Agenda

➢ Zaterdag	5 juli	Bijeenkomst Hooglanderveen.
➢ Zaterdag	2 augustus	**GEEN** Bijeenkomst Hooglanderveen.
➢ Zaterdag	9 augustus	**Bijeenkomst Hengelo**!!
➢ Zaterdag	6 september	Bijeenkomst Hooglanderveen.
➢ Zaterdag	4 oktober	Bijeenkomst Hooglanderveen.

De bijeenkomsten te Hooglanderveen worden gehouden in Dorpshuis "de Dissel" Disselplein 6 3829 MD te Hooglanderveen. De bijeenkomst in Hengelo wordt gehouden in de PV home van Thales(vroeger Holland Signaal), aan de Robijnweg in Hengelo. Het gebouw van de PV home vind u rechts naast de ingang.Route beschrijvingen op onze website of op die van onze voorzitter Bert Buiskool: www.robot.buiskool.net.

Een motortje dat maar 0.5 gram weegt! Normaal E 1.50, voor RoboBitters 1-betalen-2-krijgen. Je moet wel zelf het gewichtje van de as afschuiven. Trekt 80 mA bij 1.5 Volt. Wie laat dit motortje vliegen?

www.voti.nl/winkel/p/MOT-15.html
vermeldt "RoboBits aanbieding 8" - pas geld overmaken als u het verzoek tot betalen krijgt met het aangepaste bedrag - geldig tot de volgende RoboBits uitkomt.

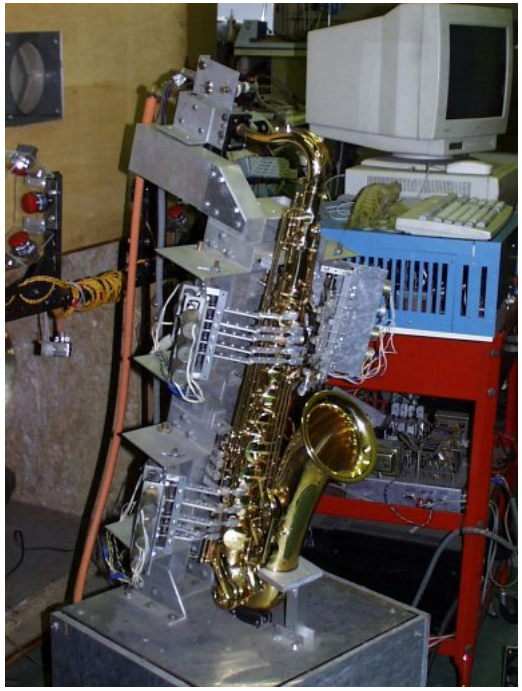
Afz.hcc Robotica gg, p.a. Henk de Gans, Anjerlaan 3, 3871 ev Hoevelaken.

TNT Post
Port betaald
Port Payé
Pays-Bas

ROBO-BITS-41

Jaargang11, nummer 2, juni 2008

CNC-dag 6 september!



9 augustus Hengelo!

hcc robotica

De Robobits is een uitgave van de hcc!robotica gebruikers groep, en wordt vier keer per jaar toegezonden aan de leden. De oplage is ongeveer 490 exemplaren. hcc!robotica is een onderdeel van de hcc! (hobby computer club), een vereniging van bijna 180.000 leden.

==========

Redactie adres: H.J. de Gans, Anjerlaan 3, 3871EV Hoevelaken.
henkdegans@kpnplanet.nl Tekst aanleveren in WORD of platte tekst in ASCII. Afbeeldingen los er bij in JPG, GIF of BMP formaat.

==========

Dagelijks bestuur:

Voorzitter:	B.T.J.A.Buiskool(Bert), robot@buiskool.net
Technisch adviseur:	Ing.H.M.A.van Bodegom(Henny) ing.h.m.a.van.bodegom@hccnet.nl
Secretaris:	A.J.Janssen (Lex) lex.janssen@hccnet.nl
Penningmeester	H.J. de Gans(Henk) henkdegans@kpnplanet.nl
Lid:	P.Smits(Paul) psmits.1@hccnet.nl
Lid:	W.C.de Boer (Wim) wim.deboer@nl.thalesgroup.com

Voorkaft: Ik heb net een Tenor saxofoon gekocht (speelde in mijn jeugd ook al 9 jaar saxofoon, en heb dit nu weer opgepakt). Ik vond op internet dit plaatje, dus het is te combineren met Robotica ;-)

inhouds opgave:

Voor ons was dit project van de WCRS aanleiding voor het idee om hier bij de Robotica GG ook een project te doen maar dan met een Roborama robot. Waarbij het bouwen en programmeren voor de Roborama wedstrijd onderdelen "Heen&Weer", "T-Tijd" en "Lijnvolgen" centraal staan. Op de Robotica GG bijeenkomst van april is dit idee besproken en is er een werkgroep samengesteld om dit idee verder uit te werken. Werkgroepleden zijn nu bezig met het bouwen prototypes. Het doel is om mogelijk al de komende winter een workshop hierover te organiseren. Maar daarover later meer.

Om te ervaren hoe het programmeren met een generator gaat hebben we een "SAM r I" kit besteld. Het SUMO Sensei programma is wel te downloaden maar je hebt de betreffende hardware nodig om er ervaring mee op te doen. Hoewel het niet ons doel was zijn de resultaten van de "SAM r I" met het SUMO Sensei programma zeker interessant. Hierdoor willen we de "SAM r I" presenteren aan Robotica GG op de bijeenkomst van 5 juli. De presentatie/demonstratie zal starten om 11:00 uur.

Met vriendelijke groeten,
Hinnie.

Presentatie van de "SAM r I" Sumo robot

Een project uit 2005 van de "Western Canadian Robotics Society" waar een complete Sumo robot kit "SAM-bot" werd aangeboden voor hun leden voor slechts 40 CAN dollar heeft mijn aandacht getrokken. Er werd een workshop georganiseerd voor het bouwen van de Sumo robot waarin de Sumo rijklaar kon worden gebouwd en geprogrammeerd in slechts 4 tot 6 uur. Voor het programmeren van de Sumo hebben zij een interessante oplossing gekozen. Met een PC programma "SUMO Sensei" kunnen grafisch parameters worden ingevuld waarna het een C-source voor een AVR processor genereert. Een bouw beschrijving van de SAM-bot is beschikbaar op
http://www.robotgames.net/SAMBot/Instructions/WCRS%20SAMbot%20Guide.pdf
Dit project van de WCRC was slechts éénmalig maar op dit moment wordt deze oplossing aangeboden door JCMinventures
http://www.jcminventures.com als "Sumo SAM r I Robot Kit" voor slechts 79,95 CAN dollar. Op deze website is ook het programma aanwezig waarmee de sumo code kan worden gegenereerd "SUMO Sensei" gratis te downloaden.

REDACTIE

Met voor de meeste mensen de vakantie voor de deur, ligt hier wederom een gevulde robobits voor u klaar! Er was zelfs zoveel te melden dat ik geen plaats meer had voor een wedstrijd verslag van de in mei gehouden RoboRama wedstrijd, die georganiseerd werd door RobotMC. De volgende Robobits daarover een uitgebreid verslag! Ook kunt u vast de datum 6 september in uw agenda schrijven, naast de normale bijeenkomst is er dan ook CNC dag! Ook de elke zomer gehouden bijeenkomst in Hengelo is een plekje in uw agenda waard, we worden daar altijd bijzonder gastvrij ontvangen! Ik wens u veel leesplezier, en een fijne vakantie toe.

Henk de Gans

LET OP!!!
De bijeenkomst van zaterdag 2 augustus in Hooglanderveen gaat niet door, 9 augustus wel bijeenkomst in Hengelo!!

---

deadline deadline deadline deadline deadline deadline deadline deadline

---

De deadline voor kopie voor Robobits 42 die eind september 2008 zal uitkomen, is gesteld op woensdag avond 24 september 2008!!!! Maar u hoeft natuurlijk niet zo lang te wachten, nu al insturen kan natuurlijk ook! Iedereen wil heel graag eens wat lezen over dat waar u mee bezig bent, en natuurlijk met onze hobby te maken heeft! Vooral van u, die niet in de gelegenheid bent onze bijeenkomsten te bezoeken! Zoals al eerder gezegd, hoeven het geen hoogdravende journalistieke meester werken te zijn(mag natuurlijk wel;-))). Gewoon een leuk stukje over uw creatie of iets dergelijks, of wat informatie over iets waarvan u gehoord of gelezen hebt enz. Ook zoek ik nog een boek bespreking(of tijdschrift).

---

deadline deadline deadline deadline deadline deadline deadline deadline

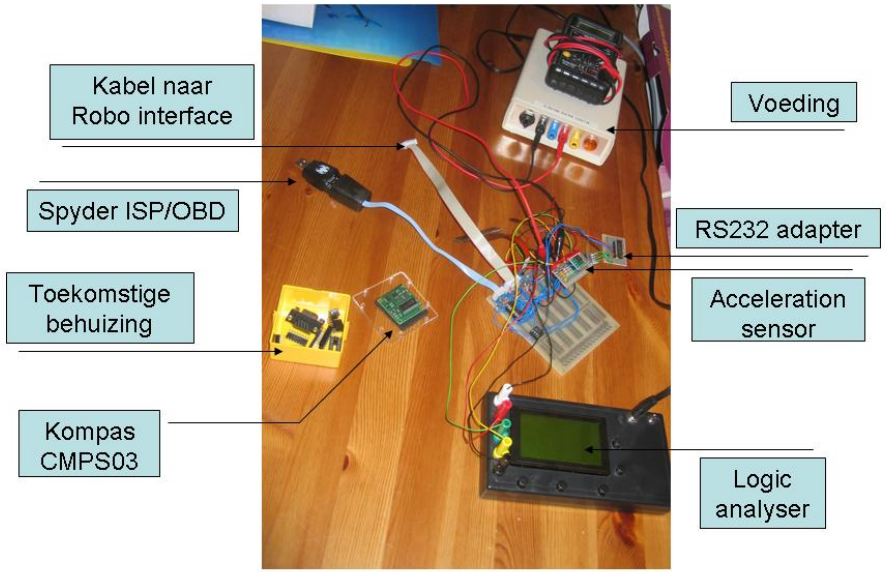
Kijk ook eens op onze website:

WWW.HCCROBOTICA.NL

sensorpakket voor mobiele robot

Door: Ad van der Weiden

Momenteel ben ik bezig met het bouwen van een sensorpakket voor gebruik met de RoboInterface van FischerTechnik. Dit pakket is ook geschikt voor andere mobiele robots en daarom misschien interessant voor de lezers van dit blad. Het pakket is nog niet helemaal af maar ik wilde mijn ideeen toch al met jullie delen omdat ik op dit moment door een gebroken been weinig anders kan dan lezen en schrijven.



Proefopstelling

Bij het ontwerp ben ik geinspireerd door een aantal artikelen in Elektor [1] en door ideeen van andere FischerTechnik fans. Voor wat betreft de FischerTechnik kant is dat vooral de interfacing naar de RoboInterface via de I/O Extension bus en eventueel naar andere interfaces via de RS232 interface. Via de RS232 interface heb ik eerder al een gemodificeerde versie van de CMUCAM3 op de RoboInterface aangesloten. De CMUCAM3 paste precies in een FT cassette van 60x60x30 mm. Daarom wilde ik ook dit sensorpakket in zo'n cassette proberen te proppen.

Al met al zeer nuttige bijeenkomsten die zeker niet alle geheimen van de betreffende hardware en het programmeren ervan weg hebben genoemen. Wel was het een zeer waardevolle en nuttige aanpak om van start te gaan met het ontwikkelen van een "eigen" programma voor de eerder gebouwde Sumo en uiteraard is er nu voldoende kennis opgedaan om later dit jaar eens een "home brewed" machientje te maken.
Wil je meer informatie, mail dan naar: WorkShop-C@buiskool.net

met dank aan Hinnie en tot ziens,
Ed Buzzi

++

Bijeenkomst Hengelo!

Ook dit jaar zijn wij weer uitgenodigd door de PCGG van Thales om in hun PV Home een bijeenkomst te organiseren. *Dit maal in plaats van de bijeenkomst in Hooglanderveen op de eerste zaterdag van de maand*, maar LET OP!!!!op de tweede zaterdag van de maand d.d.9 augustus 2008.
U bent van harte welkom, met of zonder uw eigen creatie!

aan de slag met C!

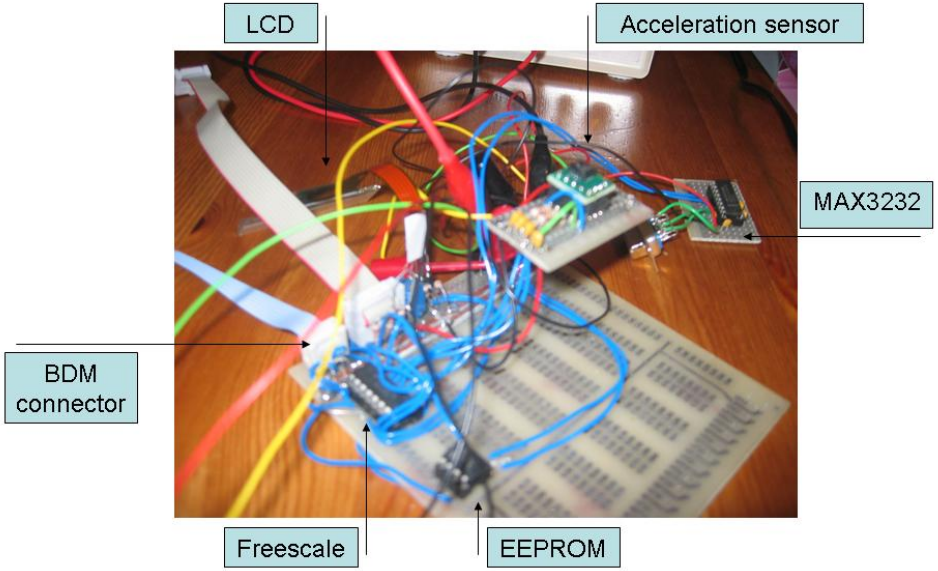
Het is de omgekeerde wereld, eerst een Sumo bouwen, dan aan een wedstrijd deelnemen met het programma van een ander, om je vervolgens te verdiepen in de programmering!! Jullie hebben kunnen lezen in Robits 38 en 39 hoe het zo allemaal gekomen is.Begin 2007 een Sumo project en dat wilden een aantal van de Robotica leden toch niet missen ...

Een van de mede-bouwers -Hinnie van Sintannaland- heeft in januari en februari twee workshops verzorgd.

De focus van de workshops was erop gericht om de deelnemers te stimuleren zich te verdiepen in de hardware (materiaalkennis zoals Hinnie dat noemt) en vervolgens in de basis software die is nodig is om in C met deze hardware te communiceren.De hardware waarmee gewerkt werd was enerzijds vergelijkbaar met de compenenten die gebruikt werden bij de bouw van de Sumo en anderzijds zijn de gebruikte componenten zo algemeen dat goede kennis opgebouwd kon worden over de meest gangbare onderdelen die nodig zijn om een eenvoudige robot te bouwen en te programmeren.

De workshop was modulair opgebouw en bestond uit een aantal oefeningen. De deelnemers werden geacht deze vooraf thuis te bestuderen en op te lossen. Tijdens de workshops werden de oefeningen uitgebreid besproken en gedemonstreerd. Verder was er gelegenheid om zelf te experimenteren en hulp te krijgen van Hinnie en andere deelnemers.

Een kort overzicht van de oefeningen:
(Als controller werd het Olimex AVR Protoboard met ATMega32 gebruikt)
- communiceren met drukknop / LED via de digitale I/O poorten
- communicatie via RS232 met hyperterminal op de PC
- uitlezen van analoge ingangen / werken met de ADC
- weergeven van de gemeten waarde v.e. afstandsensor op de PC (hyperterminal)
- motoren besturen (PWM) met de PC
- tekst uitlezen op LCD
- aansturen van LEDS via de I2C-bus
- TV afstandsbediening decoderen (RC5). Output naar LCD en/of PC
- genereren van geluid met een piezo zoemer



De experimenteerprint

De oorspronkelijke wensen waren als volgt:

1 Aansturing via I/O Extension bus
2 LCD display (Starburst display van VOTI)
3 Kompas sensor (CMPS03)
4 Versnellingssensor (MMA7260QT) voor standbepaling en berekening van snelheid en positie
5 Ingangen voor optische sensoren (8 CNY70 of 4 kwadratuur sensoren uit oude muizen)
6 Digitale ingangen voor schakelaars (4 stuks)
7 RS232 interface voor verbinding met andere FT interfaces of een computer
8 EEPROM voor log-data of programma's die naar een FT interface ge-upload kunnen worden.

Door ruimtegebrek kan ik niet alles in een standaard cassette realiseren, de ingangen voor de optische sensoren en de schakelaars zullen dus waarschijnlijk sneuvelen, misschien maak ik daar later nog een uitbreiding voor. De FT I/O Extension was gebaseerd op een ATMEGA met nog wat extra logica. Ik vond dit te ingewikkeld en ben uitgegaan

van de versnellingssensor uit Elektor. Dit projekt is gebaseerd op een Freescale microcontroller HCS08 (MC9S08QG8CPBE, gratis sample) en een 3-assige versnellingssensor (ook van Freescale). Als je de printjes besteld bij Elektor krijg je er 2 sensors bij die bovendien al op een printje gesoldeerd zijn. Ook is er van Freescale een gratis ontwikkelpakket en een zeer goedkope programmer/on-chip-debugger (Spyder, ca. 10 Euro). Inmiddels ben ik er achter gekomen dat mijn programma niet in de 8k/512 van de HCS08 gaat passen en ben dus toch maar overgestapt naar een ATMega168. Deze heeft een beetje EEPROM on-chip en dus heb ik de externe EEPROM laten vervallen (data logging had ik toch nog niet geimplementeerd). Wat ik wel erg mis is de on-chip debugger dus ik heb toch maar een AVR Dragon besteld .

De microcontroller gebruik ik eigenlijk als een grote protokol converter. Ik heb namelijk te maken met de volgende interfaces:

- I2C, voor de LCD, CMPS03, EEPROM en de I/O uitbreiding voor de digitale ingangen;
- SPI, voor de FT I/O Extension
- RS232, voor de RS232 interface
- Analoog, voor de versnellingsopnemer

De Freescale microcontroller kan ondanks zijn slechts 16 pennen al deze funkties tegelijk vervullen (in hardware)! De ATMega kan dit met 28 pennen natuurlijk ook en heeft dan zelfs nog pennen over voor bijv. een standaard LCD of andere toepassingen. Voor toepassingen in andere robots dan die met de FischerTrechnik Robo interface kan men dus het beste de SPI interface gebruiken voor communicatie met de centrale processor. De software moet op dit punt dan wel behoorlijk gewijzigd worden.

In het vervolg van dit stukje wil ik me concentreren op de signaalverwerking van het kompas en de versnellingsopnemer. De software is gebaseerd op Application Notes van Freescale voor de versnellingsopnemer voor zowel standbepaling [2] als snelheids- en positieberekening [3].

Tilt

De gebruikte sensor is een 3-assige sensor, de x en y assen geven de voorwaartse en de zijwaartse versnelling aan, de z-as geeft de neerwaartse versnelling aan, oftewel de valversnelling. Als men de sensor kantelt in pitch of roll richting zullen componenten van de

geprogrammeerd heb kan (**zeker !!**) wat professioneler. Ik houd me hiervoor aanbevolen !

Ik wil er nogmaals op wijzen dat alles valt of staat met de layout van de doolhof. De beschreven methode geldt alleen voor een doolhof, die met de zgn "rechterhandregel" op te lossen is (doolhof met slechts één oplossing)
Hebben we bijvoorbeeld te maken met **een doolhof met lussen** (doolhof met meerdere oplossingen) dan wordt het allemaal een stuk ingewikkelder. Mijn ervaring is echter tot nu toe dat het tot een "goed einde brengen" van een MSR in een doolhof zonder lussen al een grote uitdaging is (dat geldt in ieder geval voor mij).

Mochten er binnen de Club doolhof-enthousiastelingen zijn die wat kunnen vertellen over een "lussendoolhof" dan geef ik het stokje graag over.

Aan u allen een fijne vakantie toegewenst.

CNC dag 2008

In navolging van de succesvolle CNC dag van 2007, organiseren wij ook dit jaar, en wel op zaterdag 6 september 2008 wederom een CNC dag! Meer informatie kunt u krijgen bij Lex Janssen lex.janssen@hccnet.nl of op het forum van CNCzone www.cnczone.nl aanvang 10.00 uur.

```
'Bepalen  richting (N, O, Z, W) van de trajecten
(j is het aantal nieuwe knopen  dus in dit geval gelijk aan 7)
j = j – 1 (is het aantal trajecten)
For i = 1 To j
   U2 = X(i + 1)
   U1 = X(i)
   Difu = U2 - U1
   V2 = Y(i + 1)
   V1 = Y(i)
   Difv = V2 - V1
   If Difu <> Difv Then
      If  U1 = U2 And Difv = 1 Then
         Richting(i) = &B1000                    'noorden
      Elseif  U1 = U2 And Difv = -1 Then
         Richting(i) = &B0010                    'zuiden
      Elseif  V1 = V2 And Difu = 1 Then
         Richting(i) = &B0100                    'oosten
      Elseif  V1 = V2 And Difu = -1 Then
         Richting(i) = &B0001                    'westen
      End If
    End If
Next I

'Bepalen draairichting t.p.v. de
knooppunten
j = j – 1 (is het aantal draaibewegingen)
For i = 1 To j
  If Richting(i) = &B1000 Then
     If Richting(i + 1) = &B0100 Then
        Print "draaien rechts"
     Elseif Richting(i + 1) = &B0001 Then
        Print " draaien links"
     Elseif Richting(i + 1) = &B1000 Then
        Print "rechtdoor"
     End If
  Elseif Richting(i) = &B0100 Then
     If Richting(i + 1) = &B0010 Then
        Print "draaien rechts"
     Elseif Richting(i + 1) = &B1000 Then
        Print " draaien links"
     Elseif Richting(i + 1) = &B0100 Then
        Print "rechtdoor"
     End If
  Elseif Richting(i) = &B0001 Then
     If Richting(i + 1) = &B1000 Then
        Print "draaien rechts"
     Elseif Richting(i + 1) = &B0010 Then
        Print " draaien links"
     Elseif Richting(i + 1) = &B0001 Then
        Print "rechtdoor"
     End If
  Elseif Richting(i) = &B0010 Then
     If Richting(i + 1) = &B0001 Then
        Print "draaien rechts"
     Elseif Richting(i + 1) = &B0100 Then
        Print " draaien links"
     Elseif Richting(i + 1) = &B0010 Then
        Print "rechtdoor"
     End If
  End If
Next i
```

4. Ten slotte

Ook nu geldt weer dat de beschreven methode voor het bepalen van de korste route niet alleen zaligmakend is. Het is een van de vele manieren om tot een oplossing te komen. Typ op het web "maze solving" en je krijgt een vracht aan informatie. Ook de manier waarop ik het zwaartekracht vector in de x en y as terecht komen. Met behulp van wat goniometrie kunnen dan de pitch en de roll berekend worden. De yaw (draaiing om de verticale as) kan zo niet bepaald worden maar daar kan het kompas uitkomst brengen. Om deze meting betrouwbaar uit te voeren moet de robot stilstaan of eenparig bewegen omdat anders de versnellingen van de robot de metingen verstoren.

In de Application Note [2] vond ik de volgende formule voor de pitch:

$\tan\theta = \frac{A_x}{\sqrt{A_y^2 + A_z^2}}$ en in een ander document [7] $\sin\theta = \frac{A_x}{\sqrt{A_x^2 + A_y^2 + A_z^2}}$

deze formules zijn equivalent. Voor de roll staat in de Application Note [2]:

$\tan\phi = \frac{A_y}{\sqrt{A_x^2 + A_z^2}}$ en in een ander document $\sin\phi = \frac{A_y}{g\cos\theta} = \frac{A_y}{\sqrt{A_y^2 + A_z^2}}$

deze formules zijn niet equivalent, slechts als er alleen sprake is van roll of van pitch komen ze overeen. Als er sprake is van zowel pitch als roll geeft alleen de tweede formule het juiste resultaat. Het is van belang om de rotaties in de juiste volgorde uit te voeren. Eerst wordt de pitch uitgevoerd, draai het voertuig bv. eerst 45º om z'n wielas (rijdt vrij steil omhoog) en rol het vervolgens 90º om z'n lengte as. In deze volgorde zijn de g componenten [0,7 0,7 0,0], zou men eerst de rol uitvoeren en daarna de pitch dan werden de componenten [0,0 1,0 0,0], de formules geven dan een pitch van 0 wat niet overeenkomt met wat we gedaan hebben.
De volgende opgave is om deze formules te berekenen. Als dit willen berekenen met de gebruikelijke C bibliotheken is het geheugen van de microcontroller gauw vol, er blijkt echter een eenvoudigere methode die werkt met integers en qua snelheid vergelijkbaar is met vermenigvuldigen. De methode heet CORDIC en op Internet is alles erover te vinden [4]. Met CORDIC in vectoring mode kunnen berekeningen als arctan en pythagoras heel eenvoudig uitgevoerd worden.

```
#define ITERATIONS 15
int term[ITERATIONS];

void vectoring(int &x, int &y, int &z)  //C++ style call by reference
{ if (x<0)  //CORDIC werkt alleen in het rechter halfvlak
  { x *= -1;
```

```
    y *= -1;
    z += 18000;
  }
 for (int i = 0; i < ITERATIONS; i++) //het eigenlijke CORDIC proces
    if (y > 0)
    { int t = x;
      x += y>>i;
      y -= t>>i;
      z += term[i];
    }
    else
    { int t = x;
      x -= y>>i;
      y += t>>i;
      z -= term[i];
    }
}
```

Het array 'term[]' moet wel eerst gevuld worden maar dit kan met een lijst constanten. Hieronder staat C code waarmee de constanten berekend kunnen worden. Merk overigens op dat de CORDIC loop alleen optel/aftrek en schuif operaties bevat.

```
for (int i = 0; i < ITERATIONS; i++)
    term[i] = 18000/M_PI*atan(1.0/(1<<i)) + 0.5;
```

De tabel is hier zo gekozen dat hoeken worden gerepresenteerd in honderdste graden, de integer 18000 komt dus overeen met een hoek van 180,00 graden. Men is vrij een andere representatie te kiezen zolang het maar in 16 bits past. Wil men meer of minder bits dan moet het aantal iteraties worden aangepast.

De CORDIC processor heeft 3 argumenten: x, y en een hoek. Hij roteert de x,y vector over de hoek. In vectoring mode wordt de vector geroteerd tot hij langs de (positieve) x-as ligt. De hoek fungeert dan als uitgang en geeft de hoek aan waarover de vector gedraaid is. Omdat de vector na rotatie op de x-as ligt (y=0), is de x waarde noodzakelijkerwijs de lengte van de x,y vector. Althans dat zouden we graag willen maar de eenvoud van het proces heeft z'n prijs en die prijs is dat bij iedere iteratie de vector iets langer wordt. Omdat het aantal iteraties constant is, is ook de totale lengteverandering constant.en bedraagt bij 16 bit een factor 1, 647. Dit is wel iets waar men rekening mee moet houden. Om nu bv. de

5. Rijden kortste route MSR in de 2de ronde

Rest nu nog de MSR te vertellen dat 'ie in een tweede ronde het "schema" van Tabel 4 moet volgen (vanuit knooppunt 1).

In principe gaat dit in omgekeerde volgorde van hetgeen in Hfdst. 3 besproken is.

1. Bepaal vanuit de coördinaten van de knopen de richting van de trajecten;
2. Als de richting van de trajecten bekend is, dan kan hieruit de draairichting t.p.v. de knopen bepaald worden.

De richting (N of O of Z of W) kan uit de coördinaten bepaald worden bv :

1. Als (x)→ (x) en (y)→(y+1) dan betekent dit: er moet gereden worden in noordelijke richting (trajectflag = &B1000);.

of

2. Als (x)→ (x-1) en (y)→(y) dan betekent dit: er moet gereden worden in westelijke richting (trajectflag = &B0001);
3. etc., etc.

Als je richting noorden (trajectflag=&B1000) gaat en je moet bv naar het westen (trajectflag=&B0001) dan is het quotient 8 en dit betekent naar links draaien (zie Hfdst 3).

Als voorbeeld kijken we wat de MSR doet van traject 4 naar traject 5 (zie Tabel 4):

1. Traject 4 loopt van (x,y) = (1,2) naar (x,y) = (2,2) , loopt dus naar het oosten (trajectflag = &B0100);
2. Traject 5 loopt van (x,y) = (2,2) naar (x,y) = (2,3) , loopt dus naar het noorden (trajectflag = &B1000).

Quotient is ½ <1 dus quotient wordt 16 x ½ = 8 → dit betekent naar links draaien.
Of van traject 1 naar traject 2 :

3. Traject 1 loopt van (x,y) = (0,0) naar (x,y) = (0,1) , loopt dus naar het noorden (trajectflag = &B1000);
4. Traject 2 loopt van (x,y) = (0,1) naar (x,y) = (1,1) , loopt dus naar het oosten (trajectflag = &B0100).

Quotient is 2 → dit betekent naar rechts draaien.

Ik heb het als volgt geprogrammeerd:

Hoe elimineer je deze trajecten ? In principe elimineer ik de dubbele knopen
Ik kan niets beters verzinnen (in Bascom) dan:

```
'Stap1: Dubbele knopen =0 maken
 For i1 = 1 to aantal_ knopen (is in dit voorbeeld gelijk aan 13)
   V = X(i1)
   U = Y(i1)
   W = i1 + 1
    For  i2= W to aantal_knopen
      If  V = X(i2)  and  U = Y(i2) then
          knoop(i2 - 1) = 0
          knoop(i1 + 1) = 0
          knoop(i1)=0
      End if
   Next  i2
   Next i1

'Stap 2: elimineren van de nul-knopen
j = 0  (j wordt het nieuwe aantal knopen)
For i = 1 To aantal_knopen (nog steeds in dit voorbeeld gelijk aan 13)
    If  knoop (i)  <>  0  Then
       j =j + 1
       knoop (j) = J
       X(j) = X(i)
       Y(j) = Y(i)
    End If
Next i
```

In de microcomputer staan nu (*na een eerste ronde en na de eliminatie*) de volgende gegevens opgeslagen: (= dus de korste route) (zie de dik gedrukte kolommen van Tabel 4):

Tabel 4

Traject	Van			Naar		
	Knoop	X	Y	Knoop	X	y
1	**1**	**0**	**0**	2	0	1
2	**2**	**0**	**1**	3	1	1
3	**3**	**1**	**1**	4	1	2
4	**4**	**1**	**2**	5	2	2
5	**5**	**2**	**2**	6	2	3
6	**6**	**2**	**3**	7	1	3
	7	**1**	**3**			

Let op: de korste route bestaat dus uit 6 trajecten en 7 knopen (j=7) (zie ook de figuur in Hfdst 2)

pitch te berekenen uit de versnellingsvector doen we de volgende berekeningen:

```
int  dummy = 0;
vectoring(Ay, Az, dummy);
//Az=0, Ay=1,647 * SQRT(Ay*Ay+Az*Az), dummy = atan(Az/Ay)
vectoring(Ax, 0, dummy);
//Ax = 1,647*Ax, operatie om Ax te vermenigvuldigen met de CORDIC gain
int pitch = 0;
vectoring(Ay, Ax, pitch);
// pitch = atan(y/x)
```

De CORDIC processor kan ook ingezet worden voor het berekenen van sinus, cosinus, arcsinus, arccosinus en nog veel meer functies. Op die manier zou men dus ook de roll kunnen berekenen. We kunnen de sinus-formule voor de roll echter omschrijven naar een tangens-formule, deze blijkt ook nog eens veel eenvoudiger te zijn: roll = arctan(Ay/Az). Deze is met een enkele vectoring operatie te realiseren.

Dead Reckoning

Een heel andere toepassing van de versnellingsopnemer is dead reckoning, ik heb me laten vertellen door een maritieme vriend dat de nederlandse term 'gegist bestek' is. Het komt erop neer dat je uitgaande van je vertrekpunt op basis van koers en vaart uitrekent waar je zou moeten zijn. In ons geval hebben we alleen versnellingsinformatie in x en y richting. In principe zouden we dit kunnen aanvullen met informatie van het kompas maar ik wilde me in eerste instantie houden aan het principe zoals beschreven in de Application Note [3]. De code in de AN is bedoeld voor een muis maar is eenvoudig geschikt te maken voor een robot. De calibratie stap is vrijwel gelijk, het enige verschil is dat wij, vanwege het gebruik als tilt-sensor, ook de z-as gebruiken. De z-as geeft echter niet het nulpunt aan maar het 1g punt. Voor het z nulpunt zal men dus een of andere constante moeten nemen. Dead reckoning kan men zowiezo alleen in het platte vlak gebruiken omdat bij stijgen/dalen of kantelen de zwaartekracht een enorme fout introduceert. Het is voor zover ik weet niet eenvoudig mogelijk om onder die omstandigheden nog goede resultaten te bereiken.
Men zou eventueel het volgende kunnen proberen:

- Met extra sensoren de pitch en roll bepalen en hiermee de zwaartekracht vector als het ware weer terugdraaien naar de z-as. Als sensoren zijn gyro's heel geschikt maar men zou ook kunnen experimenteren met twee extra lineaire versnellingsopnemers die iets voor en iets naast de hoofdsensor worden geplaatst. Uit het verschil is dan de draaiing te bepalen.
- Aangenomen dat de versnellingen voor het verplaatsen een hogere frequentie hebben dan de versnellingen gerelateerd aan de stand zou men op basis van frequentie de versnellingen kunnen scheiden. Ik heb echter geen idee of deze aanname reeel is.
- Met een geschikt filter (bv. een Kalman filter) de stand van de robot schatten en deze schatting van tijd tot tijd calibreren met de werkelijke stand door de robot tot stilstand te brengen.

Het leuke van het gebruik van versnellingsopnemers t.o.v. bv wielsensoren is dat men ook nog een schatting van de positie krijgt als men de robot oppakt of door een ander voertuig wordt weggeduwd.

De bovenstaande problemen overwegende heeft het niet veel zin om te proberen de vertikale positie te bepalen. Alle robots die ik tot dusver gezien heb in RoboBits waren gemaakt voor een vlakke vloer.
De Freescale AN maakt voor de berekening van de snelheid en positie gebruik van de trapezium integratie. Er wordt gesuggereerd dat deze veel nauwkeuriger is dan de simpele rechthoek integratie. In de praktijk valt dat mee, in beide gevallen worden gewoon alle samples opgeteld alleen worden bij de trapezium integratie het allereerste en het allerlaatste (dus het huidige) sample maar voor de helft meegeteld. De code in de AN berekent de snelheid als volgt: v1 = v0 + a0 + (a1 – a0)/2 dit is uiteraard eenvoudiger te schrijven als: v1 = v0 + (a0 + a1)/2 dat scheelt vast weer een optelling. Het is ook eenvoudig in te zien dat het n-de sample vn = v0 + a0/2 + a1 + a2 + ... + a(n-1) + an/2. Daarom kunnen we eens kijken of simpele rechthoek integratie niet net zo goed is als trapezium integratie. Ik ga geen uitspraak doen welke methode 'beter' is maar bekijk alleen de onderlinge verschillen. Ik zal u de wiskunde besparen maar als we ervan uit gaan dat aan het begin de versnelling, snelheid en positie nul zijn, kom ik tot de volgende resultaten:
Vtrap(nT) = Vrect(nT) - ½Ta(nT) = Vrect((n-½)T)
Strap(nT) = Srect(nT) – T Vrect(nT) + ¼T²a(nT) = Srect((n-1)T) + ¼T²a(nT)
Het komt erop neer dat bij de berekening van de snelheid er een verschuiving van een halve sample periode optreedt en bij bepaling van

	Van			Draaien	Richting	Naar		
Traject	**Knoop**	**X**	**Y**		**Traject flag**	**Knoop**	**X**	**Y**
1	**1**	**0**	**0**		&B1000	2	0	y+1=1
2	**2**	**0**	**1**	Rechts	&B0100	3	x+1=1	1
3	**3**	**1**	**1**	Rechts	&B0010	4	1	y-1=0
4	**4**	**1**	**0**	Links	&B0100	5	x+1=2	0
5	**5**	**2**	**0**	Omdraaien	&B0001	6	x-1=1	0
6	**6**	**1**	**0**	Rechts	&B1000	7	1	y+1=1
7	**7**	**1**	**1**	Rechtdoor	&B1000	8	1	y+1=2
8	**8**	**1**	**2**	Rechts	&B0100	9	x+1=2	2
9	**9**	**2**	**2**	Rechts	&B0010	10	2	y-1=1
10	**10**	**2**	**1**	Omdraaien	&B1000	11	2	y+1=2
11	**11**	**2**	**2**	Rechtdoor	&B1000	12	2	y+1=3
12	**12**	**2**	**3**	Links	&B0001	13	x-1=1	3
	13	**1**	**3**					

Tabel 2

4. Elimineren doodlopende takken

Om de kortste route te vinden moeten de trajecten, die dubbel bereden worden (de zgn doodlopende takken) geëlimineerd worden. Dit zijn de volgende trajecten (zie Tabel 2 en 3):

1. Traject 4 en 5;
2. Traject 3 en 6;
3. Traject 9 en 10.

Tabel 3

Traject	Van			Naar		
	Knoop	**X**	**y**	**knoop**	**X**	**Y**
4	4	1	0	5	2	0
5	5	2	0	6	1	0
3	3	1	1	4	1	0
6	6	1	0	7	1	1
9	9	2	2	10	2	1
10	10	2	1	11	2	2

3 Vastleggen van de coördinaten van de knopen in de eerste ronde

De coördinaten van de knopen worden vastgelegd aan de hand van de richting van de trajecten .

Bekijk tabel eens en wat valt op ?
Traject 1 loopt (uit hoofde van de aanname) naar het noorden (trajectflag = &B1000).
Traject 2 loopt naar het oosten (trajectflag = &B0100).
Het lijkt er op dat naar rechts draaien voor de nieuwe richting betekent: quotient (=oude richting/nieuwe richting) =2.
Van traject 2 naar traject 3 is weer draaien naar rechts ... quotient = 2 geeft als nieuwe richting: trajectflag = &B0010 (= zuid)..... klopt !!!
Zo krijgen wij:

1. Naar rechts draaien op een kruispunt betekent voor de richting: quotient trajectflags = 2;
   (dus nieuwe richting is (in Bascom) shift trajectflag, right ,1)
2. Naar links draaien op een kruispunt betekent voor de richting: quotient = 8;
   (dus nieuwe richting is (in Bascom) shift trajectflag, right, 3)
3. Omdraaien op een kruispunt betekent quotient = 4;
4. en Rechtdoor betekent voor de richting quotient = 1.

Opmerking: omdat je niet eindeloos kunt blijven delen moet er indien het quotient <1 het "deeltal"(= oude richting) vermenigvuldigd worden met 16.

Als de richting van de trajecten bekend is kunnen de coördinaten van de knopen bepaald worden.

1. Naar het noorden rijden (trajectflag = &B1000) betekent voor de coördinaten van de volgende knoop (x)→ (x), (y)→(y+1);
2. Naar het oosten rijden (trajectflag = &B0100) betekent voor de coördinaten van de volgende knoop (x)→ (x+1), (y)→(y);
3. Naar het zuiden rijden (trajectflag = &B0010) betekent voor de coördinaten van de volgende knoop (x)→ (x), (y)→(y-1);
4. Naar het westen rijden (trajectflag = &B0001) betekent voor de coördinaten van de volgende knoop (x)→ (x-1), (y)→(y);

In de microcontroller worden nu (*in de eerste ronde*) de volgende gegevens opgeslagen (zie de dik gedrukte kolommen in Tabel 2):

1. Trajectnummers;
2. Knoopnummers met coördinaten.

de afstand een hele sample periode. Het lijkt me dat deze verschillen voor alle praktische toepassingen te verwaarlozen zijn maar puristen kunnen altijd de rechthoek integratie gebruiken vanwege het rekenwerk en dan een kleine correctie aanbrengen om het resultaat van de intuitief betere trapezium integratie te bereiken.
De vermenigvuldiging met de sampletijd T om van versnelling naar snelheid en van snelheid naar afstand te komen wordt natuurlijk niet iedere periode uitgevoerd maar pas bij het weergeven op een display o.i.d. men kan dan gelijk de gevoeligheid van de sensor (800mV/g) en de A/D conversie (3,3V/1023) verdisconteren. De berekeningen die uitgevoerd moeten worden zijn dus simpelweg:
v += a;
s += v;
en dat is heel wat eenvoudiger dan de code van Freescale.
Ik heb overigens ook code gezien waar de afstand wordt berekend met: $s += \frac{1}{2}aT^2$ dit is gewoon fout, het zou wel goed (en zelfs beter) zijn als er stond:
$s += vT + \frac{1}{2}aT^2$ maar deze laatste term is verwaarloosbaar klein

Freescale doet nog twee extra dingen om de nauwkeurigheid te verbeteren, ten eerste gebruiken ze 64-voudige oversampling die ze simpelweg middelen. Dit vermindert uiteraard de ruis maar middeling is niet zo geschikt voor het onderdrukken van hogere frequenties. Een echt decimatiefilter zou beter zijn maar vergt veel rekenwerk. Ten tweede hebben ze een discriminatievenster van ±3 gebruikt zodat hele kleine versnelling die waarschijnlijk ruis of drift zijn niet meetellen in het resultaat. Of dit veel zin heeft weet ik niet, ruis en drift worden al onderdrukt dus ik zal hier wat mee moeten experimenteren. Als laatste gebruikt Freescale een 'movement_end_check', dit houdt in dat wordt gekeken of de versnelling 25 keer achter elkaar nul is (na toepassing van het discriminatievenster). Als dat het geval is wordt aangenomen dat het object stil staat en wordt de snelheid op nul geforceerd. Dit is bij een muis wel handig maar bij een robot weten we meestal wel of hij rijdt of stilstaat (tenzij we hem oppakken natuurlijk).

In Robobits 42 gaat dit artikel (wat ik wegens zijn lengte heb moeten inkorten)verder! Ad gaat dan verder met het Kompas.
Redactie

Maze Solving Robot

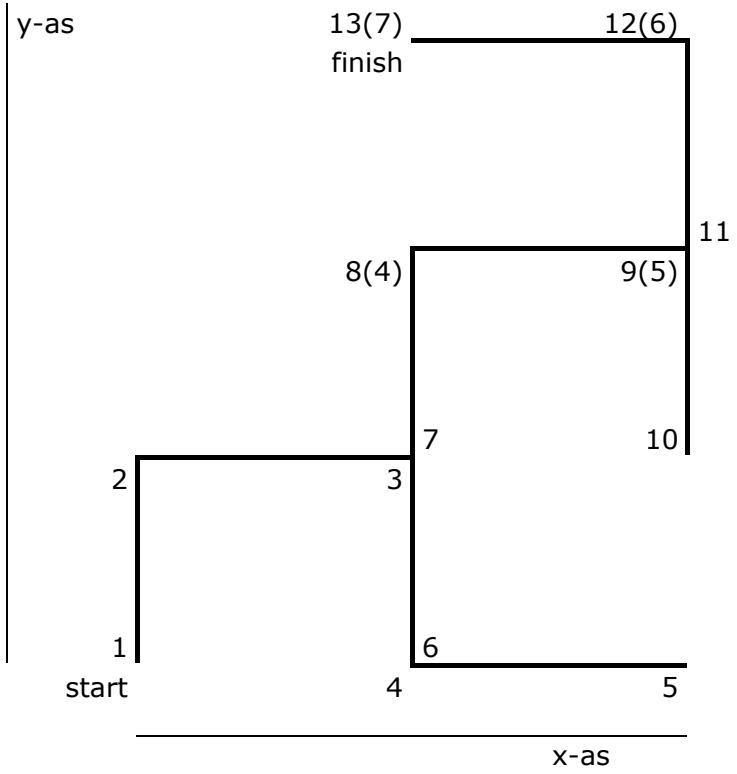
Deel 3 : De kortste route.

Door Rien van Harmelen

Deel 1 is verschenen in Robotbits-39.
Deel 2 is verschenen in Robotbits-40.

1 Inleiding

In de voorgaande delen hebben we gezien hoe de MSR tussen de kruispunten "keurig" de zwarte lijn blijft volgen en welke beslissing er genomen moet worden t.p.v. de kruispunten.In dit laatste deel wil ik uit de doeken doen hoe in een tweede ronde de kortste route gereden kan worden. In principe komt het hierop neer: de eerste ronde wordt gebruikt om de afgelegde route in het geheugen op te slaan om vervolgens de doodlopende takken (dus trajecten die dubbel bereden worden) te elimineren. Met de aldus verkregen gegevens kan in een tweede ronde sneller de finish bereikt worden.



2 Uitgangspunten

Om de doodlopende takken te kunnen elimineren moeten we (in een eerste ronde) de layout van de doolhof in de microcontroller vastleggen. Naast de uitgangspunten, genoemd in Hfdst. 2 van Deel 2, hanteer ik hierbij de volgende uitgangspunten:

1. De MSR begint in de doolhof op knoop 1 (knoop ≈ kuispunt) en rijdt achtereenvolgens naar de knoop 2, 3, etc.;
2. Knoop 1 heeft de coördinaten (x,y) = (0,0);
3. Het traject van knoop 1 naar knoop 2 is (default) naar het noorden geörienteerd;
4. De richtingen N, O, Z, en W geef ik (in Bascom) aan met resp.: trajectflag &B1000, &B0100, &B0010 en &B0001.

Om het een beetje inzichtelijk te maken ga ik in dit artikeltje uit van de bovenstaande doolhof.

Voor de knoopnummering tussen() zie Tabel 4
De lengte van de trajecten neem ik gemakshalve op 1 lengte-eenheid.

Met behulp van de les uit deel 2 rijdt de MSR, in de eerste ronde, volgens Tabel 1 door de doolhof (als tenminste alles goed gaat !):

Tabel 1

Traject	Van			Naar			Draaien
	Knoop	X	Y	Knoop	x	Y	
1	1	0	0	2	0	1	Rechts
2	2	0	1	3	1	1	Rechts
3	3	1	1	4	1	0	Links
4	4	1	0	5	2	0	Omdraaien
5	5	2	0	6	1	0	Rechts
6	6	1	0	7	1	1	Rechtdoor
7	7	1	1	8	1	2	Rechts
8	8	1	2	9	2	2	Rechts
9	9	2	2	10	2	1	Omdraaien
10	10	2	1	11	2	2	Rechtdoor
11	11	2	2	12	2	3	Links
12	12	2	3	13	1	3	Stop

Opmerking : "rechts" betekent "draaien met de klok mee", "links" betekent "draaien tegen de klok in" etc.