ROBO-BITS-56

Jaargang15, nummer 1, maart 2012

Afz.hcc Robotica gg, p.a. Henk de Gans,Koelmanhof 2 3861GG Nijkerk..

hcc robotica

Robobits is een uitgave van de hcc!robotica gebruikers groep, en wordt vier keer per jaar als PDF beschikbaar gesteld aan de leden. hcc! robotica is een onderdeel van de hcc! (hobby computer club), een vereniging van bijna 150.000 leden.

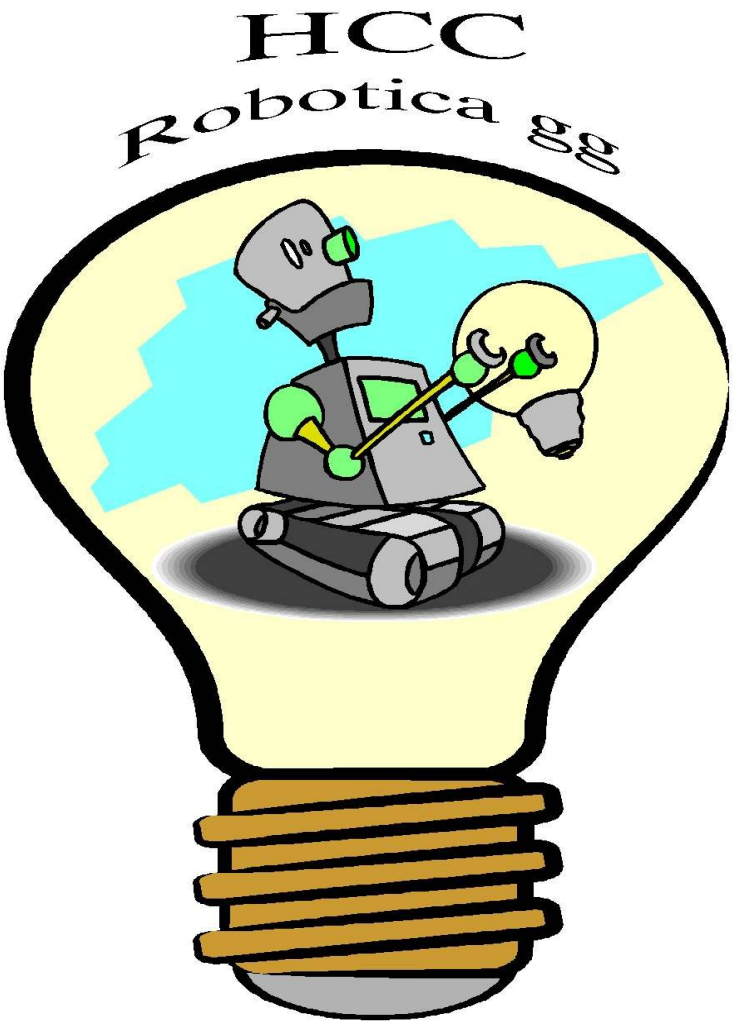
==========

Redactie adres: H.J. de Gans, Koelmanhof 2, 3816GG Nijkerk.
hj.de.gans@gmail.com Tekst aanleveren in WORD of platte tekst in ASCII.
Afbeeldingen los er bij in JPG, GIF of BMP formaat.

==========

Dagelijks bestuur:

Voorzitter:	E.F.O.Buzzi(Ed), Ed.Buzzi@net.hcc.nl
Technisch adviseur:	Z.Otten(Zeno), z.otten@chello.nl
Technisch adviseur:	H.M.P. van Sint Annaland (Hinnie) h.vansintannaland@xs4all.nl
Secretaris:	M.W.J. van Harmelen (Rien) r.van.harmelen@hetnet.nl
Penningmeester:	H.J. de Gans(Henk) hj.de.gans@gmail.com



inhouds opgave:

REDACTIE

Geachte lezer,

Voor u ligt al weer de 56 ste Robobits! Een beetje verlaat, zoals u wellicht gemerkt hebt. Maar er was weinig kopij, en ik had weinig tijd! Maar op de valreep kreeg ik nog wat kopij, dus hier is hij dan weer. Ik wens u veel leesplezier, en bedenk: ZONDER OOK UW INBRENG, GEEN ROBOBITS!!!!

Henk de Gans
redactie

==========

Denk ook eens aan onze sponsor, als u componenten nodig hebt!!

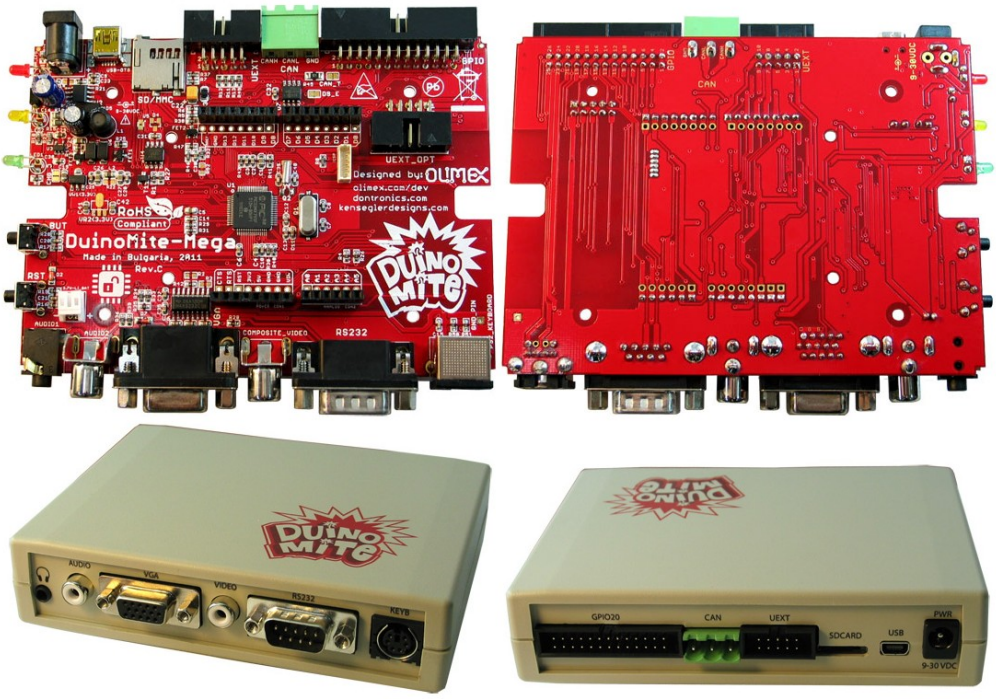
Duinomite (B+ op herhaling)

Alleen heel oude mensen kunnen het zich nog herinneren, B-plus (of: B⁺).
Even ter herinnering en voor wie het nieuw is: lang voor het PC tijdperk waren er spelcomputers zoals Commodore, NewBrain, Atari, ZX80/81, Acorn, MSX en vele anderen.
Toen kwamen er micro controllers, eerst duur en eenvoudig, maar naarmate de tijd verstreek werden ze goedkoper en uitgebreider en, 'heel belangrijk' beschikbaar voor de hobbyist.
De meest gebruikte architectuur was gebaseerd op de 8080 processor. Aanvankelijk met dure compiler software. Alleen bestemd voor de PC, dus wie rijk was kon aan de slag.

Ton Goossens en Frank Mensert uit Zoetermeer bedachten dat dit anders moest, zij ontwikkelde een micro controller bordje met 'embedded' compiler.
Je kon het geheel in 'platte' ASCII aanspreken, vrijwel elke game computer kon dit.
Vervolgens werd het programma gecompileerd en je het draaide.
De programmeertaal, die eveneens B⁺ heet, houdt het midden tussen assembler en basic.

Het B⁺ bordje bevat een aantal digitale 8-bits poorten waarmee de actuatoren en sensoren worden bediend. Dit werkt tot op de dag van heden nog steeds tot volle tevredenheid.
In de loop der jaren is dit bordje ook door andere HCC groeperingen omarmd, onder andere de Westland µC en Fort interessegroep hebben veel bijgedragen aan de populariteit van B⁺.
Maar zoals veel in computerland, raakte ook het B⁺ bordje in de vergetelheid.

Arduino is momenteel 'het' populaire bordje, maar ook anderen doen hun best in µC-land.
Olimex heeft de taal BASIC herontdekt en als interpreter ingebouwd. Een video uitgang en je kunt programmeren zonder dat er een PC nodig is. Natuurlijk blijft het mogelijk om met C++ compiler en PC op de vertrouwde toer programmeren.
Het bordje Duinomite Mega is gebaseerd op de PIC 32 kloksnelheid 80 MHz
Het bordje kost ca € 36,- en een passende behuizing heb je voor € 12,-
Er zijn drie varianten, die in prijs niet veel verschillen, dus kiezen we voor de Mega.



De interpreter maakt de boel traag, maar zeg nou zelf, is die hoge snelheid altijd wel nodig?
Wil je snel, dan kan je altijd nog met C++ en compiler programmeren.

Sluit je de USB aan op de PC dan kan je met een terminalprogramma communiceren.
De USB aansluiting kan met aanwezige software worden gebuikt als muis of toetsenbord.
Maar is ook in te stellen zodat er van alles op de Duinomite kan worden aangesloten.
Er is een standaard RS232 poort, en voor uitbreidingen is er de I^2C poort
Je kunt standaard Arduino printen op de uitbreidingspoort aansluiten, maar dan kan het kastje helaas niet meer dicht…
Het kastje heeft voorgeboorde (schroef)gaten, waardoor je het bordje eenvoudig kan plaatsen. Sluit het PS2 toetsenbord en VGA monitor aan en de basic interpreter staat paraat.
De CAN interface wordt veel in de verlichting (disco en toneel) wereld gebruikt.
Er is een PDF handleiding te downloaden.
Er zijn veel Duinomite modules, waardoor je het wiel niet meer zelf hoeft uit te vingen

Even de gegevens op een rijtje:
PIC 32 met 128 KB ram en 512 KB flash geheugen, werksnelheid is al genoemd, 80 MHz.
Voeding van 9V tot max 30V DC
USB aansluiting kan als host en server dienen
Slot voor mini SD kaartje
CAN bus aansluiting
GPIO aansluiting (wat dat ook moge zijn)
Arduino shield connector
PS2 toetsenbord aansluiting
RS 232 aansluiting
VGA aansluiting
Geluid uitgangsplugje (tulp) en 3,5 mm oortelefoon plugje
Composiet video uitgangsplugje (tulp)
Reset en diverse gebruikers knopjes
Drie status leds
Ingebouwde acculader voor LiPo batterijen

Voor meer info zie de Olimex site http://www.olimex.com/dev/duinomite-mega.html
En Antratek http://www.antratek.nl/Duinomite-Maximite-BASIC-computer.html

Veel plezier

Groeten, Dré Jansen

Bascom

Als tegenhanger van het C# geweld dat ons wordt ‘aangedaan’ een iets simpeler manier om toch je robot te programmeren. Het is een basic dialect: Bascom.
Je kan het gratis downloaden van MCSelekt.com. www.mcselec.com.
Er is een betaalde verzie, het verschil is de maximale grootte van het programma. (4KB)
Helaas kan het C# bordje van FEZmini NIET voor bascom worden gebruikt.
De cursus heeft in 2008 in het maandblad Elector gestaan, wil je het origineel lezen, dan moet je zoeken naar sep 2008 tot en met feb 2009
Elektor heeft ook een kompleet cursusboek inclusief Bascom software en datafiles uitgegeven:
Basiscursus Bascom AVR (ISBN 978-90-5381-094-1). Tot zo ver deze STER spot.
Voor de eerlijkheid en volledigheid moet ik dan ook nog het cursusboek voor C vermelden:
Microcontrollers en de taal C van Willem Dolman (ISBN 978-90-484-0835-1)

Bascom in het Nederlands

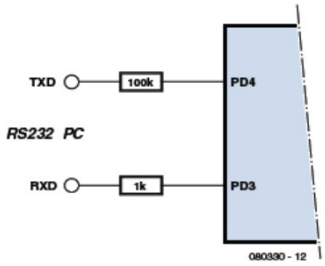
Hoewel ik best wel wat Engels ken, is het toch lastig om iets te leren, waarbij de uitleg niet in je moerstaal is geschreven. Elektor schreef in 2008/9 een Nederlandstalige cursus Bascom, dat is de basis van dit verhaal. Veel uitleg ontbreekt, enerzijds moedwillig, om het niet te ingewikkeld te maken, anderzijds: ik weet/snap ook niet alles.

Mijn bordje is het RN microcontroller bordje waarop een Atmega 32 is gemonteerd.
Kijk in de datasheet van jouw micro controller waar de pinnen en poorten zitten.
Elektor gaat uit van een mega 88, dus ook ik moest her en der wat aanpassen. Dat is goed te doen.

Enkele zaken worden meerdere malen vermeld, dat komt omdat dit verhaal niet in één avond is geschreven. Uiteraard kan ik die dubbele meldingen weghalen, maar wat is de 'schade' als je bepaalde informatie twee maal leest? Dus heb ik het laten staan.
Zijn er fouten, onwaarheden, vergissingen, etc. mail dat aan: drejansen.1@gmail.com.
Verbeteringen zijn altijd mogelijk, immers alles kan altijd beter. Ook dat graag mailen.

UART:

Dit is de seriële poort met RCx (PD0) en TXd (PD1).
Het signaal is TTL niveau, voor RS232 is een MAX232 nodig.
Op de PC gebruik je een terminal programma, zoals hyperterminal, maar de terminal van Bascom zelf werkt ook prima!
Dus twee pootjes van poort D (PD0 en PD1) zijn reeds bezet.
De meest simpele manier om van de Atmega naar de PC te gaan is met twee weerstanden. 1 K voor RXD in serie met pootje D3 en 100K voor TXD in serie met pootje D4.



COMMUNICATIE:

```
$regfile = "m32def.dat"      'instelling type micro controller
$framesize = 32              'instelling
$swstack = 32                'instelling
```

```
$hwstack = 32                'instelling
$crystal = 16000000          'kristal frequentie  (16 MHz)
$baud = 9600                 'communicatiesnelheid terminal
Sound Portd.7 , 800 , 450    'piepje (op RN bordje zit een luidsprekertje op port D7)


Do                           'begin van de lus
   Print "hallo"             'print de tekst 'hallo' op het scherm
   Waitms 2000               'wacht 2 seconden, (tijden opgeven in milliseconden)
Loop                         'terug na do voor herhaling
End                          'einde   (elke twee sec wordt 'hallo' geprint)
```

REKENEN:

Communicatie van de PC naar de MC en van MC naar PC.

```
$regfile = "m32def.dat"      'instelling type micro controller
$framesize = 32              'instelling
$swstack = 32                'instelling
$hwstack = 32                'instelling
$crystal = 16000000          'kristal frequentie  (16 MHz)
$baud = 9600                 'communicatiesnelheid terminal
Sound Portd.7 , 800 , 450    'piepje tijd, toon.   800 miliseconden, toon 450
print "{012}"                'clear screen
```

Bij het RN bordje zit op pootje D7 een luidsprekertje. Met het piepje geeft aan dat de boel is gestart. Gedurende 800 ms laat ik een toontje van 450 ?? eenheden horen.
Welke eenheid dit is, weet ik niet, vermoedelijk een tijdseenheid want een hoger getal geeft een lagere toon. Ook het volume varieert, en bij verandering van de toon wordt ook de tijdsduur dat de toon klinkt beïnvloed.
Het hierboven geschreven instel-deel komt telkens weer terug, laat ik verder achterwege.

```
Dim A As Byte                'variabele tussen 0 en 255
Dim B As Word                'variabele tussen 0 en 65000 (alleen positief)
Dim C As Single              'variabele tussen -zeer klein tot + zeer groot

Do                           'begin programma
   Print " input 0...255"    'er wordt een getal tussen 0 en 255 gevraagd
   Input A                   'dit getal wordt in variabele A geplaatst (straal v.d.
cirkel)
   B = A * A                 'inhoud is A x A x 3,14
```

Het is in Bascom **NIET** toegestaan om meerdere vermenigvuldigingen binnen een opdrachtregel te plaatsen, je moet dus een extra hulp variabele gebruiken.
Hier is dat variabele B waarin A x A wordt opgeslagen

```
   C = B * 3.1415            'met PI vermenigvuldigen, geeft cirkel oppervl in
variabele C
   Print "oppervlakte is: "; C  'het resultaat wordt afgedrukt
Loop                         'herhaal
```

Aanvulling op rekenen:
De straal in het kwadraat maal PI, geeft de oppervlakte, met een heleboel cijfers achter de komma.
Door C = round(C) wordt het getal afgerond .5 naar boven en .4 naar onderen
Wil je één cijfer achter de komma hebben, dan vermenigvuldig je met 31,4 ipv 3,14.
Na afronden deel je nogmaals door 10, waardoor je één cijfer achter de komma krijgt.
Dus:
```
C = B * 31.4
C = round(C)
C = C / 10
print "het oppervlak is "; C
```

EXTRA COMMUNICATIEPOORT

Veronderstel dat je een LCD op RS232 wil aansluiten.
Dan kan je PD3 als poort nummer 2 uitgang gebruiken.
Om data naar buiten te sturen schrijf je print #2, "hallo"
Door het signaal te inverteren (inverted) kan je het buffer weglaten, in rust is het signaal dan laag.

Voorbeeld met de verderop gebruikte spanningsmeting, hier is de tweede uitgang gebruikt.

```
        Baud = 9600                                  'transmissiesnelheid 9600 baud
I       Open “comd.3:9600,8,n,1” For Output As #2    'poort nummer 2 zonder inverteren
II      Open “comd.3:9600,8,n,1,INVERTED” For Output As #2  'poort nummer 2 met inverteren
        (Je gebruikt dus één van de twee regels, òf regel I òf regel II)
        Config Adc = Single , Prescaler= 64 , Reference = Off       'uitleg bij spanningsmeting.

        Do                                   'start van de lus
          U = Getadc(0)                      'variabele U met het omzetgetal (ADC delen) laden.
          U = U * 5                          'vermenigvuldigen met 5 (referentiespanning)
          U = U / 1023                       'delen door 1023 (1024 stappen lopen van 0 tot 1023)
          Print #2 , “ADC(0) = “ ; U ; “ V”  'printen naar poort -2- de tekst: ADC(0) = de waarde U
                                             'en de letter V voor de eenheid Volt.
          Waitms 500                         'halve seconde wachten
        Loop                                 'opnieuw, nieuwe meting
```

POORTEN

Bij opstarten zijn alle poorten ingangen.
Wanneer een hoge uitgangspoort aan een laag ingangssignaal is aangesloten, dan heb je kortsluiting, dat is ongezond voor de levensduur van deze uitgang.
Is de poort (standaard) ingang, dan is een hoge of lage aansluitwaarde nooit een probleem.
Poorten kunnen uitgang, ingang zijn. Met en zonder pull-up weerstand werken (3 registers)

DDR data direction register daar geef je aan of de poort een ingang (0) of uitgang (1) is
Config PORTC = output laat de gehele poort als uitgang functioneren
Config PORTC = 240 maak van de 4 hoogste bitjes uitgangen.
config portC = 15 maakt van de laagste 4 bitjes uitgang.

DDRC = 255 maakt van de gehele poort C een uitgang zo is van elk bitje een in/uitgang te maken
Handiger is het om &B11110000 te gebruiken, dan zie je meteen welke pootjes ingang/uitgang zijn.
Config PORTC = input maakt alle pinnen tot ingang, maar dat waren ze al bij het opstarten.
één enkel pootje tot uitgang verheffen: DDRC &B00000001 of config portc = 1 voor het LSB bitje.

WEGSCHRIJVEN
één uitgangspootje hoog maken SET PORTC.7
Op de uitgang (MSB van poort C) komt nu +5V te staan (totempole uitgang)
Er is een begrenzingweerstandje, maar of dat voldoende bescherming bied bij volle sluiting??
Ik zou het risico niet nemen, dus voorzichtigheid is niet alleen de moeder van de porseleinkast.
Met RESET PORTC.7 wordt de waarde van dat ene bitje weer 0 gemaakt.
Om naar de hele poort te schrijven: PORTC = 0 alle pootjes worden 0.

```
        Config Portc = Output
        Do
I         Portc = 85              '  / hier staat drie keer hetzelfde
II        Portc = &H55            '<  maar telkens op een andere
III       Portc = &B01010101      '  \ manier weergegeven.
          Waitms 200              '200 ms wachten
          Portc = 170             'de ledjes zijn verwisselt
I         Portc = &HAA            '  / ook hier drie keer hetzelfde
II        Portc = &B10101010      '<  wisselende ledjes elke 200 ms
III       Waitms 200              '  \ weer effe wachte
        Loop
```

Het resultaat is knipperende ledjes
Van de µC zijn niet van elke poort alle pootjes beschikbaar, lees daarvoor de gebruiksaanwijzing van de betreffende chip. (datasheet)

UITLEZEN

Het uitlezen van de poort gebeurd via een latch register PINC.
Print PINC geeft de gehele poort weer.
Print PINC.1 geeft de stand van het tweede bitje aan (0 is het eerste bitje, 7 de laatste)
Je zou de waarde -2- verwachten, maar nee, de weergave is -1- of -0-, afhankelijk van zijn toestand.

De pull-up weerstanden, kan je per pootje in/uitschakelen, maar ook voor alle pootjes tegelijk.
DDRB = 0 alle bitjes van poort B zijn ingang.
PORTB = 0 pull-up weerstanden zijn uitgeschakeld (alle ingangen zijn hoogohmig)
PORTB = 255 alle pull-up weerstanden zijn ingeschakeld (weerstand ca 50 Kohm naar +)

Dat is met een universeelmeter te controleren, de onbelaste ingang(en) is (zijn) hoog.
Wanneer je de pull-up weerstand uitschakelt dan is de ingang laag, ‘onbetrouwbaar’ is een betere omschrijving. De belasting van het meetinstrument is voldoende om de ingang laag te maken.
Uitlezen van PINB geeft bij aangesloten meter 0, bij ingeschakelde pull-up is het 1
Een open ingang (mét pull-up) is hoog.

```
Config Portb = 0        'poort B is ingang, dat is de standaard waarde
Do
  Print "{012}"         'cls scherm schoonmaken
  PORTB.0 = 0           'eerste pootje van poort B is hoogohmig
  Print "zonder R"      'pull-up weerstand is uitgeschakeld.
  Print Pinb.0 'waarde van de ingang wordt weergegeven.
  Waitms 500            'effe wachte

  Print "{012}"         'weer schoon scherm, om vergissingen uitsluiten.
  PORTB = 255           'zo worden alle pull-up's ingeschakeld, (alleen 1e wordt
uitgelezen)
  Print "met R"         'er is nu een weerstand van ca 50 ohm naar +
  Print Pinb.0 'geef de waarde van pootje 0 weer.
  Waitms 500            'effe wachte
Loop
```

Wanneer je deze ingang (pootje 0 van poort B) niet aansluit, dan geeft elke uitlezing 1.
Schakel je (naar 0) dan is uiteraard elke uitlezing laag. Maar... ik raakte een draadje met de blote handen aan, toen waren de hoogohmige uitlezingen ineens onbetrouwbaar.
Meestal waren ze 0, bij loslaten bleven ze even 0, even later waren ze weer 1
Blijkbaar zijn er parasitaire capaciteiten in het spel.
Hoe dan ook, kennis van wel of geen pull-up weerstand aan de ingang is belangrijk.

Op de poort kan je een led met voorschakel weerstand (naar massa) aansluiten.
In het Elektor schema staat een LED met 470 ohm in serie, maar zo laag durf ik niet te gaan.
Ik heb 1K weerstanden genomen, dan zie je het ook goed.

INGANG - UITGANG - PULL-UP

Condensator meting met gebruik van interne pull-up weerstand (van 1 nF tot 10 µF)

```
Dim T As Word               'verlopen tijd dual sloop meting
Dim C As Single             'gecorrigeerde condensator waarde
Do                          'start meetlus
  T = 0                     'tijd resetten
  DDRC.0 = 1                'maakt van poort C pootje 0 een uitgang
  PORTC.0 = 0               'schakel van poort C pootje 0 de weerstand uit
  Waitms 1000               'gedurende 1 seconde wordt de condensator leeg
getrokken
  DDRC.0 = 0                'maak van de uitgang een ingang
```

```
        PORTC.0 = 1             'schakel de pull-up weerstand in
        Do                      'tweede meetlus
          T = T + 1             'totdat pootje 0 van poort C de waarde -1- tijdteller
ophogen
        Loop Until PINC.0 = 1   'is afhankelijk van de kloksnelheid omslag bij ongeveer
2,5V.
        C = T * 0.0730          'gemeten ‘tijdswaarde’ omrekenen naar condensator
capaciteit
        C = Round(c)            'afronden van deze waarde (anders 5 cijfers achter de
komma)
        Print C ; " nF "        'afdrukken van de waarde in nF
      Loop                      'volgende meting
```

Hier zie je dat er slechts van één pootje de functie ingang/uitgang en de pull-up weerstand aan/uit wordt gezet.
Dat kan natuurlijk ook voor de hele poort in één opdracht. Dan laat je .0 weg en vul je 255 of 0 in.
Uiteraard is dit geen geijkte waarde, de pull-up weerstand varieert tussen 20 Kohm en 100 Kohm. Ook de klokfrequentie is mede bepalend voor de gemeten waarde.
Je kan ijken met een bekende condensator of vergelijken met een ‘echte’ capaciteitsmeetinstrument.

ANALOGE INGANG:

AD omzetter heeft een resolutie van 10 bits, da's 2^10 = 1024, afgerond komt dat op 5 mV per stap.
De Atmega 32 heeft 8 A-D convertors. Die moeten vooraf worden geïnitialiseerd.

Dat gaat als volgt: Config ADC = Single , Prescaler = 64 , Reference = Off
De A-D omzetter krijgt 1/64^{e} van de klokfrequentie. Bij 16 MHz is dat 250 KHz.
De interne referentiespanning is uitgeschakeld, er wordt gebruik gemaakt van een extern aangeboden spanning, aan te sluiten op pootje 32 AREF. Uiteraard nooit meer dan 5V= !
Je mag ook bv 2V= aansluiten, dan wordt 2V in 1024 stukjes verdeeld, waardoor je een nauwkeurigheid van 2 mV per stap krijgt.
Op het RN bordje kan je kiezen tussen 5V en 2,5V (jumpertje trekken = 2,5V)
Biedt de spanning zuiver aan, spoeltje, diode, condensator (tantaal is lekker snel)

Spanningsomzetting wordt vanuit meerdere plekken gevraagd, dus een subroutine is hier zinvol (heet hier: spanning)
Een subroutine moet in Bascom ook worden gedeclareerd: **Declare sub spanning**.
Bij een subroutine kan je argumenten meegeven, maar dat geeft extra belasting voor de processor.
Bascom staat dat wel toe, maar er is kans op fouten. Het is beter om globale variabelen te gebruiken
Een globale variabele is geldig in het gehele programma. Er zijn 8 AD convertors

SPANNINGSMETING

```
        Declare Sub Spanning          'subroutine spanningsmeting
        Dim N As Byte                 'hulpvariabele
        Config Adc = Single , Prescaler = 64 , Reference = Off
                                      'instellen van de A-D omzetter.
                                      'single is de vorm van de variabele waarin het ADC getal
                                      'wordt weergegeven.
        Start Adc                     'starten van de omzetter.
        Do                            'hoofdlus.
          N = 0 : spanning            'twee opdrachten op één regel
                                      '1e  N krijgt waarde 0 (analoge ingang -0-).
                                      '2e  de subroutine wordt aangeroepen. Bascom staat dit
toe.
          Print "ADC(0) = " ; U ; " V" "
                                      'printopdracht stukje tekst, berekende waarde U en letter
V(olt)
          N = 1 : spanning            'ook hier twee opdrachten de spanning van analoge
ingang -1-
          Print "ADC(1) = " ; U ; " V" "
                                      'printopdracht, hetzelfde als hiervoor, maar nu met ADC
1
          Print                       'schone regel printen lege regel tussen twee metingen.
          Waitms 500                  'even wachten (halve seconde)
        Loop                          'herhaal, volgende meting.

        Sub spanning                  'het berekenen van de gemeten spanning
          D = Getadc(n)               'de variabele wordt met het omzetgetal (ADC delen)
geladen
          D = D * 5                   'vermenigvuldigen met 5
          U = D / 1023                'delen door 1023 (1024 stappen lopen van 0 tot 1023)
        End Sub                       'return naar hoofdprogramma
```

Er wordt op twee analoge ingangen gemeten ingangen 0 en 1
Spanningmeting komt vaker voor dan je denkt, bij het uitlezen van vloer en afstand sensoren bijvoorbeeld. Vaak hoeft er niet naar een spanningswaarde te worden omgerekend.

Extra uitleg: config ADC
Config Adc = Single , Prescaler = 64 , Reference = Off
Het aanroepen van de analoge ingang.
Config ADC is het aanroepen van de analoog naar digitaal omzetter.
Deze A-D convertor is een 10-bits omzetter, dus 2^10 = 1024
Dat betekent dat de ingang waarde in 1024 deeltjes wordt weergegeven
Bij minimale waarde is dat 0 en bij maximale waarde is dat 1024
Gaan we uit van 5V (de maximale ingangsspanning op de referentiepin) dan is de resolutie op 5 mV per stapje.

De aangelegde spanning wordt in een evenredige delen van 0…1024 weergegeven.
2,5V zal dus de waarde 512 krijgen en wordt dan ook als 512 weergegeven.

De software moet dit naar de juiste spanningswaarde omrekenen.

Stel, je wilt maximaal 100V meten, je moet een weerstandsdeling maken.
100V moet maximaal 5V opleveren, dus (R1+R2)/R2, waarbij R2 over de ingang staat.
Stel, R2 = 10 Kohm, dan moet R1 (95/5) maal groter zijn = 190 Kohm.
Uiteraard kies je de dichtbijliggende handelswaarde.
Voor E12 en E24 is dit 180 Kohm waardoor de maximaal te meten waarde 95V is.
U = 5 x (180+10) / 10 = 95V.
De vraag: is het toegestaan om niet 100V maar 95 V als maximale waarde aan te houden?
Zo niet, dan moet er een grotere weerstand worden gekozen.
Uit de E96 reeks komt de 191 Kohm, dan is de maximale waarde 100,5V welnu, dat volstaat.

Voor de zekerheid ALTIJD nameten met een goede voltmeter, want de kans bestaat dat de weerstanden niet exact zijn.
Nu moet je dus die 100V in 1000 stukjes verdelen, dus de meetnauwkeurigheid is 100 mV
Een fout in de referentiespanning wordt dan ook 20-voudig weergegeven. De referentiespanning dient zéér zuiver te zijn. Dus: spoeltje, diode, (tantaal) elco en het liefst voeden uit een aparte bron.

STORING

Analoge waarden bevatten vaak een stoorsignaal van 50 Hz (periodetijd = 1/50 = 20 ms)
Als je de gemeten waarden over 20 ms weet uit te middelen, dan heb je de storing vrijwel gecompenseerd.

In het volgende verhaal voeren we 25 metingen uit, gedurende 20 ms.
De meting zelf heeft ook tijd nodig, hoeveel weet ik niet.
Volgens de ‘geleerde heren’ is 800 micro seconden ruim voldoende voor een meting.
Er is dan zelfs tijd om gemeten waarden bij eerdere metingen op te tellen.

Timer 2 wordt met een prescaler van 64 gebruikt: (16 MHz / 64 = 250 KHz)
Dus elke: (1/250 KHz = 4 micro seconden) een telpuls, tot de 8-bits teller (256 da's 0-255) vol is.
Wanneer je de timer 'voorlaad' met waarde 56 zal na 200 tikken een interrupt volgen.
200 µs x 4 = 800 µs Dus elke 800 µs een meting (interrupt)
Na 25 metingen ben je 25 x 800 µs = 20 ms verder.

De interrupt aanroepen worden geteld in de variabele: tikken
Na elke 25 metingen wordt de som (analoog) in gemiddeld gezet
Als ingangspoort wordt datapootje -0- van poort A gebruikt (AD0)

```
    Dim analoog As Word             'de gesommeerde waarde van elke meting.
    Dim gemiddeld As Word           'het gemiddelde van de gesommeerde metingen.
    Dim tikken as word              'het aantal interrupt tikken.
    Config Adc = Single , Prescaler = 64 , Reference = Off      'zie hieronder (uitleg
config)
    Config Timer2 = Timer , Prescale = 64                       'timer 2 met 64 bits
prescaler
```

```
        On Ovf2 meting                  'interrupt routine inschakelen
        Enable Timer2                   'timer inschakelen
        Enable Interrupts               'interrupts inschakelen

        Do                              'hoofdlus, het printen van medingen
          gemiddeld = gemiddeld / 25    'het berekenen van het gemiddelde uit 25
metingen
          Print gemiddeld               'afdrukken
          Waitms 100                    'effe wachte
        Loop

        Meting:                         'de interrupt routine
          Timer2 = 56                   'de voorsprong om precies op 800 us uit te komen
          Tikken = Tikken + 1           'tikkenteller (dit gebeurd dus 200 keer)
          analoog = analoog + Getadc(0) 'zie hieronder
          If Tikken > 24 Then           'er zijn 25 metingen gepasseerd (0-24 is 25)
            Tikken = 0                  'reset tikkenteller voor volgende meting
            gemiddeld = analoog         'de som van de metingen klaarzetten voor
bewerking
            analoog = 0                 'de optelling van de metingen resetten
          End If
        Return                          'terug naar hoofdlus
```

Uitleg config

Config ADC = single.

Single is het type variabele waarin de gemeten waarde wordt weergegeven (4 bytes)
Dit is dan ook het enige toegestane type.
De typen variabele Word of integer zouden ook passen, maar zijn helaas niet toegestaan

Config ADC = single, prescaler = 64
Voor het meten wordt dual sloop methode gebruikt, de meetfrequentie is kristal freq / 64
Beter is het om: prescaler = auto te zeggen, de compiler zoekt dan zelf het meest gunstige deeltal.
In de Bascom toelichting staat voor welke chip het beste deeltal is.

Config ADC = single, prescaler = auto, reference = off
Je kan en mag gebruik maken van de interne referentiespanning, die is 5V.
Reference = off, betekent dat de INTERNE referentiespanning is uitgeschakeld.
Extern kan je op pootje **32** een spanning (uiteraard niet meer dan 5V) aanbieden.
Geef je daar 2,5V, ligt het 10 bits bereik binnen die 2,5V je meet dan met 2,5mV nauwkeurigheid.
Vooral voor kleine signalen is dit een groot voordeel.

Start ADC is niet noodzakelijk, omdat door het configuratiecommando de ADC automatisch start.
Na: stop ADC moet wel weer worden gestart met ‘start ADC’. Zo zijn er meerdere functies in/uit te schakelen. Dit is vooral belangrijk bij batterij gevoede apparaten, elke functie kost energie.

14 ROBOBITS

analoog = analoog + Getadc(0)
In deze regel wordt de analoge waarde ingelezen en meteen bij de eerdere metingen opgeteld.
ADC is een 10-bits gebeuren, dus de laagste waarde is 0 en de hoogste waarde 2^10 = 1024
Bij een 5 V referentie wordt dus geen 5 of 5000 weergegeven, maar 1024
Hier moet je delen door 205 om voor de juiste waarde. Het is dus een kwestie van zelf omrekenen.

Wisselspanningen: veel rekenwerk.
Gemiddelde waarde, enkel of dubbelzijdige gelijkrichting, effectieve en gemiddelde waarde, etc.
Gewoon een weerstandsdeling maken en vergelijken met een goede voltmeter.
Waaruit je de deel/vermenigvuldigingsfactor bepaald. Dan is het ineens heel simpel.
Je moet toch controleren. In de Elektor cursus staat hoe je het allemaal kan berekenen.
Succes!

TIMER EN INTERRUPT

Een timer maakt gebruik van de interne klok
Een counter maakt gebruik van een externe klok voor timer-1 is dat pootje -1- van de chip.
Hierbij kan nog worden gekozen of er bij de opgaande of neergaande flank moet worden geteld.
Als de teller/timer volloopt, volgt er een interrupt.
De tellerstand kan ten alle tijden worden gelezen op **TCNT1**
De teller kan ook worden gebruikt voor PWM.
Er zijn twee 8 bits tellers (TCNT0 en TCNT2) en één 16 bits teller (TCNT1)

```
teller -1-
        Config Timer1 = Timer , Prescale = 256     'timer -1- telt kristalpulsen /256
        Start Timer1                               'niet echt nodig timer start al bij
configuratie
        Do                                         'start lus
          Print Timer1                             'afdrukken van getalletje,
          Waitms 200                               'pauze tussen twee weergaven
        Loop
```

Deze timer loopt van 0 tot 65535, doe je hetzelfde met timer 0 of 2 dan zie je dat de waarde niet hoger dan 255 komt. Voor teller 1 komt de overflow is pas bij 65535.
De klokfrequentie wordt gedeeld door 256, waardoor de klok 16000000/256 = 62500 pulsen per seconde krijgt.
65535 / 62500 = 1,05 seconde ruim elke seconde loopt deze teller over en genereert een interrupt.

Wil je weten hoe lang een opdracht duurt, dan meet je het verschil tussen twee tijdmeldingen.
Een tik duurt 1 / 62,5 KHz = 15,261 µsec twee waarden bijv. 30706 - 17864 = 12842
De tijd: 12842 x 15,261 = 196 ms

De instructie waitms wacht dus iets korter dan 1 milliseconde (verschil waarde zou 200 moeten zijn, niet 196) waitms is dus **NIET geschikt** voor nauwkeurige tijdsmetingen.

1 SECONDE MAKEN

Voor het opwekken van een nauwkeurige tijd is timer 1 iets 'te duur' een simpelere 8-bits timer volstaat. (Deze routine wordt verderop in de frequentiemeter toegepast.)
Hiervoor is timer-0 prima, na een interrupt springt hij naar een ISR (interrupt service routine)
Daarvoor hoef je geen 'echte' routine te schrijven TIM0_ISR is een label waarnaar gesprongen wordt.
De interrupt routine moet met return worden afgesloten en uiteraard zo kort mogelijk worden gehouden. (tijdens de interrupt kan er nóg een interrupt komen)

De 8-bits teller wordt gedeeld door 64, waardoor een klokfreq van 16000000/64=250 KHz ontstaat.
Hierdoor zal na elke 256 tikken een overflow optreden (8 bits lopen van 0 tot 255, da's 256 pulsen).
Hierdoor moet je de teller telkens laten starten met een voorsprongetje van 6.
Dus in plaats vanaf 0, begin je vanaf 6 te tellen tot 255, waardoor je precies elke milliseconde een interrupt hebt.

```
Dim Tikken As Word                      'aantal timer overloop interrupts
Dim Seconden As Word                    'het aantal gepasseerde seconden
Dim Seconden_oud As Word                'de vorige gepresenteerde waarde
Config Timer0 = Timer , Prescale = 64   'timer instelling

On Ovf0 secondeISR                      'spring bij een interrupt naar 'seconde'.
Enable Timer0                           'de timer moet worden ingeschakeld.
Enable Interrupts                       'de interrupt moeten worden geactiveerd

Do                                      'het hoofdprogramma: seconden
weergeven.
  If Seconden <> Seconden_oud Then      'wanneer er een seconde gepasseerd is, de
                                        'nieuwe waarde printen
    Print Seconden                      'weergave
    Seconden_oud = Seconden             'nu hoef je niet elke milliseconde een
waarde in
                                        'te voeren.
  End If
Loop

                                        'interrupt routine
SecondeISR:                             'label waarnaar bij interrupt wordt
gesprongen.
Timer0 = 6                              'de voorinstelling om op 1 ms uit te
komen.
Tikken = Tikken + 1                     'elke milliseconde is er een interrupt en
wordt
                                        'deze teller met -1- opgehoogd.
```

```
        If Tikken = 1000 Then                   'na 1000 ms wordt de secondeteller met
-1-
                                                'verhoogd.
          Tikken = 0                            'tikken(milliseconden)teller weer op 0
gezet
          Seconden = Seconden + 1               'het verhogen van de secondeteller.
        End If
        Return                                  'terug naar het hoofdprogramma
```

De teller is een onafhankelijk werkende bezigheid in de micro controller.
Zodra de timer wordt geïnitialiseerd gaat hij lopen.
Om er iets mee te doen, wordt dat met enable timer0 mogelijk gemaakt
Vervolgens wordt het interrupt mechanisme wakker gemaakt, met enable interrupts

In het hoofdprogramma staat geen 'vaste plek' waarbij naar de interrupt routine wordt gesprongen, dat wordt door de interrupt generator geregeld. Bij een interrupt wordt meteen naar de routine gesproken, ongeacht de positie van de program counter. Na de return wordt het hoofdprogramma weer vervolgd, uiteraard vanaf de plek vanwaar hij is vertrokken.

Het hoofdprogramma is het stukje tussen **do** en **loop**. Daarin wordt slechts gekeken of de seconden moeten worden geprint. Zodra er een verschil is, wordt de nieuwe waarde geprint, waarna oud en nieuw aan elkaar gelijk worden gemaakt. Het hoofdprogramma staat dus grotendeels te wachten.
Het eigenlijke werk gebeurd op de achtergrond, parallel aan het hoofdprogramma.

Een interrupt routine moet zo kort mogelijk worden gehouden, omdat er nóg een interrupt kan komen. Daarom is het beter de milliseconden teller (tikken teller) in het hoofdprogramma op te nemen.

```
        do
          If Seconden <> Seconden_oud Then
             Print Seconden
             Seconden_oud = Seconden
          End If
          If Tikken = 1000 Then
            Tikken = 0
            Seconden = Seconden + 1
          End If
        loop
```

Voor de interrupt routine rest dan:

```
        secondeISR:
          Timer0 = 6
          Tikken = Tikken + 1
        Return
```

Korter kan ik hem niet maken.

COUNTER EN INTERRUPT

Een counter is hetzelfde als een timer, bij een counter komt het kloksignaal niet van de interne oscillator (al dan niet via de prescaler) maar van een externe pin. Timer 1 heeft zijn counter aansluiting op pootje 2 van de chip dat is de 2e data ingang van poort B (dus B1).
In het Elektor voorbeeld gebruikt men een andere MC, eentje waarbij de counter ingang van timer 1 op poort D5 zit. Dus even opletten hoe dat bij jouw chip is georganiseerd.
Onderstaand testprogramma verduidelijkt het verhaal

```
Config Timer1 = Counter , Edge = Falling , Prescale = 1
Start Timer1
PORTB.1 = 1
Do
   Print Timer1
   Waitms 1000
Loop
```

Config timer1 = counter
Dat is duidelijk, hier wordt timer1 als counter toegepast.

Edge = falling.
De neergaande flanken worden geteld, dus als het een knopje zou zijn, dan is dit bij neergaande spanningsflank (schakelen naar GND aangesloten met een pull-up weerstandje)

Prescale = 1
Ofwel, er is géén prescale, elke binnenkomende puls wordt geteld.
Dit tellen gaat door tot de maximale waarde wordt bereikt. dan volgt er (indien ingeschakeld) een interrupt. Uiteraard schakel je dat in, anders is het zinloos om een counter toe te passen.

Ook hier is de kreet: 'start timer' overbodig, de counter start al bij de initialisatie ervan.
Deze teller (1) telt tot aan 2^16 = 65536 waarna een interrupt volgt en hij telt weer vanaf 0.
De tellers 0 en 2 tellen tot 256, waarna een interrupt volgt.
Sluit een knopje aan op pootje B1 en je kan pulsen geven, het schakelaartje dendert enorm, dus met enkele indrukken heb je al honderden pulsen.
Laat je de boel los, dan tikt de teller er lustig op los, open ingangen geven veel stoorpulsen
Ook hier is het dus van belang om in rust een gedefinieerde waarde aan te bieden.

Door de opdracht: PORTB.1 = 1 schakel je de pull-up weerstand van dit pootje in.
Waardoor alles veel stabieler wordt, de contactdender is er nog wel, maar dat is een ander verhaal.
Voor het ontdenderen (debounce) zijn legio hardware en software oplossingen bedacht.

FREQUENTIEMETER

De telleringang kan maximaal een frequentie van 4 MHz tellen, daarna gaat hij fouten maken.
Zoals elk instrument, kent ook deze frequentiemeter zijn grenzen.
Als timer kan het mechanisme de klok freq van 16 MHz aan, maar als counter maximaal 4 MHz.

Voor het meten van frequenties zijn twee timers nodig, de een als tijdbasis, de andere als teller.
Twee timers, dus eer zijn ook twee interrupts.

Timer -0- verzorgd de tijdbasis van 1 seconde, zie 1 seconde maken zoals eerder bij het stukje ‘een seconde maken’ is beschreven. Na elke seconde (timer-0) wordt counter-1 gereset. Daarna begint het tellen overnieuw voor de volgende meting.

Timer -1- wordt als counter ingezet, dit is een 16 bits teller.
Wanneer je een frequentie boven 65 KHz hebt, begint de teller weer overnieuw te tellen.
Dat kan je oplossen door korter dan 1 sec te meten, maar hier tellen we een hele seconde.

We maken een hoog en laag byte, waardoor je tot wel 2^32 = 4 GHz kan meten.
De grens is al bij 4 MHz bereikt, maar je bent dan wel voorbij de grens van 65 KHz.
We tellen het aantal malen dat de counter 'overloopt'.
Naderhand tellen we even zoveel maal 2^16 = 65536 bij de actuele waarde op.

FREQUENTIEMETER

(Aansluiten op portB.1)

```
      declare sub seconde              'het bouwen van een seconde (interrupt routine)
      declare sub tellerhoogdeel       'aantal malen overlopen van teller 1 (int routine)
      Dim laagdeel As Word             'het lage deel van de telling (laagdeel maken)
      Dim hoogdeel As Word             'het hoge deel van de telling (aantal volle
hoogdelen)
      Dim tikken As Word               'kloktikken voor het maken van 1 seconde
      Dim Frequentie As Long           'de gemeten frequentie (som van laag en
hoogdeel)
      Config Timer0 = Timer , Prescale = 64      'timer-0  voor tijdbasis
      Config Timer1 = Counter , Edge = Falling , Prescale = 1
                                       'timer-1 wordt als teller gebruikt telt aantal neergaande
flanken
      On Ovf0 seconde                  'als timer-0 overloopt, spring naar 'seconde'
      On Ovf1 tellerhoogdeel           'als counter-0 overloopt, spring naar 'teller
hoogdeel
      Enable Timer0                    'timer-0 inschakelen
      Enable Timer1                    'timer-1 inschakelen
      Enable Interrupts                'interrupts inschakelen
      PORTB.1 = 1                      'inschakelen van de pull-up weerstand.

      Do                               'hoofdroutine
        Print "{012}"                  'schoon scherm maken
        Print Freq                     'het weergeven van de gemeten frequentie
        Waitms 1000                    'wacht 1 seconde, voor de volgende weergave
      Loop

      seconde:                         'het bouwen van de tijdbasis
      Timer0 = 6                       'zoals is beschreven bij hoofdstuk timer
```

```
        Tikken = Tikken + 1                 'hartslag, elke miliseconde kloktik hoger
        If Tikken = 1 Then                  'na de eerste tik:
          Timer1 = 0                        'counter resetten voor nieuwe meting
          hoogdeel = 0                      'evenzo het aantal overlopen resetten (aantal
hoogdelen)
        End If

        If Tikken = 1001 Then               'na 1000 ms
          laagdeel = Timer1                 'laagdeel wordt geborgd in laagdeel variabele
          Frequentie = hoogdeel * 65536     'hoogdeel wordt berekend en in frequentie
bewaard
          Frequentie = Frequentie + laagdeel     'laagdeel wordt er bij opgeteld, de totale
frequentie.
          Tikken = 0                        'hartslag op 0 voor volgende meting
        End If
        Return                              'klaar met interrupt (terug naar hoofdroutine)

        tellerhoogdeel:                     'hier is de counter overgelopen
          Hoogdeel = Hoogdeel + 1           'het aantal malen dat de teller overloopt
(hoogdeel).
        Return
```

Subroutines dienen te worden gedeclareerd, voor interrupt routines is dit niet nodig.
Dit is hier wel gedaan, is wat netter en duidelijker.
laagdeel, en hoogdeel, de onderste 16 bitjes en de bovenste 16 bitjes van de uiteindelijke frequentie.
De hartslag elke milliseconden een tik. tot er 1000 tikken zijn gepasseerd, dan volgt een totalisering van telpulsen. De frequentie is het uiteindelijke resultaat.

Timer 0 wordt als tijdbasis, timer, ingesteld.
Timer 1 wordt als counter ingesteld

Bij **ovf0** wordt de seconde routine aangeroepen er is nu een milliseconde gepasseerd
Wanneer de boel net is begonnen, dus bij de eerste tel (ms) van de meting, worden hoogdeel en laagdeel gereset.
Timer 1 is de frequentie teller, dus het aantal neergaande flanken van de te meten spanning.
Als de tikkenteller de 1000 is gepasseerd, dus na een seconde, wordt het aantal malen overlopen van
counter 1 opgeteld bij de huidige tellerstand.
Het aantal malen overlopen wordt vermenigvuldigd met de hoeveelheid van de maximale tellerinhoud. (65536)
De tikkenteller wordt op 0 gezet en het verhaal begint overnieuw.
De telling loopt niet van 0 tot 1000, maar van 1 tot 1001
Bij **ovf1** is de teller overgelopen, dan moet het hoogdeel met -1- worden opgehoogd.
Interrupts worden aangezet en het interne pull-upweerstandje van 50 Kohm wordt ingeschakeld.
De hoofdroutine, die blinkt uit door eenvoud, geeft elke seconde de frequentie weer.

MOTORSTURING PWM

Een collectormotor kan je in snelheid regelen door de spanning te variëren, maar ook door de tijd dat de spanning is aangesloten te variëren.
Het variëren van de spanning behoeft geen uitleg, een serieweerstand is voldoende.

Bij pulsbreedte regeling (PWM) varieert de tijd dat de spanning aanwezig is.
De verhouding aan/uit of hoog/laag.
Hoe langer de hoogtijd, des te meer energie naar de motor, waardoor hij sneller zal draaien.

Voordeel van PWM regeling is, dat bij PWM het koppel (vrijwel) constant blijft.
Daardoor kan de belaste motor heel langzaam aanlopen, wat bij spanningregeling niet mogelijk is. De motor kan bij te lage spanning niet aanlopen, maar wel verbranden.

Sluit je het PWM signaal via een diode op een condensator aan, (uiteraard met een belastingsweerstand), dan kan je met een PWM signaal digitaal-analoog omzetting realiseren.
Zo hebt je een interne A-D omzetter en kan je heel eenvoudig ook een D-A omzetting maken.

Met timer-1- kan je TWEE onafhankelijke PWM signalen maken. (elk 8-bits breed)
PWM1a en PWM1b op de pootjes 1 en 2 van poort B PB1 en PB2 (het tweede en derde databitje)
Deze pootjes kan je via een weerstand op een LEDje aansluiten, of een motordriver IC (ULN2003.)

Een Atmega 32 (mijn controller) heeft 4 PWM kanalen, de mega 88, van Elektor, heeft er zes!

port B3 timer-0 oc0 uitgangsbitje poort B.3
port D5 timer-1 oc1A uitgangsbitje poort D.5
port D4 timer-1 oc1B uitgangsbitje poort D.4
port D7 timer-2 oc2 uitgangsbitje poort D.7 hierop zit bij het RN-bordje de luidspreker.

De pulsen worden volgens het 'fase correct' protocol opgewekt. dwz elke puls zit midden in de cycle. Voor motor regeling maakt dat allemaal niet zo veel uit.

PWM

```
Dim P As Word                                   'de variabele waarmee wordt vergeleken.
Config Timer1 = Pwm , Prescale = 8 , Pwm = 10 , Compare A Pwm = Clear Down ,
Compare B Pwm = Clear Down
```

Timer -1- heeft twee PWM kanalen, A en B die ook als 10 bits PWM generator kunnen werken dus tot 1023 tellen.
De prescale is 8 de teller wordt door 8 gedeeld, voor sneller stappen je kan ook sneller zijn door de pauze in te korten.
Om van die 10-bits mogelijkheid gebruik te maken moet je PWM = 10 benoemen

Compare A Pwm = clear down , compare B Pwm = clear down.
De teller **telt omlaag**, en vergelijkt ondertussen met de opgegeven waarde (P).
Zo kan je ook compare B Pwm = clear up meegeven, dan wordt er omhoog geteld.

```
Do                          'de hoofdlus
  For P = 0 To 1023         '10-bits teller, P loopt van 0 tot 1023
    Pwm1a = P               'generator A wordt vanaf 0 telkens hoger
    Pwm1b = 1023 - P        'generator B start hoog, (1023) en telt bij elke stap omlaag
    Waitms 5                'korte wachttijd
  Next P                    'nog een keer, volgende stap: 1 hoger voor A en 1 lager voor B.
Loop
```

Het gevolg is dat de led A steeds helderder wordt en plotseling donker. Oplopende zaagtand.
Led B begint helder, loopt langzaam naar donker en plotseling weer helder. Aflopende zaagtand.

Op de poort kan je een led met voorschakel weerstand (naar massa) aansluiten.
In het schema staat een LED met een 470 ohm weerstand in serie, maar zo laag durf ik niet te gaan.
met 1K weerstand zie je het ook goed.

DE WAVE, een golfpatroon van ledjes

```
Dim A As Single     'hulpvariabele voor sinusberkening van de wave
Dim B As Single     'hulpvariabele voor sinusberkening van de wave
Dim I As Byte       'de 60 verschillende helderheid standen van de ledjes.
Dim K As Byte       'de actuele helderheid van het betreffende ledje.
Declare Sub Wave    'berekende helderheid van het aangewezen ledje.

Config Timer0 = Pwm , Prescale = 8 , Compare Pwm = Clear Down
Config Timer1 = Pwm , Prescale = 8 , Pwm = 8 , Compare A Pwm = Clear Down ,
                                                Compare B Pwm = Clear Down
Config Timer2 = Pwm , Prescale = 8 , Compare Pwm = Clear Down

Do                          'hoofdlus
  For I = 1 To 60           'de ledjes varieren in 60 stappen van minimaal, maximaal, minimaal.
    K = I                   'de brandstand van het betreffende ledje begin-brandstand 0
    Wave                    'bereken de brandstand van het eerste ledje

    Pwm0 = Pwm              'schakel het ledje in de juiste sterkte aan, in de begin positie is dat 'uit'
    K = I + 15              'ledje 2 heeft een hogere brandstand
    wave                    'bereken de actuele helderheid.

    Pwm1a = Pwm             'ledje 2 inschakelen
```

```
        K = I + 30          'waarde voor ledje 3
        Wave                'bereken deze waarde

        Pwm1b = Pwm         'ledje 3 inschakelen
        K = I + 45          'vierde ledje
        Wave                'bereken waarde

        Pwm2 = Pwm          'schakel vierde ledje in (bij mij lukt dit niet, hier zit de
luidspreker)
        K = I + 60          'nieuwe waarde voor eerste ledje
       Wave                 'bereken

        Waitms 50           'effe wachten zodat alle ledjes even in deze stand branden.
        Next Pwm            'volgende ronde
      Loop                  'blijven herhalen
      Sub Wave              'PWM waarden berekenen.
        A = 6.1415 * K
        A = A / 60
        B = Sin(a)
        B = B + 1
        B = B * B
        B = B * 63
        Pwm = Int(B)
        If Pwm < 2 Then Pwm = 2
      End Sub
```

Hierboven de subroutine waarin de vergelijkingswaarden voor de ledjes worden berekend. Elektor heeft een rekenkundig centrum vol met hoogbegaafden, dat blijkt wel uit deze complexe berekening. Hoe komt men bijvoorbeeld aan de waarde 6.1415? Iets meer dan 6 maar zeker geen 7. Het kan iets met radialen te maken hebben, maar 2π rad is 6,28….
Bij de waarde K kan ik me wel iets voorstellen, in stappen van 15 (oorspronkelijk 10 bij 6 leds) worden de waarden telkens met 1 (for-next-lus I) verhoogd, waarna dit getal door 60 wordt gedeeld. Dit voel ik ergens wel aan. De waarde K loopt in stappen naar iets meer dan 360, de cirkelomtrek, waarvan de sinus wordt berekend. Allemaal voer voor wiskundigen.

Het uiteindelijk resultaat is de sinus waarde die het gewenste golfpatroon weergeeft.
Van de optelling, +1 snap ik niet waarom dat is.
Het onbegrip groeit met het kwadraat, waar ik ook al niet veel van snap.
Mijn snappertje knapt geheel af, bij de vermenigvuldiging met 63.
Het laatste stukje kan ik me wel voorstellen, zo veel cijfers achter de komma, dat kan je wel missen.
Er wordt niet afgerond, het deel achter de komma wordt eenvoudig weggelaten.

Wanneer je ledjes aansluit, dan zie je een prachtig sinus-vormige beweging. Zou je 6 ledjes hebben, dan lijkt het me nog mooier, maar ja, ik heb er slechts 3 (de 4e plek is de luidspreker).

Door een vergelijkingswaarde naar PWM0 / PWM1A / PWM1B / PWM2 te sturen krijg je een PWM signaal, op achtereenvolgens: B3 / D5 / D4 / D7

In plaats van een ledje feller of minder fel te laten branden, kan je ook een motor sneller of trager laten draaien.

Groeten, Dré Jansen

nogmaals PWM!

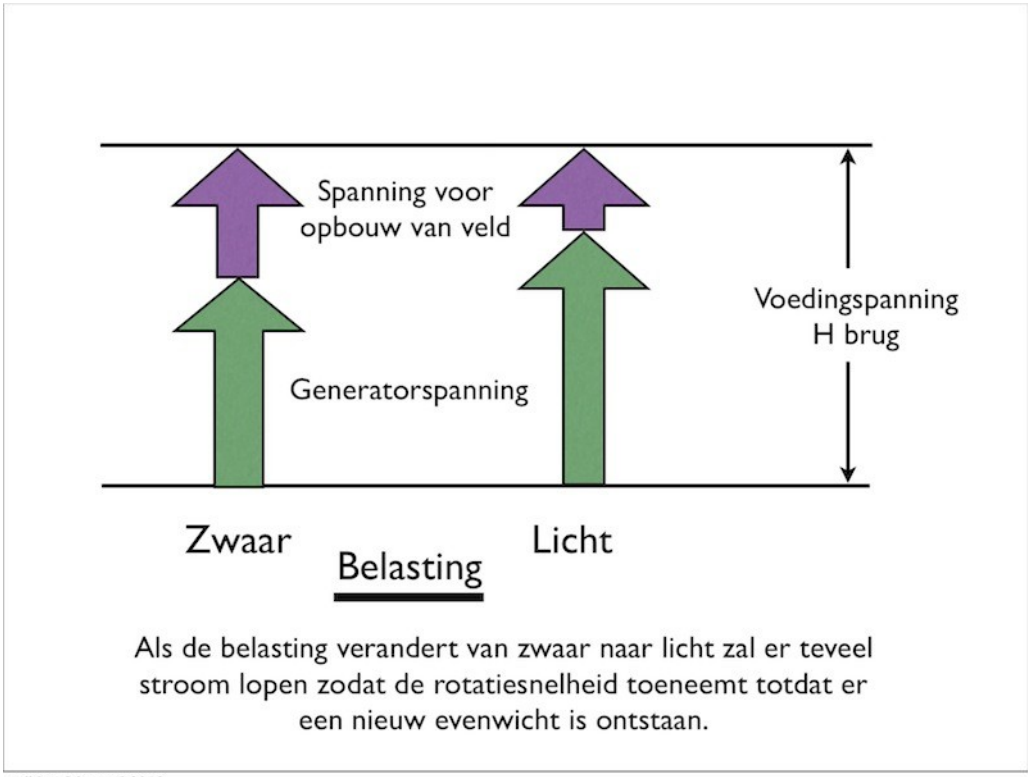
Bij een aantal leden van de groep is nog enige onvrede over het antwoord op de vraag wat Pulse Width Modulation nu eigenlijk echt is. Dit ondanks mijn artikel in ROBOBITS-52 waarvan toegegeven moet worden dat het wellicht niet overal even duidelijk is en zeker niet volledig. Dit is een poging de theorie van het onderwerp vergaand te behandelen.

Uitgangspunten zijn: het gaat over motoren van hooguit 24 volt en 1 amp□re met permanente magneten en een commutator. Bovendien moet men bedenken dat zo een motor op STROOM werkt want: Een stroomvoerende draad in een magnetisch veld ondervindt een kracht die haaks op het magnetisch veld staat. En dat noemen we een motor. En in een draad die haaks op een magnetisch veld wordt bewogen ontstaat een elektrische spanning. En dat noemen we een generator. Maar bovendien blijkt nu dat er geen enkel verschil bestaat tussen een motor en een generator. Nog iets: een zelfinductie is gemaakt van gesoldeerd metaaldraad, gewonden rond een kern. En metaaldraad heeft altijd weerstand dus een zelfinductie is de serieschakeling van een zuivere zelfinductie en een weerstand.

Nu een vereenvoudiging: de zelfinductie in een motor wordt als konstant beschouwd. Dit is in werkelijkheid niet waar want in een driepolige rotor worden steeds wisselend twee delen van de rotor in serie geschakeld en een deel van de rotor apart gebruikt. Dit maakt voor het betoog weinig uit.

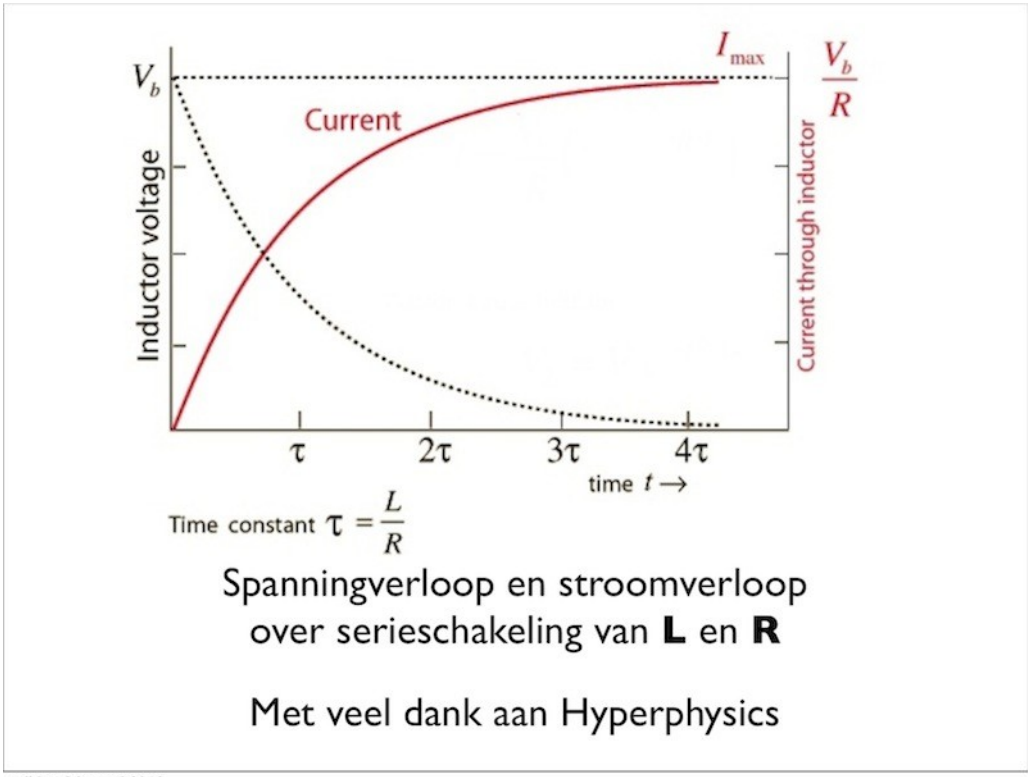
Daar gaan we.
Punt een: waarom draait een motor zo snel als hij draait?

Dit lijkt een onzinvraag maar is het niet. Stel, we zetten een spanningbron van 12 volt op de genoemde motor in onbelaste toestand. Er gaat stroom lopen en de motor draait. En wel met een toerental dat overeenkomt met: Voedingspanning = Generatorspanning+Spanning over de weerstand van de rotor want bij draaiende motor ontstaat tevens de spanning van de generator! Zie tekening ook voor belaste motor.

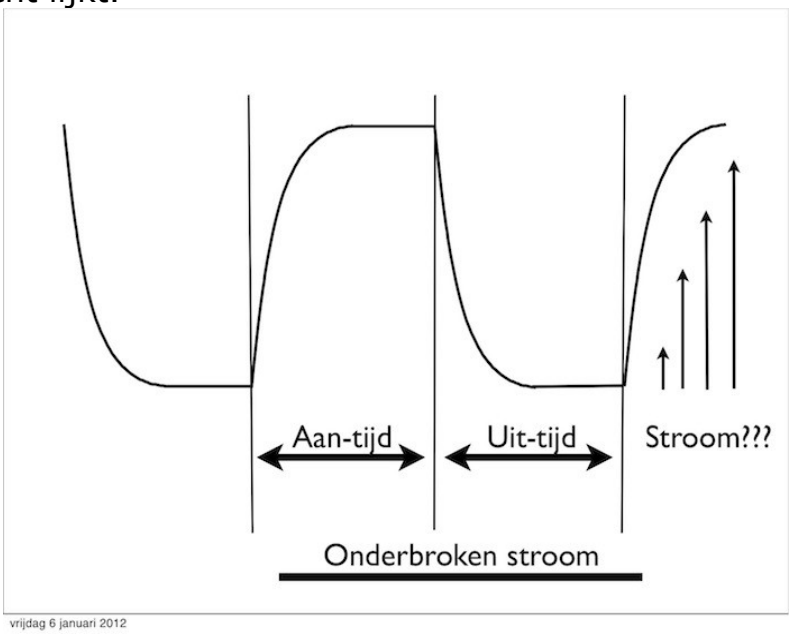


Dan punt twee:
Van de Hyperphysics site is bijgaand plaatje geplukt. Wat hier staat is het stroomverloop door een zelfinduktie -plus onontkoombare serieweerstand- als functie van de tijd. De stroom begint bij nul en loopt op tot het maximaal haalbare, begrensd door de weerstand. Zover willen we het niet laten komen want dat is de situatie waarin de motor geblokkeerd is en dus niet meer draait.

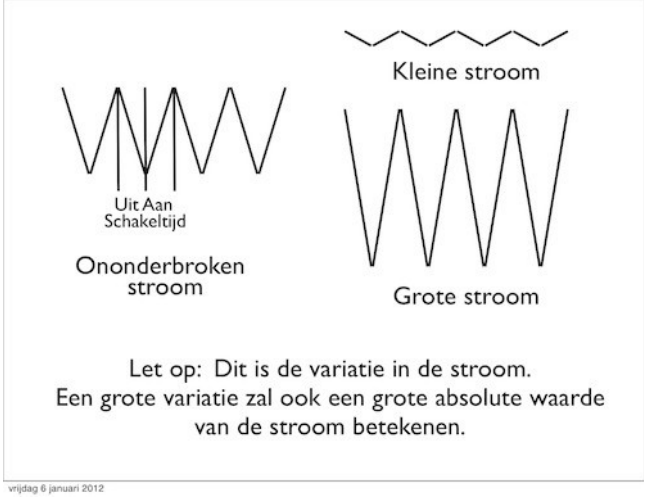
Wat we wel willen is dat de stroom bij draaiende motor ergens tussen bijna nul -dit kan niet anders want een onbelaste motor moet vanwege verliezen toch enige stroom opnemen- en ergens richting maximaal ligt, afhankelijk van de belasting. Hoe kunnen we nu de stroom variabel maken maar ook ononderbroken? Door te schakelen! Maar dat moet wel goed gebeuren.



Als er geschakeld wordt gelden de volgende plaatjes. De toename van de stroom heeft hetzelfde verloop als de afname met dit verschil dat de afname ge•nverteerdd is. Beide curves zijn achter elkaar gemonteerd met zelfs een tijd van max stroom ertussen en als laatste enige tijd minimale stroom. Echter, dit is zeer ongewenst. De motor gedraagt zich veruit het beste als de stroom door de rotor konstant is. Dit wordt bereikt door de spanning aan en uit te schakelen, hoe merkwaardig dit op het eerste gezicht wellicht lijkt.



Hier nog een keer de vorige curves met daarin aangegeven wanneer de spanning wordt aan en afgeschakeld. Op de verticale as wat de bijbehorende stroom is. Het blijkt nu dat de stroom tamelijk konstant is. En dat is wat we willen! Er is nu dezelfde situatie als in het begin met konstante spanning voeding.



En nu punt drie:
Als we op de goede manier gaan schakelen verandert de aangelegde spanning aan de motor met de puls-pauze verhouding. Dit betekent dat bij een puls van 95 procent zeg maar alle voedingspanning aanwezig is. En bij een puls van 5 procent zo ongeveer niets meer. Daar tussenin naar evenredigheid een deel van de terbeschikking staande voedingspanning. Een motor van dit type heeft een draaisnelheid die rechtevenredig is met de voedingspanning. Dat is prachtig maar dit is alleen geldig bij onbelaste motor. Als de motor belast wordt, neemt de draaisnelheid af. Als dus niet een zwaar overgemeten motor wordt gebruikt zal bij een 50-50 puls pauze verhouding de motor in het algemeen niet op de helft van de maximale snelheid draaien.

Tenslotte punt vier:
Gezien de zeer a-lineaire laadkromme van het magnetisch veld -wat dat is het- zal bij een variatie in de kleine puls pauze verhoudingen de variatie in de stroom betrekkelijk groot zijn. Daarentegen bij een grote puls pauze verhouding zal een kleine variatie geen noemenswaardig effect hebben. Dit betekent dat, zelfs als aan alle genoemde voorwaarden is voldaan, het niet zo zal zijn dat er een recht evenredig verband zal zijn van de rotatiesnelheid en de pulspauze verhouding.

Hoe bruikbaar is PWM dan eigenlijk?
In principe heel goed als je maar weet wat je doet. Het zal voor een echt goed resultaat echter heel wenselijk zijn een regeling toe te passen die uitgaat van de generatorspanning van de motor want die is recht evenredig met de rotatiesnelheid.

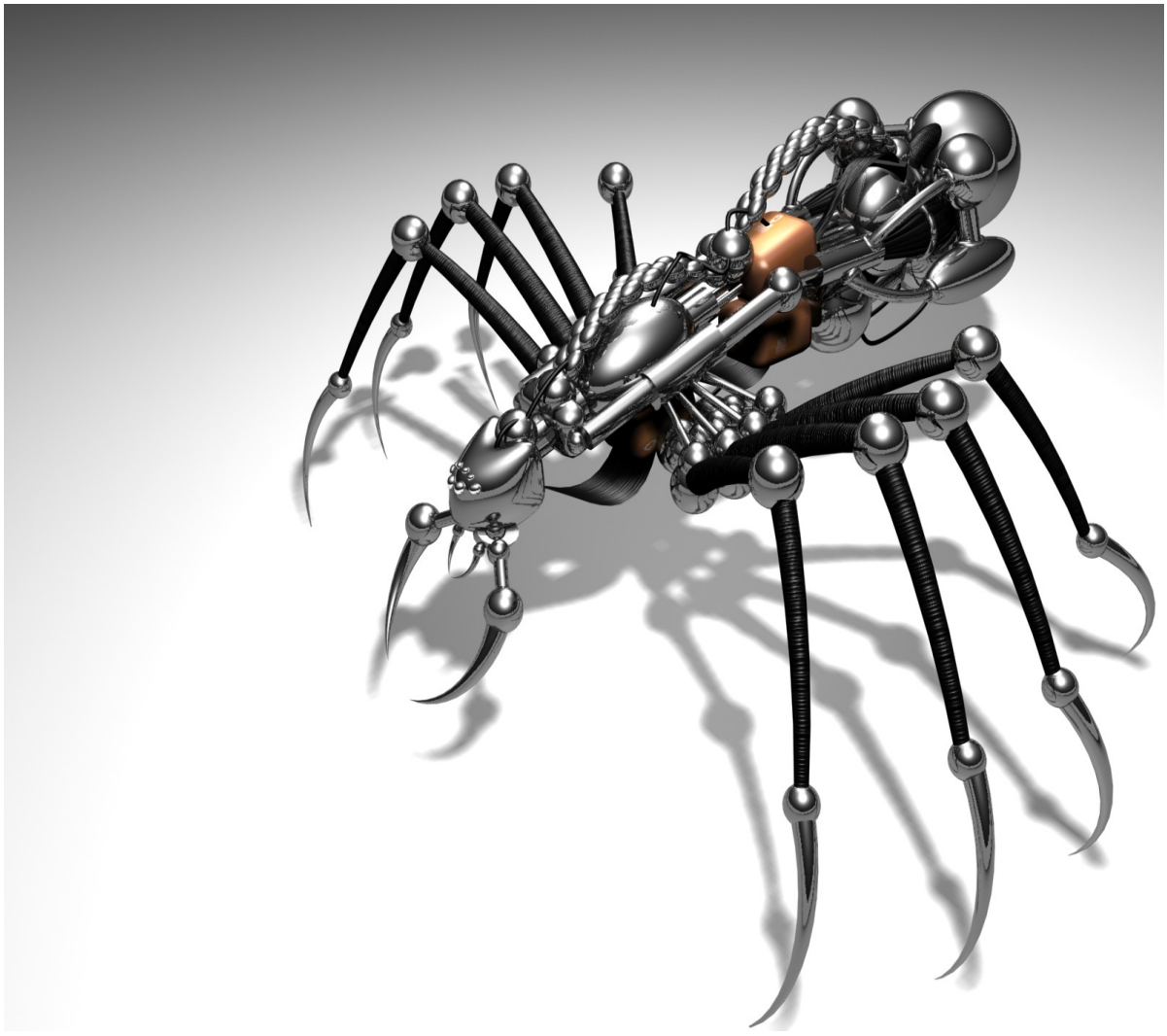
Wat men dan doet is steeds even de puls uitschakelen en de generator spanning meten. En hierop de puls pauze verhouding sturen. Of monteer een encoder op de uitgaande as en gebruik dat signaal voor de regeling.

Onderstaande referentie geeft een komplete motorregeling voor deze toepassingen. Iets om te overwegen als men beslist de snelheid van zijn robot goed onder kontrole wil hebben?

http://www.acroname.com/robotics/parts/S11-3A-EMF-HBRIDGE.html

Ik hoop dat dit alles duidelijkheid brengt in een onderwerp waar veel misverstand over bestaat.
Uiteraard ben ik altijd bereid om van gedachten te wisselen over dit alles op de bijeenkomsten.
Verder kan ik het niet laten enige reklame te maken voor mijn webpage: CONAPP.ORG

Jan Blok.



Kalmanfilter deel 2

Het Kalman filter (n.a.v. lezing van Joep in jan 2011)- Deel 2

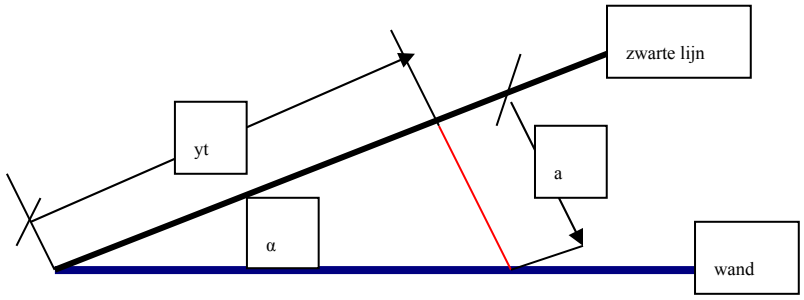
Door Rien van Harmelen **met dank aan Joep voor het voorbeeld en zijn opmerkingen**

- **Inleiding**

In een van de vorige aflevering (zie Robobits #53) is er een voorbeeld gegeven van een **stilstaande** robot die mbv een GP2D12 een afstand tot de wand meet waarbij het Kalman filter gebruikt wordt om de spreiding in de uitlezingen te middelen. We zullen nu een voorbeeld geven van een robot die over een (zwarte) lijn **rijdt** en waarbij wij geïnteresseerd zijn in de plaats van de robot op die (zwarte) lijn (tov bijvoorbeeld de oorsprong) .

- **Gegevens voorbeeld**

De zwarte lijn lijn maakt een hoek a met een wand.



Als de afstand (a) tot de wand (via bv een GP2D12) gemeten wordt dan kan yt (= afstand op de zwarte lijn tot de oorsprong) berekend worden uit :

$$yt = \frac{a}{\tan(\alpha)} \qquad (1)$$

Als wij dus de afstand tot de wand meten dan is het mogelijk de **locatie** van de robot op de zwarte lijn te bepalen. *Al **rijdend** blijkt/is, bij gebruik van het Kalman filter, dit niet zo'n nauwkeurige methode: de spreiding is groot.*

Als je een tijdje **stil zou staan** dan kun je de uitlezingen gaan middelen (via het gebruik van het Kalman filter) en dan krijg je wel een nauwkeurig positie op de lijn, de spreiding wordt minder. Dit is, overigens, de methode die in Robobits 53 besproken is.

- **De rijdende robot**

Maar nu naar een **rijdende** robot. Wij laten de robot met een bepaalde snelheid rijden over de zwarte lijn . Om de 0,5 s wordt de afstand a gemeten met een GP2D12. De afstand van de robot tot de oorsprong (= yt) wordt berekend uit formule (1). Zie Tabel 1 – kolom 4.

Als wij nu op deze waarden het Kalman filter loslaten (om de spreiding in de uitlezing te middelen) dan is het resultaat bedroevend. Zie Tabel 1-kolom 6 of Figuur 1. De schatting is gemaakt met de volgende parameters: Q = 1, R = 12, P0 = 1 en x0 = 47,7. De Kalman schatting (xt/t) blijft ver achter bij de waarden verkregen via de GP2D12 (yt)

Opm.
Voor de gebruikte formules zie Robobits #53

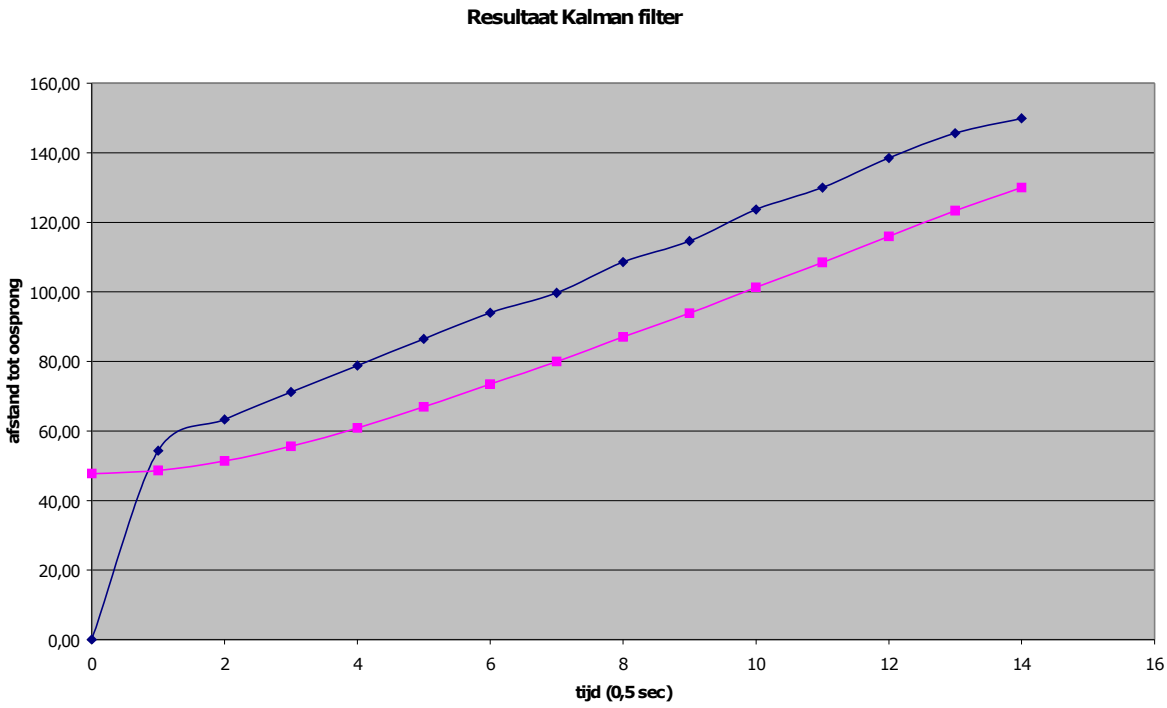
Vanwaar dit slechte resultaat:

- Het model;
- De betrouwbaarheid van het proces (gekozen is een waarde Q = 1).

P0 = 1
Q = 1
R = 12

	predict		update			
t (0,5s)	xt/t-1	pt/t-1	yt	Kt	xt/t	Pt/t
0	-	-	-	-	47,7	1
1	47,70	2,000	54,3	0,1429	48,64	1,7143
2	48,64	2,7143	63,3	0,1845	51,35	2,2136
3	51,35	3,2136	71,2	0,2112	55,54	2,5348
4	55,54	3,5348	78,8	0,2275	60,83	2,7305
5	60,83	3,7305	86,5	0,2371	66,92	2,8458
6	66,92	3,8458	94,0	0,2427	73,49	2,9124
7	73,49	3,9124	99,7	0,2459	79,94	2,9505
8	79,94	3,9505	108,6	0,2477	87,04	2,9720
9	87,04	3,9720	114,6	0,2487	93,89	2,9842
10	93,89	3,9842	123,7	0,2493	101,32	2,9911
11	101,32	3,9911	130,0	0,2496	108,48	2,9950
12	108,48	3,9950	138,5	0,2498	115,98	2,9972
13	115,98	3,9972	145,6	0,2499	123,38	2,9984
14	123,38	3,9984	149,9	0,2499	130,01	2,9991

Tabel 1



Figuur 1

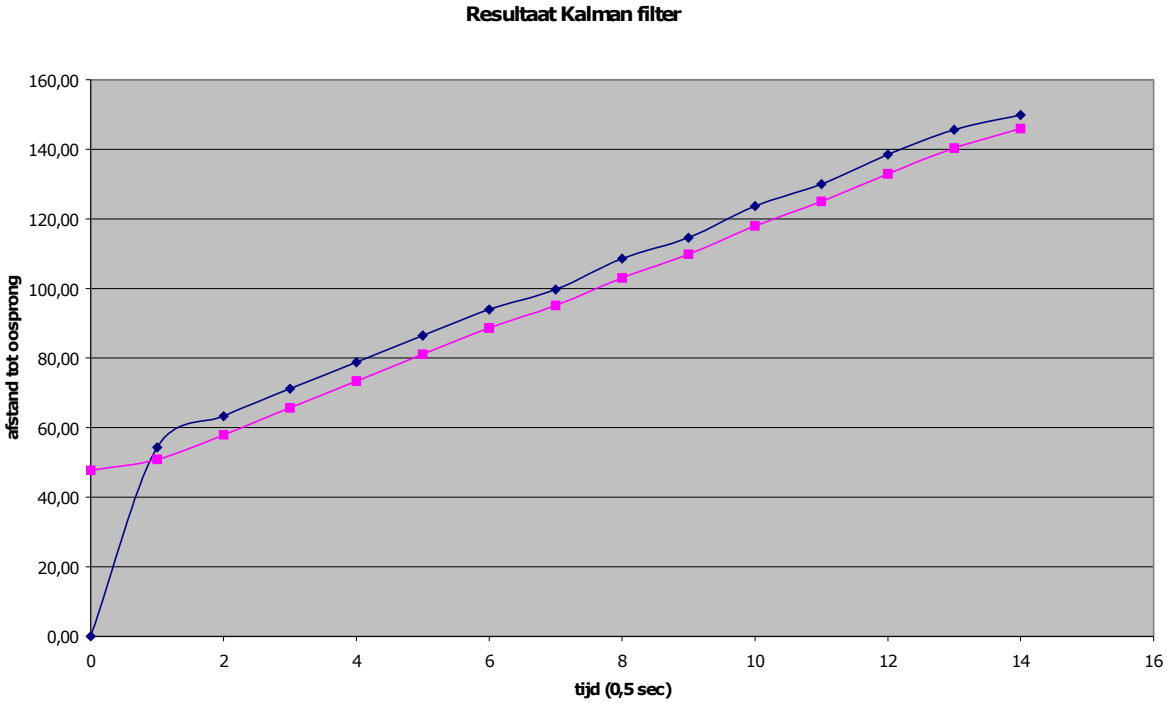
Als wij uitgaan van een Q = 10 (dwz een veel grotere procesfout) is het resultaat al veel beter. Zie Tabel 2-kolom 6 of Figuur 2

P0 =	1
Q =	10
R =	12

	predict		update			
t (0,5s)	xt/t-1	pt/t-1	yt	Kt	xt/t	Pt/t
0	-	-	-	-	47,7	1
1	47,70	11,000	54,3	0,4783	50,86	5,7391
2	50,86	15,7391	63,3	0,5674	57,92	6,8088
3	57,92	16,8088	71,2	0,5835	65,67	7,0015
4	65,67	17,0015	78,8	0,5862	73,37	7,0347
5	73,37	17,0347	86,5	0,5867	81,07	7,0404
6	81,07	17,0404	94,0	0,5868	88,66	7,0414
7	88,66	17,0414	99,7	0,5868	95,14	7,0416
8	95,14	17,0416	108,6	0,5868	103,04	7,0416
9	103,04	17,0416	114,6	0,5868	109,82	7,0416

10	109,82	17,0416	123,7	0,5868	117,97	7,0416
11	117,97	17,0416	130,0	0,5868	125,03	7,0416
12	125,03	17,0416	138,5	0,5868	132,93	7,0416
13	132,93	17,0416	145,6	0,5868	140,37	7,0416
14	140,37	17,0416	149,9	0,5868	145,96	7,0416

Tabel 2



Figuur 2

Maar..... een veel betere methode om de schatting te verbeteren is het aanpassen van het model.
Wij zouden de gereden afstand per tijdseenheid kunnen bepalen met bv odometrie op de wielen.
Odometrie geeft je geen absolute positie, maar een positie relatief ten opzichte van de voorgaande.
Het geeft je dus een voorspelling van de verplaatsing (= predict).

Via de odometrie maken wij (al rijdend) een "schatting" van de afgelegde afstand (= stap qt), ook weer met een tijdsinterval van 0.5 s. (zie Tabel 3- kolom 2). Als wij nu het Kalman filter gebruiken om de spreiding in de uitlezingen te middelen is het resultaat "niet slecht" In Tabel 3- kolom 7 en Figuur 3 is het resultaat gegeven. Voor gebruikte formules zie Hfdst. 4.

Opm:
De gebruikte formules zijn uiteraard identiek aan die gebruikt in Robobits #53 met dat verschil dat er nu rekening gehouden is met het rijden:

xt/t-1 = xt-1/t- wordt nu xt/t-1 = xt-1/t-1 + qt

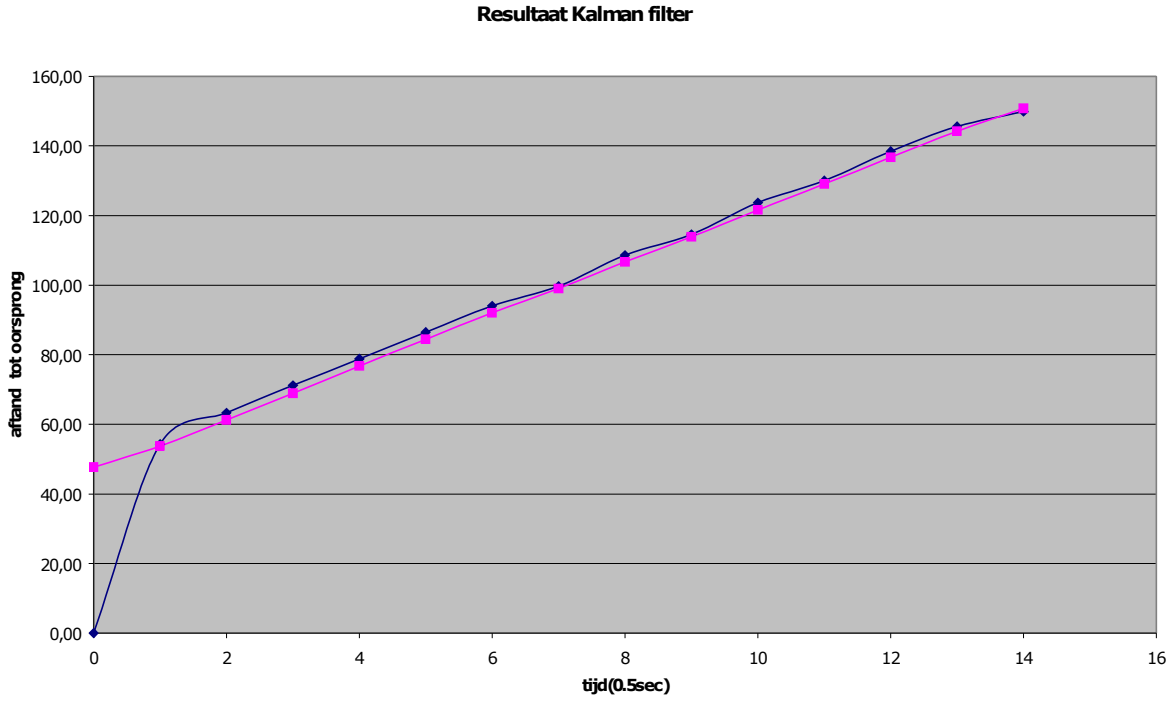
P0 = 1

Q = 1

R = 12

tijd (0,5s)	Predict			Update			
	stap qt	xt/t-1	pt/t-1	yt	Kt	xt/t	Pt/t
0		-	-	-	-	47,7	1
1	6,0	53,70	2,000	54,3	0,1429	53,79	1,7143
2	7,0	60,79	2,7143	63,3	0,1845	61,25	2,2136
3	7,1	68,35	3,2136	71,2	0,2112	68,95	2,5348
4	7,2	76,15	3,5348	78,8	0,2275	76,75	2,7305
5	7,0	83,75	3,7305	86,5	0,2371	84,41	2,8458
6	7,0	91,41	3,8458	94,0	0,2427	92,04	2,9124
7	6,8	98,84	3,9124	99,7	0,2459	99,05	2,9505
8	7,0	106,05	3,9505	108,6	0,2477	106,68	2,9720
9	7,0	113,68	3,9720	114,6	0,2487	113,91	2,9842
10	7,0	120,91	3,9842	123,7	0,2493	121,60	2,9911
11	7,1	128,70	3,9911	130,0	0,2496	129,03	2,9950
12	7,1	136,13	3,9950	138,5	0,2498	136,72	2,9972
13	7,0	143,72	3,9972	145,6	0,2499	144,19	2,9984
14	6,9	151,09	3,9984	149,9	0,2499	150,79	2,9991

Tabel 3



Figuur 3

Je kunt nu tot "lering en vermaak' een beetje spelen met Tabel 3.

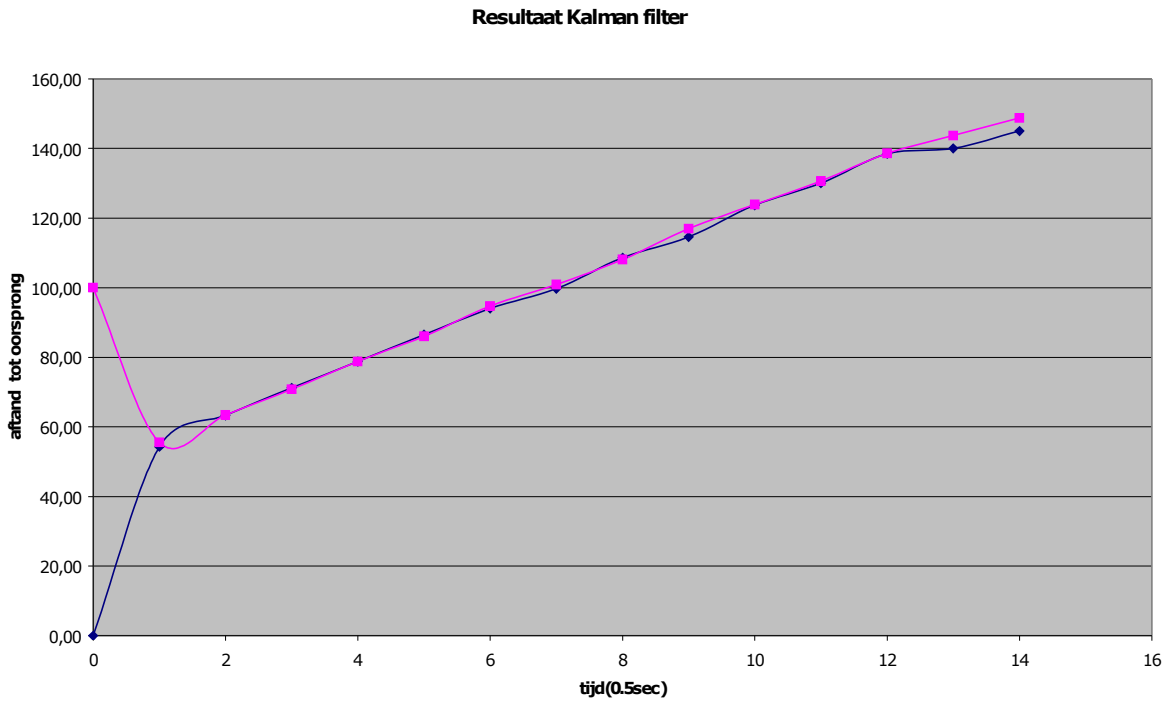
- Geef bv Po bv eens een zeer hoge waarde, maw je hebt weinig vertrouwen in de xt-waarden. Om het effect te zien kun je xo op 100 zetten , hoewel het weinig relevant is omdat je toch geen vertrouwen hebt in xt.
- Verander ook eens de twee belangrijkste parameters Q en R, beide het kwadraat van de spreiding (De spreiding is de grens, waarbinnen 2/3 van alle metingen liggen.
    1. Dus voor R = 12 (= 3.46^2) van de sharp GP2D12 hebben 2/3 van alle metingen van de sharp een afwijking van ± 3.46 cm;
    1. en voor Q = 1 (= 1^2) van de odometer hebben 2/3 van alle metingen een afwijking van ± 1 cm
    2. en voor Q = 2 (=1.41^2) geldt 2/3 van alle metingen hebben een afwijking van ± 1.41 cm);
- Veronderstel ook eens dat er slip op de wielen optreedt;

Zie Tabel 4 en Figuur 5

P0	500
Q	2
R	12

tijd (0,5s)	Predict			Update			
	Stap qt	xt/t-1	Pt/t-1	yt	Kt	xt/t	Pt/t
0		-	-	-	-	100,0	500
1	6,0	106,00	502,000	54,3	0,9767	55,51	11,7198
2	8,0	63,51	13,7198	63,3	0,5334	63,40	6,4012
3	7,1	70,50	8,4012	71,2	0,4118	70,79	4,9416
4	8,0	78,79	6,9416	78,8	0,3665	78,79	4,3977
5	7,0	85,79	6,3977	86,5	0,3477	86,04	4,1729
6	9,0	95,04	6,1729	94,0	0,3397	94,69	4,0761
7	6,8	101,49	6,0761	99,7	0,3361	100,89	4,0337
8	7,0	107,89	6,0337	108,6	0,3346	108,12	4,0149
9	10,0	118,12	6,0149	114,6	0,3339	116,95	4,0066
10	7,0	123,95	6,0066	123,7	0,3336	123,87	4,0029
11	7,1	130,97	6,0029	130,0	0,3334	130,64	4,0013
12	8,0	138,64	6,0013	138,5	0,3334	138,60	4,0006
13	7,0	145,60	6,0006	140,0	0,3334	143,73	4,0003
14	6,9	150,63	6,0003	145,0	0,3333	148,75	4,0001

Tabel 4



Figuur 4

4. Opmeringen

Voor degenen, die ook een beetje willen "spelen" met bovenstaande tabel, kunnen deze tabel als Excel file downloaden vanaf onze website.

Gebruikte formules :in de Tabellen 3 en 4

- Predict :
  $xt/t\text{-}1 = xt\text{-}1/t\text{-}1 + qt$ ..... Kolom 3
  $pt/t\text{-}1 = pt\text{-}1/t\text{-}1 + Q$ ..... Kolom 4
- Update :
  $xt/t = xt/t\text{-}1 + Kt\ (yt - xt/t\text{-}1)$ ..... Kolom 7
  $Kt = pt/t\text{-}1\ (pt/t\text{-}1 + R)^{-1}$ ..... Kolom 6
  $pt/t = (1 - Kt)\ pt/t\text{-}1$ ..... Kolom 8

Kolom 2 zijn de waarden gemeten met de odometer: qt
Kolom 5 zijn de waarden gemeten/berekend met de sharp: yt

PWM signalen in BASCOM-AVR

- **Inleiding**

In Bascom-AVR wordt de Timer1 in de Atmega- processor gebruikt voor het genereren van een PWM signaal. Met Timer1 heb je de beschikking over twee PWM signalen (Pwm1a en Pwm1b) met een resolutie van 8 of 9 of 10 bits.(In enkele (meer recente Atmega's) kan ook Timero en/of Timer3 gebruikt voor de uitvoer van een PWM signaal).

Wij gaan voor het verdere verloop even uit van een Atmega32

- **Genereren van het PWM signaal**

Met de volgende BASCOM-AVR instructie wordt een PWM signaal verkregen:
Config Timer 1 = PWM , PWM = 8, Compare A Pwm = Clear Down/Up,
Compare B Pwm = Clear Down/Up, Prescale = 8
Voor de "PWM = " kunnen de volgende waarden worden gekozen: 8, 9, 10
Voor de "Prescale = " kunnen de volgende waarden worden gekozen: 1, 8, 64, 256, 1024

Voor de Atmega32 zijn de PWM uitgangssignalen beschikbaar op de OC1A (PB1) en OC1B (PB2).

Hoe wordt nu dit PWM signaal verkregen ?
De voor de PWM signaal geconfigureerde Timer telt voortdurend van 0 tot een maximale waarde en weer terug. De maximale waarde wordt in de bovenstaande BASCOM instructie vastgelegd met de PWM waarde (8,9 of 10), dit zijn bits waarden dus wordt er geteld tot
$2^8 - 1 = 255$, $2^9 - 1 = 511$ of $2^{10} - 1 = 1023$.

De **snelheid** van tellen wordt bepaald door de kristal-frequentie en de Prescale-waarde.
Bv stel kristal-frequentie = 16000000 Hz = 16Mhz en een Prescale-waarde van 8 dan is de telfreqeuntie 160000000 / 8 = 2000000 Hz of maw iedere 1/2000000 sec wordt de Timer een tel opgehoogd . Stel dat PWM- waarde = 10, er wordt dus geteld van 0 tot 1023 en weer terug . De totale tijd die Timer hier over doet is 2 x 1023 * 1 / 2000000 ≈ 0.0010 s De periode van de Timer is dus 1 / 0,0010 = 977 Hz.

In onderstaande tabel zijn de perioden van de Timer gegeven als functie van de PWM-waarde en de Prescale-waarde voor een kristalfrequentie van 16 MHz .

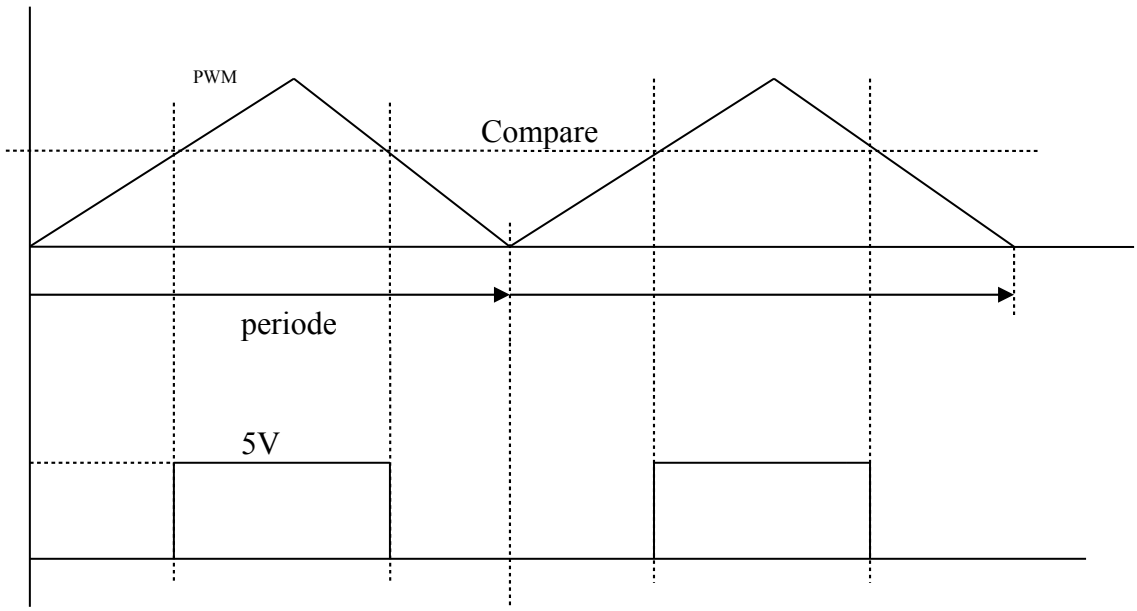
	Periode in Hz		
Prescale	PWM=8	PWM=9	PWM=10
1	31250	15625	7813
8	3906	1953	977
64	488	244	122
256	122	61	31
1024	31	15	8

Tabel 1

Nu terug naar het PWM signaal
In het compare register worden met een BASCOM-AVR instructie twee Compare waarden vastgelegd: Compare1a en Compare1b (ook wel Pwm1a en Pwm1b)

De Compare-waarde bepaalt bij welke waarde van de Timer overgegaan wordt tot actie. De aktie bestaat er bv hierin dat voor de Timer-waarde > de Compare-waarde de uitgangspin (PB.1 of PB.2) op 5 volt gezet wordt en bij waarde < de Compare-waarde op 0 V gezet wordt.

Het is ook mogelijk om te kiezen voor Timer-waarde > Compare waarde → 0 Volt
En Timer-waarde < Compare waarde → 5 Volt . In de BASCOM – AVR instructie wordt hier de (sub) instructie Clear Dowm of Clear Up voor gebruikt.



Figuur 1

Nu weer even terug naar Tabel 1.
Het is duidelijk dat de PWM- periode gelijk is aan de Timer- periode (zie ook bovenstaande Figuur)
Nu loopt het motortje niet (lekker) bij alle in Tabel 1 genoemde periodes. Een vuistregel kan hier niet voor gegeven worden..... 'Trial and error" net zolang tot het motortje soepel loopt

.