time besturingsprogramma's die worden getriggerd door sensors, die ik gedragingen ("behaviors") noem. Deze gedragingen zijn eenvoudige lagen van besturing die allemaal parallel uitgevoerd worden. Met conflicterende sensorwaarden wordt afgerekend door het samensmelten van gedragingen ("behavior fusion"). Er is een arbitrageprogramma aktief die bepaald welk gedraging het meest dominant is. Alle gedragingen kunnen alle sensorgegevens direct lezen. Nergens wordt een gedraging als een subroutine aangeroepen door een ander, maar gedraging draaien parallel naast elkaar. Een dominante gedraging kan wel een lagere gedraging onderdrukken, in die zin dat de uitvoer niet wordt doorgegeven aan de actuator (motor). Omdat elk van deze gedragingen erg eenvoudig is is weinig rekenkracht nodig en werkt dit op een eenvoudige processor met weinig RAM, bijvoorbeeld een 80552 of AVR.

# Subsumption architecture

Als je nu denkt dat je multitasking nodig hebt, dan heb je het mis. Verderop laat ik zien hoe je dit in C kunt oplossen met een paar slimme trucjes.

**Een voorbeeld: Saaibot**
Hier is een voorbeeld hoe subsumption architecture werkt in een simpele robot met motoren, lichtsensors en sonar rondom en een kleine 8-bits processor. Het mag ook infrarood zijn in plaats van sonar. Laat ik het vehikel maar even Saaibot noemen. Saaibot is een levenloos stuk plastic en metaal, maar nu ga ik hem iets spannends laten doen. (elke gelijkenis met bestaande robots berust op ZUIVER toeval)

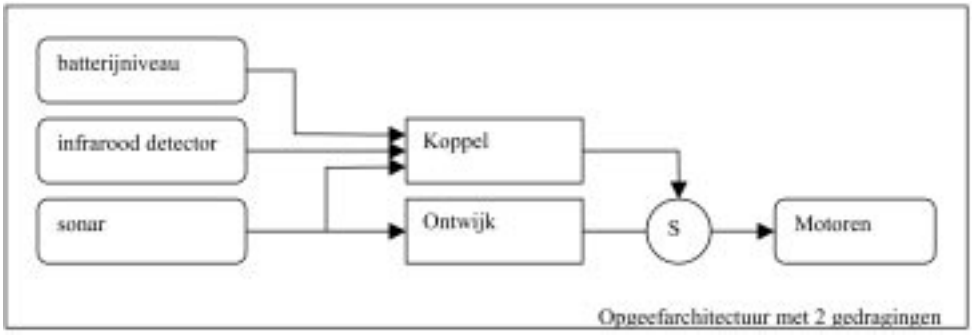


Opgeefarchitectuur van Saaibot met 1 gedraging

Het eerste wat ik Saaibot wil laten doen is obstakels ontwijken. Sonar is een programmamodule of subroutine die de sonars uitleest en de afstand geeft tot objecten. De module Ontwijk leest de invoer die Sonar geeft, en stuurt commando's naar de Motoren. De module Motoren is een routine die stroom stuurt naar de motoren naar gelang de commando's die het ontvangt. Module Ontwijk is een stukje programmacode dat werkt als een reflex: als de voorkantsonar iets dichtbij opmerkt, dan stopt Ontwijk de voorwaartse beweging. Als de achterzijdesonar de kortste afstand meet, dan stuurt Ontwijk het commando vooruit naar Motoren. Als een andere sonar dan de achterkant een afstand meet onder de drempelwaarde, dan draait Ontwijk de robot zodat het object recht achter de robot is. Als alle sonarsensors een waarde onder de drempel (of trigger)waarde afgeven, dan geeft Ontwijk helemaal geen commando's aan Motoren.
Op deze manier heb je een besturing voor een robot die direct reageert, maar die

# Subsumption architecture

op een kleine processor kan worden uitgevoerd. Het gedrag wat hieruit voortvloeit is dat Saaibot altijd een minimum afstand houdt tot elk object in zijn omgeving. Ontwijk is een minimale doelgerichte gedraging.
Saaibot kan nu makkelijk hogere niveaus van doelgericht gedrag bereiken door meer eenvoudige lagen van gedragingen toe te voegen aan de bestaande gedragingen. Ik wil nu Saaibot zelf het laadstation laten opzoeken als de batterijen leegraken. Saaibot heeft daarvoor een batterijspanningsmeter en een infrarood sensor tot zijn beschikking. Het laadstation kan Saaibot vinden door een infraroodbaken die een gecodeerd signaal uitzendt. De module Koppel zorgt ervoor dat Saaiboot de weg naar het laadstation vindt, terwijl het obstakels ontwijkt.

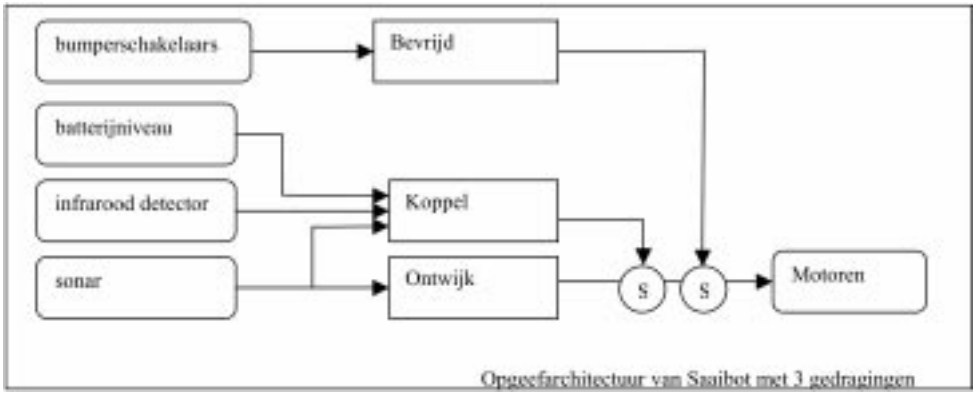


Opgeefarchitectuur met 2 gedragingen

De uitvoer van Koppel en Ontwijk worden verbonden met een onderdrukker ("supressor node") aan Motoren. In de figuur aangegeven met de Cirkel S. De commando's kun je opvatten als berichten door een pijp. De berichten van Ontwijk worden doorgelaten door de onderdrukker alleen zolang er geen berichten van Koppel gestuurd worden. De verbinding met Koppel heeft voorrang of prioriteit.
Saaibot werkt nu zo: Ontwijk is aktief wanneer er een object in de buurt is. Wanneer de batterij leegraakt dan wordt Koppel aktief en zoekt het laadstation met behulp van sonar en infrarood. Koppel leidt Saaibot naar het laadstation met behulp van de sonargegevens en infrarood, maar onderdrukt Ontwijk om te voorkomen dat Saaiboot wegrijdt van het laadstation.

# Subsumption architecture

Als ik meer gedragingen wil toevoegen, dan is dat heel makkelijk. Zou ik merken dat de infrarood sensors niet alle objecten opmerken, wat ook vaak gebeurt (bijvoorbeeld omdat de straal te smal is), dan kan ik linksvoor , middenvoor en rechtsvoor bumperschakelaars op Saaibot monteren. Ik maak een gedraging genaamd Bevrijd die de bumperschakelaars afleest en ervoor zorgt dat Saaibot terug rijdt als hij ergens tegenop knalt. Omdat de bumperschakelaars als laatste reageren van alle sensoren wanneer Saaibot op iets dreigt te botsen, heeft Bevrijd de hoogste prioriteit. Ik ga er nu maar even vanuit dat het laadstation nooit tegen de bumpers zal drukken. Ik zeg dus eigenlijk tegen Saaibot: als je botst rij dan ALTIJD terug. Module Bevrijd onderdrukt daarom de berichten van alle andere gedragingen en heeft het laatste woord.
Merk op: Het gaat anders wanneer Saaibot tegen het laadstation kan botsen en de bumpers aktief worden. Als dat kan, dan zou Koppel in staat moeten zijn om ook berichten van Bevrijd te kunnen onderdrukken! Nadenkertje: hoe moeten de leidingen dan verbonden worden?



Opgeefarchitectuur van Saaibot met 3 gedragingen

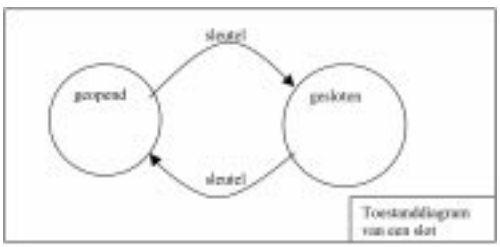
Door zijn gedragingen is Saaibot een robot die snel reageert op omstandigheden uit zijn omgeving, terwijl Saaiboot toch geen compleet beeld nodig heeft van de wereld. Gedragingen zijn dus net als reflexen. Je ziet dat Saaibot ‘slimmer’ gewor-

# Subsumption architecture

den is door een gedraging toe te voegen. Saaibot is toch niet langzamer geworden, omdat beide gedragingen tegelijkertijd worden uitgevoerd. Dit is een simpel voorbeeld, maar het aantal gedragingen zou nog verder uitgebreid kunnen worden en daardoor kan Saaibot nog slimmer worden. Ook kunnen er meer dan één soort actuator gestuurd worden: lampjes, luidspreker, enz. Je zou bijvoorbeeld een grijperarm kunnen monteren op Saaibot, en een gedraging maken die alleen de robotarm bedient.
Mooi, maar hoe doe ik dat nu precies?
Of om het met een mooi woord te zeggen: de implementatie. De opgeefmethode gaat ervan uit dat alle gedragingen parallel uitgevoerd worden. Dit kun je dus goed doen op een besturingscomputer met multitasking. Elk gedrag kan worden uitgevoerd in een eigen proces. Een scheduler geeft aan elk proces een beetje processortijd, waardoor het lijkt alsof elk proces gelijktijdig uitgevoerd wordt, maar in werkelijkheid worden ze één voor één uitgevoerd. Zo zorgt ook Windows ervoor dat Word, Outlook en Explorer tegelijk worden uitgevoerd.

Maar op mijn robot heb ik helemaal geen Windows, dus wat nu? Je krijgt dit voor elkaar als je hoofdprogramma een oneindige lus uitvoert in een regelmatig tempo. In deze lus worden alle gedragingen achter elkaar uitgevoerd. Je moet ervoor zorgen dat elke gedraging snel doorlopen wordt, zodat alle gedragingen regelmatig aan de beurt komen. Elke gedraging wordt opgebouwd als een eindige toestandautomaat of ETA ("finite state machines"). Tenslotte wordt een scheidsrechterprogramma uitgevoerd die bekijkt welke gedraging zijn uitvoer aan de Motoren mag doorgeven.

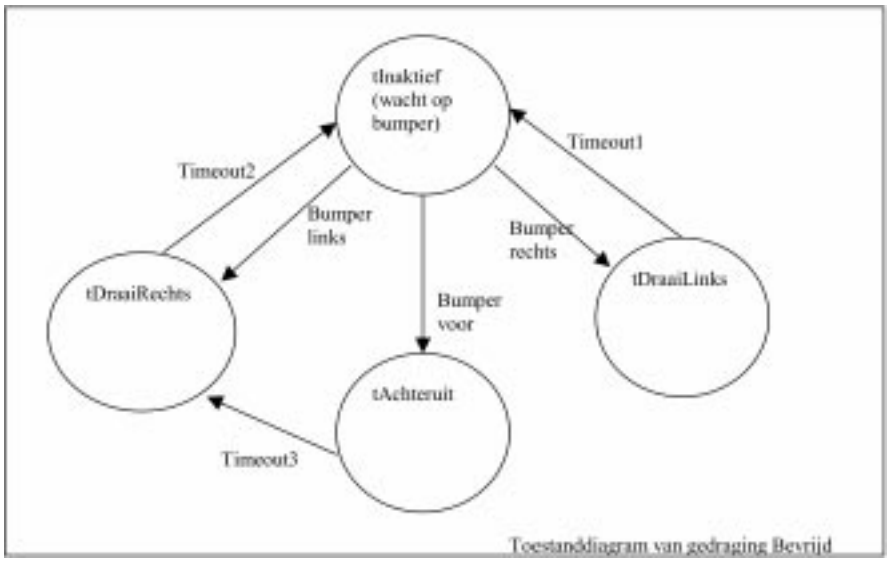
Een ETA kun je beschouwen als een ding dat zich in verschillende toestanden kan bevinden. Door een gebeurtenis kan de ETA van een toestand naar een andere worden gebracht. Een voorbeeld van een eenvoudige ETA is een slot: hij is altijd



Toestandsdiagram van een slot

# Subsumption architecture

gesloten of geopend. Een sleutel brengt het slot van geopend naar gesloten, of andersom. Functies als snoepautomaten worden ook beschouwd als ETA's.




De gedraging Ontwijk van Saaibot kan beschouwd worden als een ETA met 4 toestanden.
1. Inaktief.
2. DraaiRechts.
3. DraaiLinks
4. Achteruit.

In C krijg je dan de volgende functie.

```
void  Bevrijd(void)
{
   switch(toestand)  {
```

# Subsumption architecture

```
  case tInaktief:
    tijd-in-toestand = 0;
    if(Bumper() == ContactLinks)
      toestand = tDraaiRechts;
    else if(Bumper() == ContactRechts)
      toestand = tDraaiLinks;
    else if(Bumper() == ContactVoor)
      toestand = tAchteruit;
   else
      Bevrijd-aktief = 0;

  case tDraaiRechts:
    Bevrijd-aktief = 1;
    if(tijd-in-toestand > timeout1)
      Toestand = tInaktief;

  case tDraaiLinks:
    Bevrijd-aktief = 1;
    if(tijd-in-toestand > timeout2)
      toestand = tInaktief;

  case tAchteruit:
    Bevrijd-aktief = 1;
    if(tijd-in-toestand > timeout3)
      toestand = tDraaiRechts;
  }
}
```

Hier vind je nog een andere truc: het gebruik van timers. De variabele tijd-in-toestand wordt automatisch door de processor regelmatig opgehoogd (meestal verzorgd een interruptroutine dit klusje). Deze variabele wordt gereset wanneer de ETA in de inaktieve toestand is, en wordt opgehoogd wanneer de ETA in een

andere toestand komt. Hiermee is de tijd één van de triggers geworden die ervoor zorgt dat een ETA verandert van toestand! (Men spreek ook wel van een uitgebreide eindige toestandautomaat of "augmented FSM")
Omdat we hier niet te maken hebben met multitasking, kunnen we niet gewoon een keiharde vertraging inbouwen zoals bijvoorbeeld sleep(0.5) of milliseconden(500) , want de processor moet ook nog andere gedragingen kunnen afwerken. Dit is belangrijk om in gedachten te houden. Door regelmatig te testen of de tijdsvariabele hoog genoeg is, kun je op deze manier een vertraging inbouwen of tijdsduur afpassen, zonder de processor op te houden.

Een scheidsrechterprogramma bepaalt welk gedraging zijn commando's doorgeeft aan de motoren. Als de prioriteiten vastliggen zoals in Saaibot, kan dat er zo uitzien in een C programma.

```
void  Scheidsrechter(void)  {
   if(Ontwijk-aktief)
     Motoren-invoer  =  Ontwijk-uitvoer;
   else  if(Koppel-aktief)
     Motoren-invoer  =  Koppel-uitvoer;
   else  if(Bevrijd-aktief)
     Motoren-invoer  =  Bevrijd-uitvoer;
  else
     Motoren-invoer  =  MotorenVooruit;
}

void  Schedule  (void)  {
   LeesSensors();
   Ontwijk();
   Koppel();
   Bevrijd();
   Scheidsrechter();
   StuurMotoren();
}
```

# Subsumption architecture

De LeesSensors() aanroep voert een routine uit, die alle sensorswaarden inleest en opslaat in variabelen, die door alle andere subroutines kunnen worden gelezen. Het is vooral handig om tijd te besparen, als er bijvoorbeeld sensorgegevens in te lezen zijn uit een schuifregister, of I2C-bouwstenen of andere handelingen die tijd kosten eenmaal per verwerkingcyclus te laten doen.

Vervolgens zorgen we ervoor dat in het hoofdprogramma de Schedule functie regelmatig uitgevoerd wordt. Het is dus belangrijk om ervoor te zorgen dat in de gedragingen geen langdurige bewerkingen gedaan worden.

```
void main(void) {
   initialisatie();
   while(1) {
      schedule();
     wacht-op-tick()
   }
}
```

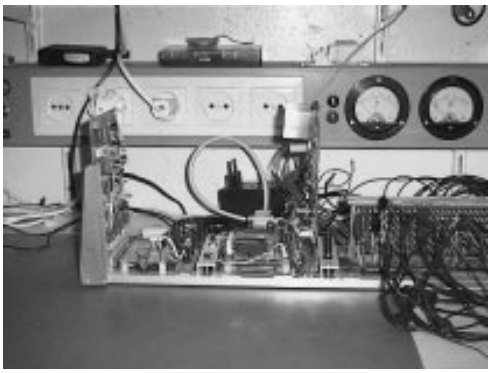
Ik hoop dat ik zo duidelijk heb kunnen maken dat de opgeefarchitectuur aandacht verdient omdat het een aantal voordelen bevat boven andere manieren van programmering. Bovendien biedt de opgeefarchitectuur de mogelijkheid om de functies van een robot flink uit te breiden, met behoud van een snelle reactietijd.

**Bronnen & Literatuur**

De subsumption architecture wordt op veel plaatsen gebruikt: begin 90'er jaren op de Rug Warrior, en Handyboard, en ook in de beeldprogrammeertaal Robolab voor de LEGO-Brick, zie Elektuur juni 2003.
Joseph Jones, Anita Flynn - Mobile Robots Inspiration to Implementation.
Elektuur juni 2003
Enkele links naar publicaties van de oorspronkelijke bedenker kun je vinden op mijn homepage http://home.hccnet.nl/p.c.wiegmans onder kopje Subsumption.

Wie kan mij een betere vertaling geven van 'subsumption architecture' ?
A.U.B. geen wachtwoorden zetten op kopieen van deze tekst!

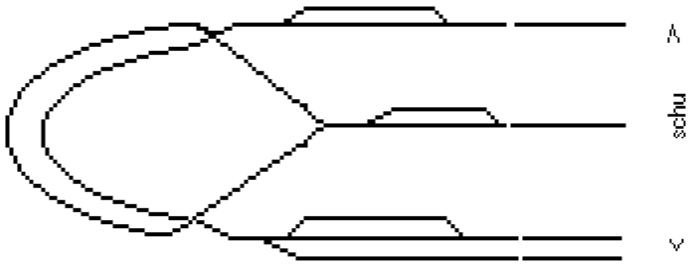
# Trein besturing



Ik hen het B+ bord in een bestaande modelspoorbaan toegepast. Door ruimte gebrek heb ik een baan (zie boven) met een eind station. Aan het eind station heb ik nu een deel aangebouwd dat zich als een schuif beweegt. Hierdoor kan een loc zich van het ene spoor naar het andere spoor begeven.

**Wat heb ik nu precies gedaan**

Ik heb een kistje gemaakt met daarin een stappenmotor, die via tandwielen een spindel aandrijft. Aan de zijkanten van het kistje zitten twee ladegeleiders, waarop het plateau zit bevestigd met de rails, waarop de trein voor verplaatsing kan rijden. Onder het plateau zitten twee hoekprofieltjes die over de wandelmoer vallen van de spindel waardoor het plateau wordt voortbewogen. Het kistje is met een instelbaar frame aan de baan gemonteerd zodat de rail van het kistje op gelijke hoogte van de baan is in te stellen.




**Hoe werkt het**

Na het inschakelen van de voeding en nadat het programma geladen is, wordt eerst het plateau in de juiste positie geplaatst. Daarna wordt pas de baan vrij gegeven Een loc rijdt met vertraagde snelheid het plateau op, in de rails op het plateau zit een read contact dat de trein laat stoppen, en de rij spanning af schakelt. De stappenmotor krijg dan opdracht het plateau te verplaatsen naar een ander spoor. Is het plateau bij het juiste spoor aangekomen dan wordt de rij spanning

# Trein besturing

weer in geschakeld en rijdt de loc weer van het plateau af.
Wanneer de loc weer in de baan is aangekomen krijgt het plateau opdracht terug te gaan naar zijn positie De loc rijdt nu een rondje en komt weer bij het eind station aan. Echter via een wissel komt hij ook op een ander perron aan. Na de wachttijd rijdt hij met een vertraagde snelheid het plateau op. En de loc wordt weer naar een ander spoor verplaatst. Ook de baanvakken worden door het B+ Bordje gecontroleerd. Readcontacten geven aan in welk baanvak de loc zich bevindt, een achterliggent baanvak is altijd uitgeschakeld
Gelet op de beperkte ingangen kan niet alles door het B+ bordje geregeld worden en bestuurd. Het was voor mij interessant om te zien wat dit bordje met alle vervuiling van de stoorpulsen die een treinbaan veroorzaakt doet.
Tot op heden hebben zich geen problemen voor gedaan. Het is echt een perfect besturings bordje. Nu ben ik van plan om met een AVR bordje hetzelfde te doen.

Ruud
Verweijen.

# De computer-gestuurde modelbaan

**Het blokgestuurde HCCM-systeem**

Het belangrijkste kenmerk van het HCCM-systeem is dat het een digitaal blok-gestuurd systeem is, waarbij de modelbaan in blokken ingedeeld wordt die steeds apart kunnen worden aangestuurd vanuit de computer. Dit in tegenstelling tot de z.g. Digitaalsystemen waarbij treinen individueel worden aangestuurd. Belangrijk daarbij is ook dat er bij tweerailsystemen niets in de locomotieven veranderd of ingebouwd behoeft te worden. Bij de schalen Z en N is dat een groot voordeel. Bij drierailsystemen (m.n. Märklin) is het alleen nodig van de wisselstroommotor een gelijkstroommotor te maken, wat eenvoudig, goedkoop en bovendien zeer snel herstelbaar is. Het systeem is bruikbaar voor alle denkbare schaalgroottes; van Z, N, h0 t/m LGB.

In de loop der tijd is het hardware-concept van blokbesturing met terugmelding steeds verder verbeterd. In principe gaat het om een zelfbouwsysteem, printplaten en losse onderdelen zijn bij de HCCM te verkrijgen.
Naast de hardware is ook software beschikbaar. Samen biedt het een complete oplossing voor een goed werkend geautomatiseerd modeltreinbedrijf.
Dit artikel geeft een korte beschrijving van het systeem. Uitgebreide informatie staat in het handboek 'De computer-gestuurde modelbaan' dat bij de HCCM te koop is.

# De computer-gestuurde modelbaan

**Opbouw van het systeem**

Aan het systeem zijn enkele eisen gesteld, te weten;
# geschikt zijn voor modeltreinen en -rails van alle merken en in alle schalen;
# aan de treinen en de rails mag niets worden gewijzigd;
# geen beperkingen aan het sporenplan van de modelbaan, ook niet voor wat betreft de exploitatie;
# aansluitbaar zijn op alle merken en typen computers;
# wijziging van de modelbaan moet eenvoudig kunnen worden uitgevoerd;
# betaalbaar zijn.

Het HCCM-systeem voldoet hier in ruime mate aan.


**Hardware**

De hardware van het systeem heeft drie te onderscheiden hoofdonderdelen, t.w.:
# De modelbaan (M)
# Het interface (I)
# De computer (C)

Deze onderdelen moeten om treinen te laten rijden met elkaar 'communiceren'.

De functie van het interface is vooral het geschikt maken van de signalen uit de computer voor de modelbaan (treinen, wissels en seinen) en omgekeerd.
De modelbaan is voor iedere gebruiker volgens zijn eigen wens te ontwerpen, maar er dient bij de bouw wel rekening gehouden te worden met de toepassing van het HCCM-systeem.
De computer kan een willekeurige PC zijn van XT tot Pentium. In de huidige situatie wordt ervan uitgegaan dat het meestal toch een PC zal zijn met Windows95 of nieuwer.

# De computer-gestuurde modelbaan

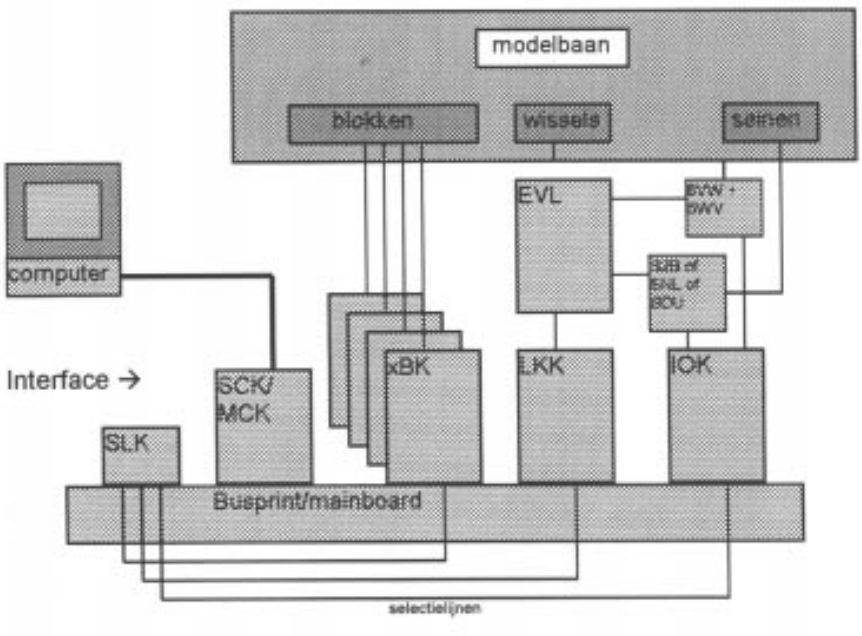
**Opbouw van het interface**

Het interface bestaat uit één tot vier 19-inch-rekken, elk met een busprint(mainboard), één (seriële) multicentrale kaart (MCK), één selectiekaart (SLK) per rek, een aantal blokkaarten (max. ca. 60), een latchkeuzekaart (LKK) en/of facultatief een input-outputkaart (IOK).

Onder de baan bevinden zich de tot het interface behorende enkelvoudige latchkaarten(EVL), de wisselkaarten(6VW) en -versterkers(5WV) alsmede de seinkaarten (S2B en/of SNL of SDU). De omvang van het interface is afhankelijk van de grootte van de modelbaan en in het bijzonder van het aantal blokken.

**Bloksysteem**

De baan wordt in blokken verdeeld, waarbij ieder blok de rijspanning ontvangt van een blokkaart in het interface. In het systeem wordt onder een blok verstaan een stuk spoor dat volledig elektrisch is gescheiden van de andere sporen. Meestal



# De computer-gestuurde modelbaan

is het vergelijkbaar met het echte spoorwegbedrijf maar in complexe wisselstraten komen extra zogenoemde doorrijblokken voor. Op deze manier kunnen de treinen in de verschillende blokken volledig onafhankelijk van elkaar door de computer worden bestuurd.
De blokkaarten zorgen ervoor dat de rails van een pulsvormige rijspanning wordt voorzien; de snelheid kan daarbij in zestien stapjes vooruit en achteruit worden geregeld.

**Koppeling met de computer**
De blokkaarten worden in een 19-inch-rek gestoken. Aan de achterkant zorgt een busprint voor de koppeling met de centrale kaart(SCK of MCK). Op zijn beurt is die via een RS232-kabel met de computer verbonden, waarover tenminste vier maal per seconde alle blokkaarten een snelheidsinstelling ontvangen en waarover van iedere blokkaart de bezetmeldingen worden teruggemeld.

**Bezetmeldingen**
Ieder blok wordt elektrisch volledig geïsoleerd van de naastliggende blokken. De spanning wordt aan de linker spoorstaaf toegevoerd, terwijl de andere spoorstaaf in maximaal vier secties wordt onderverdeeld. De blokkaart meet of via de bedrading naar een blok stroom loopt en zet dit om in een signaal naar de computer. De computer krijgt zo van elke blokkaart een 4-bits bezetmelding. Een bitje '1' betekent dat in de betreffende bloksectie een stroom loopt. Per blok kan zodoende bijna exact worden bepaald waar de trein zich in dat blok bevindt.
Als de trein voor een rood tonend sein moet stoppen wordt de snelheid door het programma geleidelijk aan verminderd totdat de trein in de laatste sectie van het blok is aangekomen, waarna hij voor het sein tot stilstand komt.
De assen van alle wagens worden over de isolatie geleidend gemaakt met weerstandslak. Een weerstand van 10 kOhm per asje is daarvoor voldoende. Omdat op deze manier de gehele trein gedetecteerd wordt, kan zonder problemen met bijv. trek/duwtreinen worden gereden. Ook ontdekt de computer of er wagens door een trein verloren worden.

# De computer-gestuurde modelbaan

**Keerlussen**
Bij tweerail-systemen vergt een keerlus speciale voorzieningen. Het HCCM-systeem heeft daar helemaal geen moeite mee, het getekende baanplan wordt zonder enig probleem door onze treinen doorlopen.

**Wissel- en seinaansturing**
Het assortiment biedt diverse kaarten voor wissel- en seinaansturing.
De huidige software is geschikt voor het Duitse en Nederlandse seinstelsel.

**Software HCCM-programma**
Een computer is in feite een dom ding, hij doet alleen wat hem opgedragen wordt. Het programma bevat de procedures voor alle denkbare situaties die op de baan kunnen voorkomen, die hoeft u zelf niet meer te schrijven. Dit programma is geschreven in de taal 'Forth' en werkt onder het besturingsprogramma DOS.

Maar het programma weet niet hoe de modelbaan is ingedeeld en hoe de treinen zich op de baan moeten gedragen. Dit zal de gebruiker zelf met gebruikmaking van het handboek moeten bedenken en door het invullen van de baan- en treinadministratie aan het programma moeten toevoegen. Pas als ook al die gegevens correct zijn opgegeven kan de computer de baan zonder problemen besturen.
Bij het opstarten van het programma scant de computer alle blokkaarten een keer af en weet daarna waar zich treinen bevinden. Vervolgens kan de gebruiker via het toetsenbord aangeven welke treinen dit zijn. Daarna geeft hij alle treinen of een selectie daaruit opdracht te gaan rijden. De computer zorgt er daarna voor dat er voor alle gestarte treinen vrije blokken worden gereserveerd, waarna de treinen automatisch gaan rijden als er vrije blokken beschikbaar zijn. De snelheid is vanaf het toetsenbord aan te passen. Het is mogelijk om met een functietoets een trein van automatisch rijden naar rangeren om te schakelen.

**Programma Koploper**
Aan het begin van 2000 is het programma Koploper van Paul Haagsma beschikbaar gekomen voor gebruikers van een modelbaan volgens het HCCM-systeem.

# De computer-gestuurde modelbaan

Dit programma is gemaakt voor gebruik onder tenminste Windows95. Er is dus een modernere computer voor nodig. Oorspronkelijk was het programma gemaakt voor het treingestuurde Selectrix-systeem. Het is nu naast Selectrix en HCCM-systeem ook geschikt voor de treingestuurde systemen Märklin Digital, Edits, Intellibox, Fleischmann Twin-Center en Lenz. Het uiterlijke verschil ten opzichte van het HCCM-programma is dat in plaats van tabellen op het scherm grafische beelden met de weergave van het sporenplan en met schermpjes met treingegevens te zien zijn.
Het programma wordt steeds verder ontwikkeld. De functionaliteit is nu tenminste gelijk aan de laatste versie van het HCCM-programma. Om met dit programma te kunnen gaan rijden, moeten ongeveer de zelfde soort handelingen worden verricht om de baan- en treingegevens in te voeren. Het gebeurt alleen op andere manier.
Het programma Koploper is als freeware te vinden op de internetsite "http://www.pahasoft.nl". De ondersteuning voor HCCM-gebruikers wordt gedaan door HCCM-leden via de mailinglist 'hccm@yahoogroups.com', waarvoor u zich kunt aanmelden door een e-mailbericht te sturen aan
HCCM-subscribe@yahoogroups.com.

**Informatie**
Naast deze informatie is er ook nog een algemene folder (A) over de HCCM en een specifieke folder over digitale treinbesturing met fabriekssystemen (D).

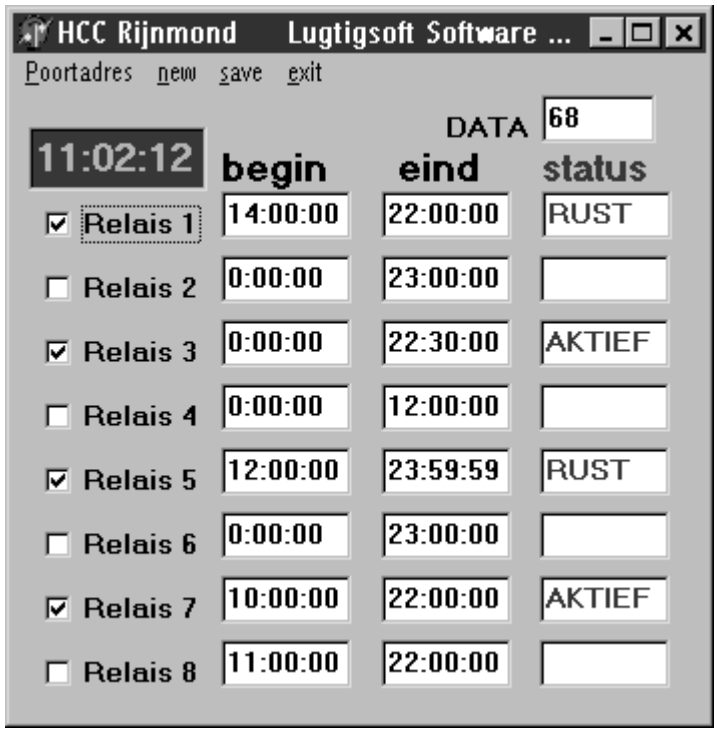
Meer informatie is te vinden op internet op de adressen:
http://www.hccm.nl
http://www.hcc.nl
http://www.pahasoft.nl

Voor overige vragen is het adres van de secretaris:
Karel Dirks, M. Mooystraat 8, 1759 XK CALLANTSOOG
Telefoon: (+31) (0)224 581703 E-mail: c.k.dirks@hccnet.nl
- In het voorjaar 2003 wisselt het secretariaat.
Kijk op http://www.hccm.nl voor het nieuwe adres.

# Relais besturing

**De ontwikkeling**
Iemand had een printje met wat relais en een aanstuur IC wat op een ULN2803 lijkt gemaakt, zodat de aansturing eenvoudig is. Er zijn twee soorten schakelingen voor aan de LPT poort, direct en met een strobe (deze geeft aan wanneer de data geldig is). Ik heb gekozen voor de directe aansluiting (strobe wordt niet gebruikt). Het is geschreven in DELPHI, want daar kan je van de ingebouwde assembler gebruikmaken. Dit is werkelijk een groot voordeel.

**POORT ADRES**
We beginnen met het selecteren van het poortadres, waar we de data naartoe sturen, het is het eerste Item in de menubalk.




**De klok**
Deze regelt het hele programma. De belangrijkste gebeurtenis is onTimer die de checkbox van ieder relais checked. Indien er een kruisje staat dan begint een andere lus, anders gaat hij naar het volgende relais.

**Het algorithme**
We kijken eerst of het relais aangevinkt is. Indien het relais aangevinkt is, dan wordt gekeken of de actuele tijd gelegen is in het interval tussen begintijd en eindtijd. Er wordt dan een status gegeven en als het

# Relais besturing


relais geactiveerd moet worden, een bit geset. Al die bit's tezamen vormen de databyte welke zichtbaar is.

**De code**

Dit is een stukje assembler-code, die de inhoud van de poort variable in register DX stopt en het databyte in register AL. Er volgt dan een out instruktie, er wordt geschreven naar outputpoort DX en wel de waarde AL. Merk ook op de // voor een instruktie, dit is gedaan om de instruktie te veranderen in commentaar, hij heeft dan geen invloed meer.

```
asm
MOV DX,poort;
MOV AL,datout;
OUT DX,AL;
//   int $17;
end;
```

Hans Lugtigheid

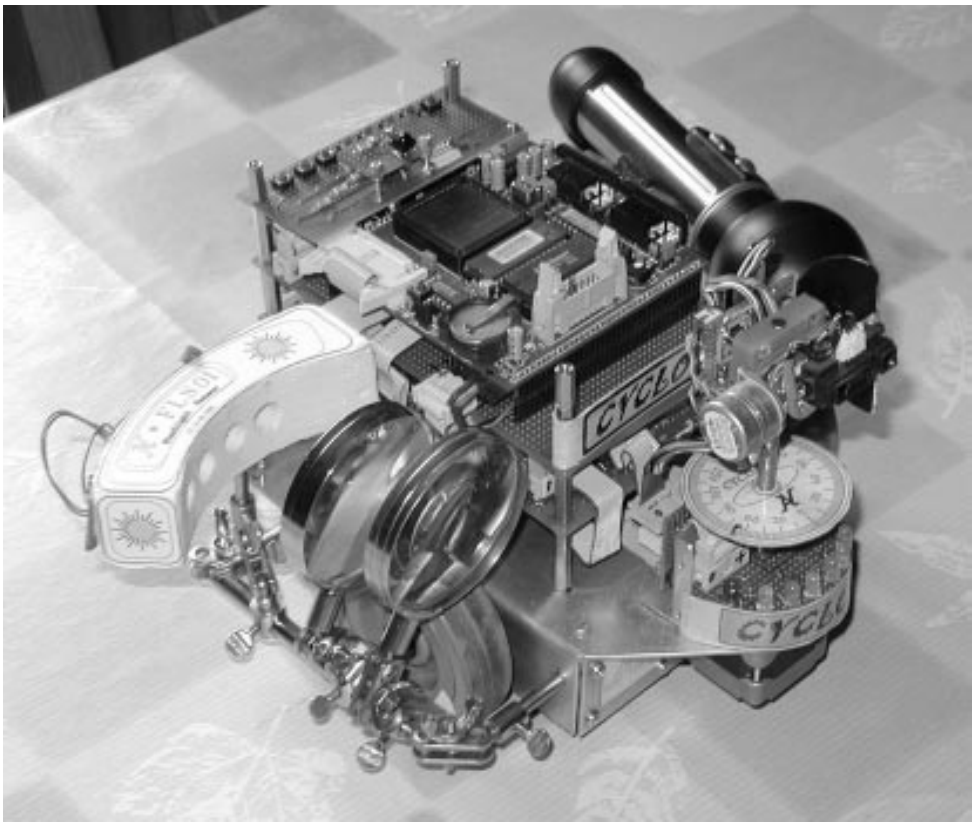
# Cybot

Binnen de robotica-GG is het Cybot-virus losgebroken. Discussies van het nut van een bipolaire condensator op de motorstuurprint tot het komen tot een gezamenlijk cybotproject voor de HCC dagen dit najaar. Het wordt in de discussie groep van de robotica besproken. Op de clubbijeenkomsten is er de nodige aandacht voor.Ook is er een zeer goede site van een club-genoot:

**http://**
**www.cybot.buiskool.net**

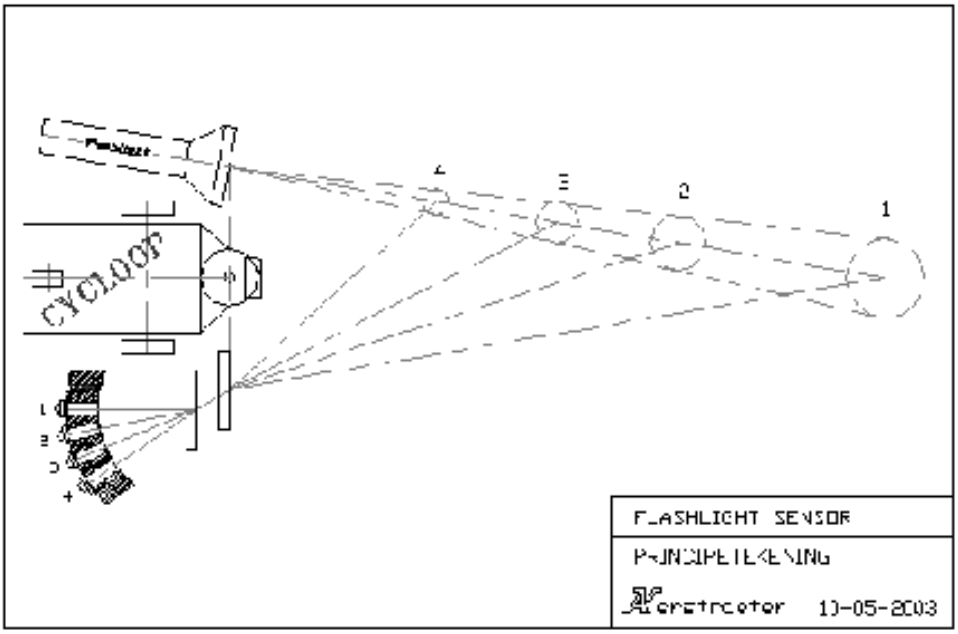
# flashlightsensor

Onderstaand verhaal is een vrije interpretatie van ondergetekende over een gezonde discussie in een actieve club, en heeft dan ook geen enkele juridische of wetenschappelijke waarde en is daarom ook niet bedoeld als oplossing voor het betreffende discussiepunt.
Iedereen die regelmatig Gouda bezoekt en/of de HCCrobotica mailing-list leest, zal ongetwijfeld iets vernomen hebben over de discussie over het werkingsprincipe van de Sharp GP2D12 en aanverwante type sensoren. Het is niet de bedoeling om hier die discussie over te doen, maar in het kort kwam het hierop neer: werkt deze op licht(sterkte) meting of is het een driehoeksmeting. Zelf heb ik ook enkele van deze sensoren gebruikt en ben daar redelijk tevreden over, indien er zich proble-



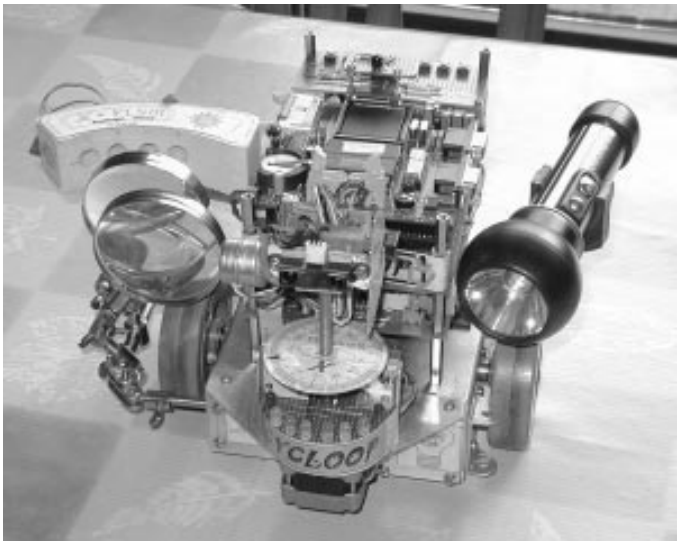
# flashlightsensor

men voordoen hebben die meestal meer te maken met mijn eigen beperkingen dan met de beperkingen van de sensor. Om ze toe te passen is het lezen van de specificaties voldoende, hoe ze functioneren is een andere vraag. Maar ondanks mijn onwetendheid over het functioneren, heb ik toch een blik geworpen en een oor te luister gelegd in beide kampen. Cees Nobel had ook het één en ander opgevangen en naar aanleiding hiervan heeft die met Fischertechnik een meetopstelling in elkaar gezet. Deze meetopstelling ( opgesteld in Gouda) heeft aangetoond dat de waarden van het bewuste "figuur 8" wel goed waren. Hieruit bleek dat de I.R. lichtspot ongeveer een diameter van 30 mm heeft, waardoor driehoeksmeting volgens sommigen niet zo voor de hand lag. Op weg naar huis, ik heb ruim 1.5 uur tijd om het één en ander te overdenken, begon ik mij in dat verhaal te interesseren. Mijn gedachten gingen naar het feit dat je met een vergrootglas het zonlicht kan opvangen en het brandpunt op een voorwerp kunt laten schijnen. Indien aan het vergrootglas een schietlood en een kompas waren bevestigd zou je hiermee de stand van de zon kunnen bepalen. Ik denk dat de zeerovers ook al zoiets gebruik-



# flashlightsensor

ten. Maar ik moet toegeven dat ik op dat moment twijfelde of je door met een lichtbron te schijnen op een voorwerp het gereflecteerde licht weer zou kunnen opvangen met een vergrootglas. Thuis gekomen heb ik direct uit de ene schuif de zaklamp, en uit de andere schuif het vergrootglas genomen om hiermee een test te doen. Het resultaat was boven alle verwachting, ondanks dat de conditie van de batterijen niet meer zo goed was. Het leek me leuk om hiervan op een houten plankje een proefopstelling te maken. Met deze opstelling kon je goed zien hoe het brandpunt zich verplaatste indien het voorwerp dichter bij de lichtbron werd gehouden. Bij deze opstelling rees de vraag of het mogelijk zou zijn om met het geprojecteerde licht ook een LDR te activeren, en daar enkele LED's mee aan te sturen. Ik had er namelijk nog enkele LDR's liggen uit wat oude 5" floppydrives. Na wat experimenteren was de geactiveerde LDR zichtbaar doormiddel van de daaraan gekoppelde LED. Aangezien het resultaat zo goed was kwam het idee om deze constructie op een robotwagentje te plaatsen. Een paar jaar geleden had ik mijn cycloop wagentje gebouwd met de bewuste GP2D12 sensor op een twee stappenmotoren systeem. Het is mooi om de originele sensor samen met de surrogaat sensor op één wagentje te plaatsen. Er was wel een probleem aangezien mijn testopstelling met één vergrootglas was gebouwd, daardoor lag het brandpunt +/- 250 mm achter het vergrootglas. Dit was te ver om een mooie opbouw te kunnen realiseren. Uit de fotografie weet ik dat je met meerdere lenzen wel het één en ander kan aanpassen. Na wat prutsen met een tweede vergrootglas kon ik het brandpunt



# flashlightsensor


op +/- 100 mm achter het vergrootglas projecteren. Deze afmetingen waren aanvaardbaar voor de opbouw. Na diverse opmetingen van mijn testopstelling met het opstakel in verschillende posities, heb ik een tekening gemaakt om de behuizing in piepschuim te bepalen, waar de vier LDR's op bevestigd zijn. Eigenaardig genoeg bleken deze afstanden ongeveer overeen te komen met GP2D12 namelijk tussen de 15 en de 70 cm. Het geheel werkt goed als er geen felle lichtbron in de omgeving is, zoals direct zonlicht of een spot. Maar nogmaals ik wil helemaal niet beweren dat mijn creatie de werking van de GP2D12 verklaart, ik ken enkel de bij de sensor meegeleverde specificaties en heb er ook nog geen één open gemaakt. Het is gewoon het resultaat van wat fantasie over hetgeen ik gezien, gehoord en gelezen heb. Het leuke hieraan is dat het nog werkt ook. Het werkingsprincipe van mijn ontwerp zou ik omschrijven als een stokoude optische technologie voorzien van een digitaal jasje :-). Of is het toch meten met licht dat in driehoeken wordt geprojecteerd?

Aloys Verstraeten.

# Safety Trophy 2003

Lego-robot wint Safety Trophy 2003

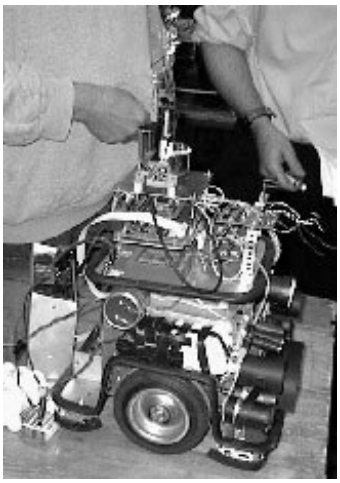
Paul van Dijk en Hans Stekelenburg, twee Nederlandse software-ontwikkelaars, sleepten kortgeleden met hun robot "Keen Machine Small" de overwinning in de wacht van de eerste Melexis Safety Trophy.
In deze wedstrijd kwam het erop aan om met een autonome robot (zonder afstandsbediening) een hindernissenparcours af te leggen. Het omverrijden van tennisballen leverde bonuspunten op en het omverrijden van theelichtjes en soepblikken strafpunten. Hun Lego robot die ze aanpasten en voorzagen van enkele extra sensoren en specifieke software, haalde het dankzij zijn erg eenvoudig maar doeltreffend concept en een tikkeltje geluk op het juiste moment.

Hun strategie bestond erin om met een erg klein robotje zo snel mogelijk naar de eindbakens toe te rijden. Ze hadden het wagentje bovendien zo geprogrammeerd dat het onderweg elk obstakel zou ontwijken. Dat de robot in de finale toch enkele tennisballen omverreed, leverde hen verdiend de overwinning en de gloednieuwe Citroën C3 op.
Als tweede eindigde de robot van Stijn Goffin en Steven Geussens, twee ingenieurstudenten aan de Katholieke Hogeschool Limburg. Hun robot was net iets trager en maaide bovendien één blik meer tegen de grond.

Naast de winnaar viel ook de meest innovatieve robot in de prijzen met een driedaags

# Safety Trophy 2003

beursbezoek aan een elektronicavakbeurs in de VS. De jury koos unaniem voor "Viktor de steenezel", de robot van Willem Maes, omdat hij alle objecten kon waarnemen en daarenboven ging de robot ook actief op zoek naar de lucratieve tennisballen.
Hoewel hij de beste was tijdens de kwalificatieronde werd hij mede door pech uitgeschakeld in de kwartfinale door de Franse robot MAHAR.
Steven Peleman en Thomas De Smet werden beloond met de pechvogelprijs omdat hun wagen door een blokkerend wiel de strijd moesten staken. Zij krijgen VIP tickets voor de GP Formule 3000 in Hockenheim en tickets voor de GP Formule 1.



http://www.trophy.melexis.com/

# [KeenBots Team]

Het KeenBots team bestaat uit 6 enthousiaste robot hobbyisten, die allen als collega's werkzaam zijn in de IT. Deze club houdt zich in haar vrije tijd bezig met het bouwen van autonome robots. Als team zijn ze steeds op zoek naar nieuwe uitdagingen en gaan deze ook graag aan. De Safety Trophy is zo 'n uitdaging waar het KeenBots team zich prima in kan vinden. Het meedoen is dan ook belangrijker dan de winst. Maar dat neemt niet weg dat het team natuurlijk haar uiterste best gaat doen om bovenaan te eindigen.
Bijzonder aan dit team is dat ze met een robot van Lego meedoen. Waarom Lego? Omdat de constructie zeer makkelijk is aan te passen. Verder zijn er veel standaard sensoren te krijgen voor de Lego microcontroller, zodat er op het algoritme geconcerteerd kan worden i.p.v. op de altijd lastige hardware.

# Robotica-GG op Internet

De HCC Robotica-GG op Internet.

Ook wij kunnen niet achterblijven, bezoek onze website en meld je met projecten aan om deze ook op onze clubsite te zetten.

**http://www.robotica.hccnet.nl/**

# Route beschrijving Hengelo

## De route beschrijving voor de bijeenkomst op zaterdag 5 Juli te Hengelo is als volgt:

Komend vanuit het westen via de A1, het noorden via Almelo of het zuiden via Enschede kies de A35 en houd Hengelo Zuid aan. Bij afrit 27, industrie terrein Twentekanaal, verlaat u de snelweg. Komend vanuit het westen/noorden gaat u bij de stoplichten links, onder de snelweg door en bij de stoplichten daarna recht door. Verkeer komend vanuit Enschede gaat bij dit stoplicht rechts.
Houdt vervolgens rechts aan en ga bij het stoplicht rechtsaf de Diamantstaat in. U kunt rechtsafslaan zonder voor een stoplicht te stoppen. Neem op de Diamantstraat de eerste afslag links. Na ongeveer 250 meter maakt de weg eerst een bocht naar links en dan weer naar rechts, u krijgt een zijstraat van links, kort daarna gevolgd door nog een bocht naar rechts. Meteen na deze bocht licht de ingang van het parkeer terrein. Rijdt helemaal door totaan hek en parkeer rechts. Het linker gebouwtje is een gebouw van de bedrijfsbeveiliging met rechts daarvan de PV Home en de fietsenstalling.