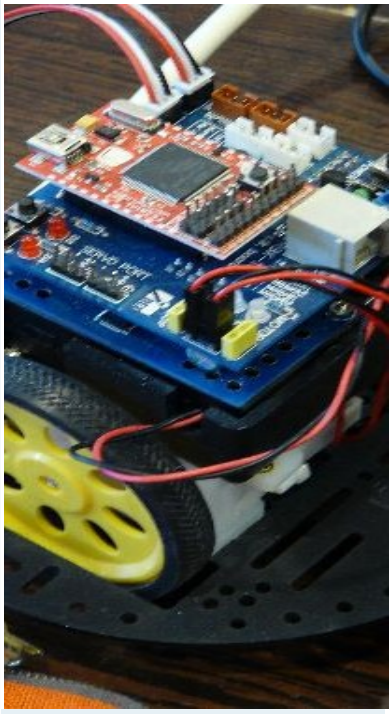
HCC ROBOTICA

September 2016
Jaargang 19
Nummer 3



ROBOBITS-74

VAN DE BESTUURSTAFEL

Beste lezer,

De zomer is weer voorbij, de herfst is begonnen. Dit zijn mooie tijden voor mensen die houden van robots bouwen. Dus niet eenzaam thuis voor de kachel zitten (al is dat op het ogenblik echt niet nodig).

Robots bouwen die mee kunnen doen aan onze robot wedstrijden.
Dit jaar organiseren we voor de 10e keer de ROBORAMA wedstrijden. Leuk om met z'n allen er een spektakel van te gaan maken.

Dit jaar zelfs een geheel nieuw onderdeel, maar daarover later meer.

Als we dat spektakel allemaal achter de rug hebben dan komen we ook nog eens met een schitterende cursus, die net als de vorige cursus (Arduino programmeren) een groot succes lijkt te gaan worden. Dit alles om onze kennis weer te vergroten en onze robots nog meer mogelijkheden te geven.

Of jullie nu wel of geen deelnemer zijn kom gerust naar de wedstrijden of ga je opgeven voor de cursus. Vertel mensen hierover en nodig ze vooral uit om in ieder geval een keer te komen kijken. Hoe meer zielen hoe meer vreugd!

Met vriendelijke groet,

Bert Berrevoets.

IN DIT NUMMER

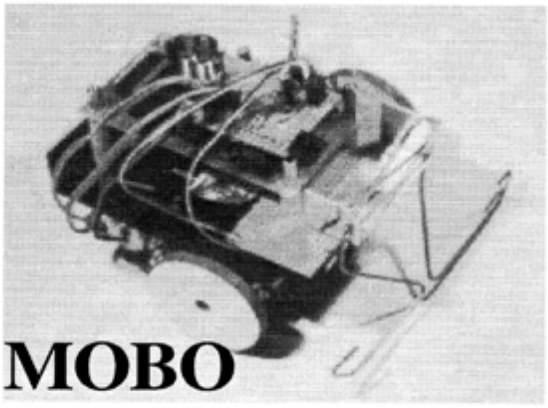
Hobot ramenlapper

RETRO ARTIKEL: Finite State Machines om een robot gedrag te leren.

Deze rubriek herhaalt een publicatie uit eerdere Robobits. Deze keer een artikel uit Robobits 7, september 1999 van Paul Wiegmans.

…. Wat we zouden willen is dat tijdens elk punt van het programma alle sensors uitgelezen worden en het programma daarop blijft reageren ……

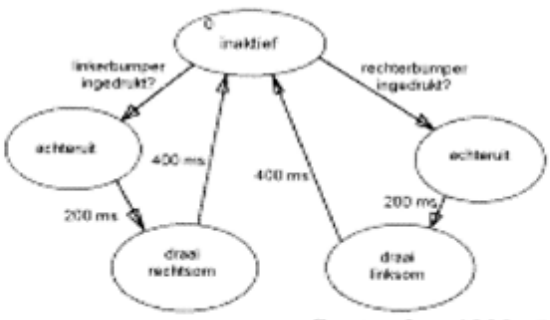
Finite State Machines om een Robot gedrag te leren



Het nadeel van dit programma is dat de robot alleen obstakels voelt wanneer hij vooruit gaat. Wanneer de rechterbumper contact maakt met een obstakel en programmatak B wordt uitgevoerd, dan wordt gedurende 600 ms geen enkele sensor afgelezen en is de robot totaal blind voor alle sensoren. Wat nu als de linkerbumper contact met een obstakel maakt wanneer het programma zijn afwerking voor de rechterbumper uitvoert? Het programma houdt geen enkele rekening met die mogelijkheid. Wat we zouden willen, is dat tijdens elk punt van het programma alle sensors afgelezen worden, en het programma daarop blijft reageren. Dit is nu mogelijk door de doorloop van de hoofdlus zo kort mogelijk te houden, en te voorkomen dat het programma voor een ontoelaatbaar lange tijd verzandt in een zijtak. Dit nu is mogelijk met behulp van een FSM.

Door het programma verloop voor te stellen in een toestanddiagram wordt duidelijk hoe de zaken ervoor staan. De ballonnen stellen een bepaalde toestand voor, waarin de FSM verkeert. Alleen op een voorwaarde gaat de FSM naar een andere toestand. Deze toestand-

Bumper toestanddiagram



IFA-beurs in Berlijn

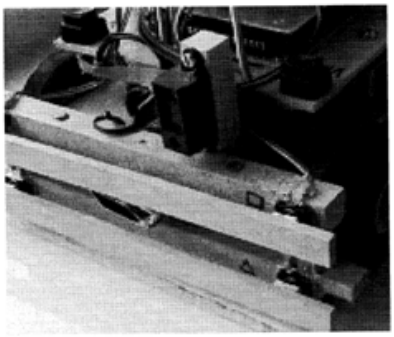
Recent is de IFA-beurs in Berlijn gehouden.

Allerlei vernieuwingen op electronica gebied waren hier te vinden; boodschappen doen vanuit de koelkast, een robot ramenlapper en bijvoorbeeld een robotauto die de kinderen ophaalt.

De sanbot was een van de 'human-like' robots op IFA. Met een vriendelijke glimlach kan hij spelletjes met je spelen en met je praten dankzij sensoren die gebaren registreren.

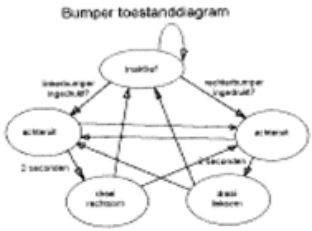
Voor meer info klik op deze link.

Finite State Machines om een Robot gedrag te leren



overgangen worden voorgesteld als een pijl. De voorwaarde voor een toestandovergang staat naast de pijl. Een voorwaarde kan een sluitende schakelaar of een andere sensor of een verstreken tijdsduur zijn.

We nemen aan dat de robot altijd vooruit blijft gaan, maar als er iets gebeurt op de bumper, dan wordt er aktie ondernomen. Deze aktie wordt uitgevoerd door de FSM, het programma dat we BUMPER noemen. BUMPER staat altijd in toestand 0 en leest continu de bumperschakelaars af. Zodra één van de bumpers contact maakt, dan gaat de BUMPER naar toestand 1 wanneer de rechterbumper contact maakt, en naar toestand 3 wanneer de linkerbumper contact maakt, en het command achteruit wordt gegeven. Na 200 milliseconden het command achteruit te hebben gegeven, geeft BUMPER het commando om weg te draaien naar de andere richting, gedurende 400 milliseconden. Nadat 400 milliseconden verstreken zijn komt BUMPER weer in toestand 0 waarin BUMPER geen command meer geeft.



Een voorbeeld maakt dit duidelijk. Het volgende sourcecodefragment is een routine in Forth uit mijn vorige robot Mobo die reageert op een bumperschakelaar en de robot achteruit en daarna links of rechts stuurt.

Circuit of life..

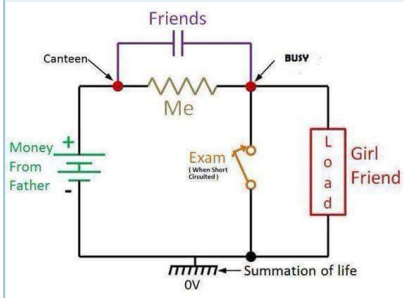
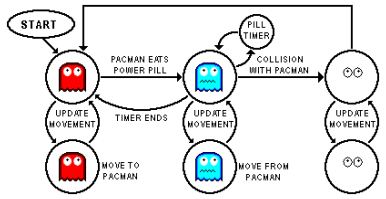


Fig: Circuit of life

Parts list:

- Friends
- Me
- Girl friend
- Money from father



Er is meer informative te vinden over Finite State machines. Zie onderstaande links:

Youtube

Wikepedia

Theorie en praktijk

Finite State Machines om een Robot gedrag te leren

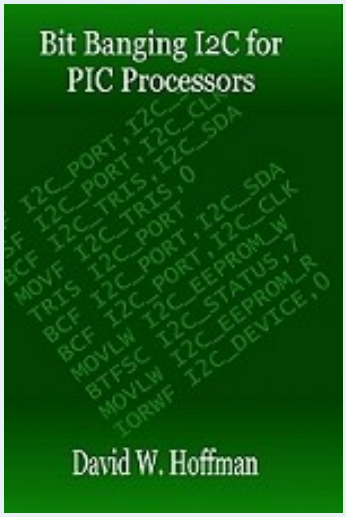

```
: bump  ( - )                          \ behavior "escape" after collision
  bumpright? if                        \ if right bumper hit an obstacle..
     1 to bumpstate                    \ ..set state variable to state 1
    ticks to bumptime                  \ save current time to variable
    green-off                          \ turn signal off
  then
  bumpleft? if                         \ if left bumper hit an obstacle
     3 to bumpstate                    \ go to state 3
    ticks to bumptime
    green-off                          \ signal LED off
  then
  bumpstate case                       \ dit is een finite state machine
    0 of                               \ toestand 0: er gebeurt niets (op bumpers)
       false to bump_akt               \ signaleer bumper niet aktief
       green-on                        \ signal LED on
    endof
    1 of                               \ bumped right, so first go backward
       true to bump_akt                \ signaleer bumper aktief
       red-on
      backward backward merge to bump_cmd    \ going backward
      200 timedout? if                 \ 200 ms elapsed?
          2 to bumpstate               \ go to next state
        ticks to bumptime
        red-off
      then
    endof
    2 of
       true to bump_akt
       geel-on
      backward forward merge to bump_cmd     \ left turn
      400 timedout? if
          0 to bumpstate
        ticks to bumptime
        geel-off
      then
    endof
    3 of                               \ bumped left: backward
      true to bump_akt
      red-on
```

Boeken:

BIT_BANGING I2C

DAVID_WAYNE HOFFMAN

Read the first 22 pages of
Bit Banging I2C!!

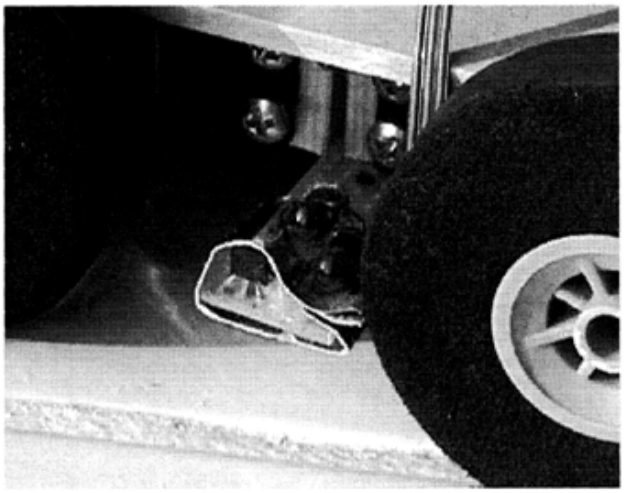


Genoeg van Linux
op de Raspberry Pi ??

Create Internet of Things devices
with Windows 10 IoT
and **Raspberry Pi :**

Finite State Machines om een Robot gedrag te leren

```
  backward backward merge to bump_cmd          \ backward
  200 timedout? if
     4 to bumpstate
     ticks to bumptime
     red-off
   then
 endof
 4 of
    true to bump_akt
    geel-on
    forward backward merge to bump_cmd         \ draai rechtsom
    400 timedout? if
       0 to bumpstate
       ticks to bumptime
       geel-off
     then
  endof
endcase
;
```



Het grote voordeel hiervan is dat het programma nooit in een lus of programmatak blijft hangen. Routine BUMP wordt aangeroepen, test de bumpers en test de voorwaarden waarop een toestandovergang plaatsvindt, en wordt direkt beeindigd. In een andere deel van het pogramma staat de hoofdlus, en daarin wordt BUMP aangeroepen. Ondertussen wordt wel gedurende iedere doorgang door BUMP de bumperschakelaars uitgelezen.

In een goed gestructureerd robotbesturingsprogramma wordt de hoofdlus ongeveer 200 keer per seconde doorlopen, en dus ook routine BUMP. Er is nergens een hardgecodeerde delay in BUMP dus BUMP wordt ook 200 keer per seconde aangeroepen en ook de bumperschakelaars uitgelezen.

Paul Wiegmans.

Aansturen van WS2812b met een Microchip PIC.

Naar aanleiding van het arikel over LEDs in Robobits 73 hierbij een aanvulling, t.w. het aansturen van WS2812b LEDs met behulp van Microchip PICs met de programmeertaal JAL.
Wie onbekend is met JAL of er meer over wil weten verwijs ik graag naar de volgende sites:

JAL users homepage
Jallib at Github

Het is niet de bedoeling om tot in detail in te gaan op JAL en beschikbare bibliotheken, maar om aan te geven hoe eenvoudig een keten van WS2812 LEDs is aan te sturen met een PIC en JAL.

Jallib is een verzameling JAL programma-bibliotheken, waaronder 4 voor het aansturen van een keten van WS2812b LEDs. Waarom 4 van die bibliotheken? Wel, de verschillen zitten in de manier waarop het vereiste WS2812b signaal wordt gegenereerd, een vorm die niet standaard voorhanden is. De signaalvorm kan worden verkregen d.m.v. een USART (Asynchroon of Synchroon protocol), SPI, of 'bit-banging'. Hieronder een uitleg van de overeenkomsten van en verschillen tussen de bibliotheken, enkele argumenten om voor de ene of andere methode te kiezen en tenslotte enkele programma voorbeelden.

Overeenkomsten
Bij gebruik van elk van de bibliotheken is de procedure van aansturen van een serie WS2812s gelijk en vrij simpel. Ongeacht de te kiezen bibliotheek moet een programma het volgende definiëren:

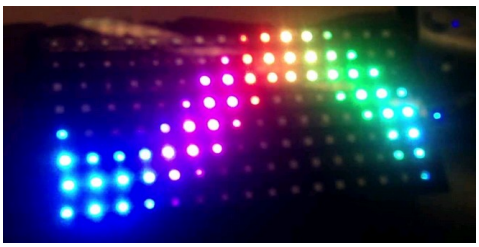
-het aantal WS2812Bs in de keten
-3 arrays met een byte per LED voor groen, rood en blauw (elke byte in een array vertegenwoordigt een LED).
-de output pin die naar de WS2812s DIN loopt

De arrays moeten worden gevuld met de gewenste intensiteit van de betreffende kleur. Vervolgens moet de procedure ws2812b_refresh(.., .., ..) worden aangeroepen om de kleurinstellingen over te zenden naar de WS2812B LEDs.
Zolang deze procedure niet opnieuw wordt aangeroepen blijft het lichtpatroon onveranderd.
Door veranderen van de inhoud van de arrays en pauzes tussen de refresh cycli kunnen allerlei dynamische lichtpatronen worden bereikt. Een kwestie van fantasie en een beetje programmeer-kunst!
Om de benodigde transfersnelheid van 800 Kbps in termen van 'WS2812-bits' te behalen is in alle gevallen een PIC met oscillator frequentie van tenminste 32 MHz vereist. De wat nieuwere PICs kunnen dat allemaal, ook met de interne oscillator.

USART met ASYNCHROON protocol
Bij gebruik van USART met Asynchroon protocol wordt de benodigde vorm van het output signaal voor een '0' of '1' WS2812b-'bit' bereikt door 3 USART data-bits te gebruiken:

'0' is 0b100
'1' is 0b110

De USART wordt ingesteld op 8 data-bits en het start-bit wordt gebruikt als 9-de bit (altijd '1').
Daarmee kunnen dus 3 WS2812b-'bits' per USART databyte worden verstuurd. Om de vereiste snelheid voor de WS2812b te behalen wordt de USART daarbij op 2.4Mbps ingesteld! Met het onvermijdelijke stopbit (altijd '0') is de timing niet geheel volgens de WS2812b specificaties maar blijkt voldoende binnen de tolerantiegrenzen te liggen.



De USART van een PIC gebruikt 'negatieve' polariteit omdat doorgaans een MAX232 o.i.d. wordt gebruikt om een RS232- signaal te verkrijgen (+/- 12V). Maar in dit geval worden de LEDs direct vanuit de PIC op TTL-niveau aangestuurd en moet de output van de USART dus worden ge-inverteerd. Dit kan met een eenvoudig extern circuit, bijv. met een FET BS170, maar onder de nieuwere PICs zijn er vele bij welke de polariteit intern kan worden ge-inverteerd, zoals bijv. de 12f1840, 18f14k22 en 18f25k22 (uitsluitend voor Asynchroon protocol!). De bibliotheek gebruikt deze faciliteit automatisch of geeft een waarschuwing wanneer de PIC dit niet ondersteunt.

Door een verschil in naamgeving van het 'inversie-bit' tussen groepen PICs onderling ondersteunt de huidige ws2812b_async bibliotheek niet de mid-range en enige 18Fs welke wel deze faciliteit hebben.

USART met SYNCHROON protocol

Bij gebruik van USART met Synchroon protocol wordt vrijwel dezelfde methode gebruikt om de benodigde bitstroom te verkrijgen als bij Aynchroon protocol. Echter bij Synchroon protocol worden geen start- en stop-bits gebruikt en wordt de USART ingesteld op 9 databits waarmede toch 3 WS2812b-'bits' per USART databyte kunnen worden verstuurd.
Net als bij asynchroon protocol moet de output van de USART worden geïnverteerd. Dit moet in dit geval wel met een externe schakeling daar de interne USART polariteit-inverter alleen voor Asynchroon protocol werkt.
De noodzaak voor een externe polariteit inverter is een beperking van de huidige bibliotheek. Zonder de bij Asynchroon protocol verplichte start- en stop-bits met voorgeschreven polariteit is de polariteit van de output bij Synchroon protocol geheel onder software-controle!

SPI

De SPI bibliotheek maakt gebruik van de MSSP module met SPI-master-protocol en instelbare baudrate, zoals bijv. bij de 18f25k22.
Omdat SPI een 8-bits protocol is, worden voor een WS2812b-'bit' 4 databits gebruikt, waarmee 2 WS2812b-'bits' per SPI byte kunnen worden verstuurd:

'0' is 0b1000
'1' is 0b1110

Tests wijzen uit dat de daarmee gepaard gaande afwijking van de signaal-timing binnen de tolerantiegrens van de WS2812b ligt. Om de 800 Kbps in termen van 'WS2812-bits' te behalen moet de baudrate worden ingesteld op 3.2MHz.
Het output signaal heeft de benodigde polariteit, er is geen inversie circuit nodig.

NB: De 18f14k22 en enige andere PICs ondersteunen instelbare baudrate alleen voor I2C protocol, niet voor SPI protocol. Voor die PICs is deze library dus niet bruikbaar.

Bit-banging

De 'bit banging' variant is een software implementatie voor aansturen van de WS2812Bs. De vereiste timing wordt hierbij bereikt door de benodigde NOPs te plaatsen tussen het flippen van de output pin. Het output signaal heeft de benodigde polariteit, er is geen inversie circuit nodig.

Welke variant te kiezen?

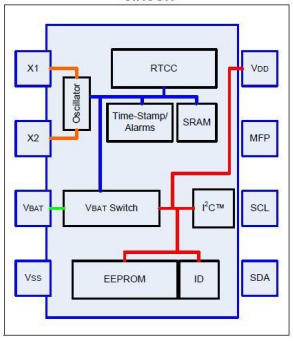
De keuze van de te gebruiken bibliotheek hangt in de eerste plaats af van de beschikbaarheid van modules als USART of MSS voor de toegepaste PIC. Maar er zijn meer argumenten.

De bit-banging methode is met alle PICs toepasbaar: geen afhankelijkheid van USART of MSSP welke wellicht voor andere doeleinden in gebruik zijn. Een nadeel is echter dat de compiler enigszins verschillende code genereert voor de verschillende klassen PICs. Dit verstoort de bit timing enigszins en dat kan een instabiele situatie veroorzaken, hoewel tests uitwijzen dat dit reuze meevalt.

De bibliotheken met behulp van PIC hardware (USART of MSSP) produceren een signaal dat dichter bij de WS2812b specificaties ligt dan met bit-banging: de timing is hardware-gestuurd! Een USART met Synchroon protocol benadert het WS2812b signaal het beste.
Bij gebruik van SPI of USART is er één hardware-bepaalde pin welke het stuursignaal voor de WS2812b bevat. Bij PICs met PPS (Peripheral Pin Select) is dit een aanzienlijk minder probleem, omdat daarmee het output signaal naar een pin naar keuze kan worden gesluisd.

Met SPI en bit-banging kan het stuursignaal worden gegenereerd met de juiste polariteit, met een USART is een polariteit inverter-schakeling nodig. Voor Asynchroon protocol kan bij nieuwere PICs de interne inverter worden toegepast, anders is een extern circuit benodigd.

FIGURE 1-1: TYPICAL OPERATING CIRCUIT



Presentatie van de HCC Forth IG.

Op 3 september werd er een presentatie en demonstratie gehouden door Willem Ouwerkerk van de HCC Forth IG. Onderwerp waren zijn zelf ontworpen en geprogrammeerde tweebenige en zes potige robots, Biped en Hexapod.
Alvorens in te gaan op de bouw, werking en programmering van de robots gaf Willem een toelichting op de keuze van het gebruikte platform.

De Forth IG heeft een paar jaar geleden de keuze gemaakt voor het Launchpad experimenteerbord met MSP430 microcontroller van Texas Instruments. Het bord was destijds erg goedkoop en bij de introductie zelfs gratis te verkrijgen. Voor deze microchip is door de Forth IG een Forth interpreter ontwikkeld. Inmiddels is het bord wat kostbaarder geworden en heeft Willem een eigen bord ontworpen eveneens met een MSP430, de Egel kit. Zowel de Launchpad als de Egel kit zijn te gebruiken in combinatie met de Forth interpreter en vormen het hart van de Biped en Hexapod.

Voor kort verslag van de presentatie, ga naar: ai.hcc.nl/nieuws.html. Voor meer informatie over Forth, de Egel kit en meer, ga naar: forth.hcc.nl. Met dank aan Willem Ouwerkerk voor de presentatie. Met dank aan Gerard Vriens (AI) voor de organisatie en het verslag.

Programma Voorbeelden

Hieronder voorbeelden voor een PIC12F1840. Dit is een van de kleinste PICs (8 pins) die bruikbaar is met alle WS2812 bibliotheken. De 12f1840 heeft een interne oscillator tot 32 Mhz, voldoende geheugen, een snelle USART met polariteit-inverter en een MSSP met instelbare baudrate voor SPI.

Elk van de 4 voorbeelden bestaat uit 3 onderdelen in de aangegeven volgorde:

1. Basiscode specifiek voor de gekozen PIC (gelijk voor alle bibliotheken)
2. Setup voor de gekozen methode/bibliotheek (a,b,c,d)
3. De eigenlijke applicatie (gelijk voor alle bibliotheken)

Dit programma geeft een uiterst simpel lichtpatroon, het is ook vooral bedoeld om de eenvoud van aansturing en de goede werking te demonstreren! Jallib sample directory bevat enkele meer interessante voorbeelden, ook voor andere type PICs.

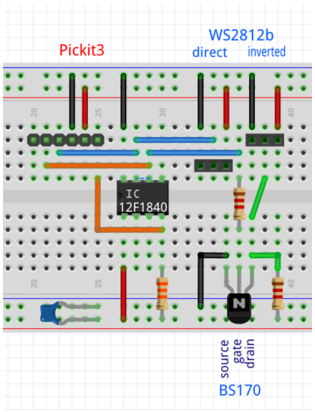
Voor de oplettende lezer:
Wanneer de ws2812b_refresh() procedure niet wordt aangeroepen behoeven de arrays met kleuren-intensiteiten ook niet te worden gedeclareerd!
Voor Asynchroon protocol is een gemodificeerde Jallib bibliotheek gebruikt die wel de interne polariteit inverter van de 12f1840 ondersteunt.

Voorbeeld Circuit

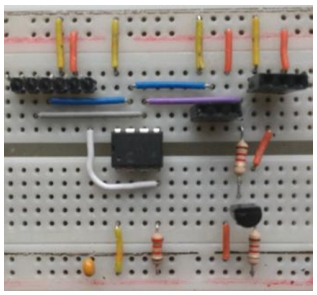
Onderstaande figuur (gemaakt met Fritzing) geeft een mogelijk circuit welk geschikt is voor alle bovenstaande voorbeelden. Voor alle behalve de Synchrone bibliotheek is pin A0 de output. De 12f1840 heeft niet de mogelijkheid om de output van de USART voor Synchroon protocol naar A0 door te sluizen en is daarom op pin A1, zoals ook al uit de code blijkt. Daar dit ook de enige situatie is waarbij een externe polariteit inverter nodig is is dat wel zo overzichtelijk.

Alternatieve methode

Er is tenminste nog één andere methode voor aansturen van WS2812B mogelijk, namelijk middels CLC (Configurable Logic Cell). Vermoedelijk kan hiermee de signaalvorm van de WS2812b exact worden verkregen en het is waarschijnlijk ook de meest efficiente methode. Jallib heeft hiervoor echter geen bibliotheek. Een voorbeeld hiervan (in Assembler code) is te vinden in Microchip Application Note AN1606.



Rob Hamerling



Hiernaast een foto van de werkelijke opbouw (met iets anders gekleurde bedrading!).

1. BASISCODE VOOR DE 12F1840

```
include 12f1840
pragma target CLOCK      32_000_000          -- oscillator frequency
pragma target OSC        INTOSC_NOCLKOUT     -- internal oscillator
pragma target WDT        DISABLED            -- watchdog off
pragma target MCLR       EXTERNAL            -- reset externally
pragma target PLLEN      ENABLED             -- PLL enabled
pragma target LVP        DISABLED            -- no LVP

OSCCON_IRCF   = 0b1110                       -- 8 MHz (+ PLL -> 32 MHz)
OSCCON_SCS    = 0b00                         -- clock determined by fuses

enable_digital_io()                          -- all pins digital
include delay                                -- get delay library

const WS2812B_NUM = 30                       – chain of 30 LEDs
var byte agreen[WS2812B_NUM]                 -- green array
var byte ared[WS2812B_NUM]                   -- red array
var byte ablue[WS2812B_NUM]                  -- blue array
```

2a. Setup voor de ASYNCHRONE bibliotheek

```
APFCON_TXCKSEL = 0                           -- transmit pin configured on pin A0
include ws2812b_async                        -- support library (USART ASYNC variant)
ws2812b_async_init()                         -- initialize
```

2b. Setup voor de SYNCHRONE bibliotheek

```
APFCON_RXDTSEL = 0                           -- data-pin configured on pin A1
include ws2812b_sync                         -- support library (USART SYNC variant)
ws2812b_sync_init()                          -- initialize
```

2c. Setup voor de SPI bibliotheek

```
APFCON_SDOSEL = 0                            -- SDO pin configured on pin A0
pin_SDO_RA0_direction = OUTPUT               -- configure SDO pin for output
include ws2812b_spi                          -- support library (MSSP SPI variant)
ws2812b_spi_init()                           -- initialize
```

2d. Setup voor de bit-banging bibliotheek

```
alias pin_WS2812B is pin_A0                  -- pin A0 pin used
pin_A0_direction = output
include ws2812b_sw                           -- support library (bit-banging variant)
ws2812b_sw_init()                            -- initialize
```

3. De applicatie

```
-- toon achtereenvolgens alle LEDs groen, rood, blauw
forever loop
  ws2812b_all_color(63,0,0)
  delay_100ms(50)
  ws2812b_all_color(0,63,0)
  delay_100ms(50)
  ws2812b_all_color(0,0,63)
  delay_100ms(50)
end loop
```

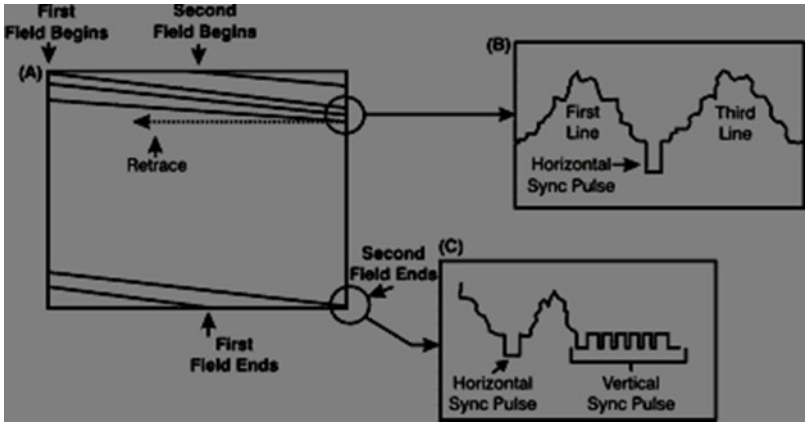
Lijnvolger Sensor op basis van een video camera.

Bij de Roborama wedstrijden zien we verschillende robot's rondrijden met verschillende sensoren die de te volgen lijn herkennen. De meeste van deze sensoren zijn reflectie sensoren. Zij ontvangen in meer of mindere mate de reflectie van een door een Led uitgezonden licht op een een witte of zwarte lijn. Deze mate van reflectie wordt vertaald in een lijn of de rest van het veld.

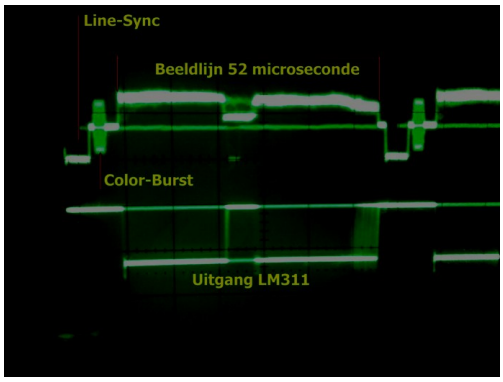
Naast de reflectie sensoren zijn er natuurlijk ook andere manieren om een lijn te detecteren. Een mede Robotica lid, Coen Roos, heeft dit gedaan met behulp van een mini-video camara module. Hierdoor geinspireerd ben ik ook met dit idee aan de slag gegaan en bij deze een verslag van mijn reis om een lijn met behulp van een mini-video camera module te herkennen.

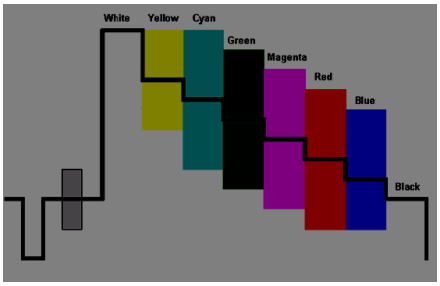
Opbouw video signaal

Er zijn meerdere video-standaarden in de wereld aanwezig. De bekendste zijn PAL en NTSC.
PAL komt voor in de meeste delen van Europa en NTSC komt in de grootste delen van Amerika voor. Het PAL-composite-video signaal is als volgt opgebouwd. Het beeld bestaat uit 625 beeldlijnen. Het is verdeeld in 2 beelden, eerst worden de oneven beeldlijnen (van het eerste veld, Field) getoond en daarna de even lijnen (van het 2e veld, Field). Elk veld(Field) bestaat dus uit 313 beeldlijnen waarvan er aan het begin en het einde een aantal synchronisatie lijnen zijn. Uiteindelijk zijn er per veld, (Field of beeld) 288 actieve, zichtbare lijnen.



Aan het einde van een beeldlijn, die ongeveer 64 microseconde duurt, is een Horizontal/Line-sync puls en aan het einde van een veld (of Field) is een Vertical/Field-sync puls in het signaal aanwezig. Aan de hand van deze 2 Sync-pulsen weten we waar we zitten in de opbouw van een video beeld. Hoe een beeldlijn is opgebouwd zien we op het scoop beeld. Eerst is er een Line-Sync, daarna een colorburst en daarna gedurende 52 microseconde de informatie op het beeld. Als we een zwart/wit beeld zouden bekijken zien we wit als hoog signaal en zwart als laag signaal. Hieronder een schematische weergave van een beeldlijn in een PAL video kleuren signaal. Het mooie van de Roborama wedstrijden is dat de ondergrond helder wit is en de lijn mooi zwart. Dit betekent dat we een goed contrast hebben, maar ook dat we in een videobeeld een duidelijk verschil in nivo hebben.

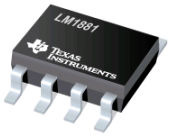




Op het scoop signaal zien we in de beeldlijn een witte ondergrond met in het midden duidelijk zichtbaar een zwarte lijn die de robot moet gaan volgen.

Hardware

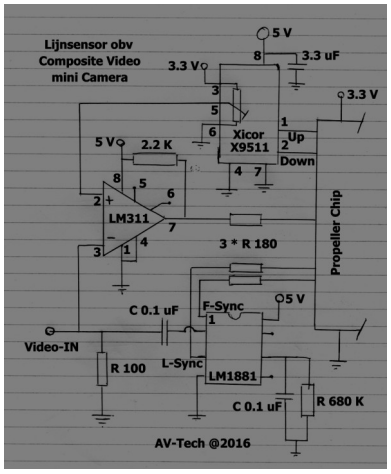
Nu we weten hoe een beeld en beeldlijn opgebouwd is moeten we dit met behulp van wat hardware aan gaan passen zodat we dit in ons programma (software) kunnen gaan gebruiken.



Toevallig (?) is er een IC op de markt wat voor ons heel veel handelingen uit handen kan nemen. Dit is de LM1881 en wordt ook wel Sync-separator genoemd. Aan dit IC geven we een video-signaal en het geeft ons op verschillende uitgangen een (negatieve) puls bij een Line-Sync en bij een Field-Sync. Hoe mooi is dat !!! We weten nu dat we bovenaan een beeld staan EN we weten dat we aan het begin van een beeldlijn staan. Kortom, als we nu de informatie van een bepaalde beeldlijn willen hebben hoeven we alleen nog maar te tellen.

Maar we hebben wel met een analoog signaal te maken en de processor die ik gebruik (de Propellerchip van Parallax) heeft net geen analoge ingangen.
Met behulp van een spannings vergelijker gaan we bepalen of een spanningsnivo boven of onder een bepaald nivo ligt. In feite trekken we een denkbeeldige lijn dwars door ons scoopbeeld heen en zeggen we is het signaal hoger of lager.
We doen dit door middel van een Spannings Comperator LM311. De hoogte van de denkbeeldige lijn kunnen we instellen met een potmeter of in een later stadium met een digitale potmeter zodat we deze met behulp van onze Propellerchip via een programma in kunnen stellen.

Op het scoopbeeld zien we het uitgangssignaal van de LM311 en dat is een mooi digitaal signaal waar we in combinatie met de kennis van de Field- en Line-Sync signalen best wel wat mee kunnen. Omdat de Propeller Chip op 3.3 Volt werkt heb ik 180 Ohm weerstanden gebruikt om de nivo aanpassing te doen van 5 --> 3.3 Volt.

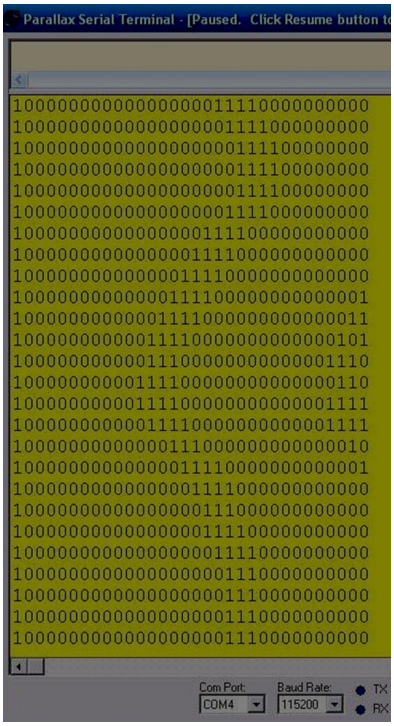


Software

Het programma in de microcontroller, de Propellerchip in dit geval, is best wel eenvoudig.
We wachten op de Field-Sync, wachten vervolgens op de 100-ste Line-Sync (omdat we alleen geïnteresseerd zijn in de 100-ste beeldlijn). En vervolgens doen we in 52 microseconde 32 (digitale) metingen op een ingangspin en schuiven deze informatie in een variabele van het type long.

Waarom de 100-ste beeldlijn ?, waarom niet? Je kunt ook een andere kiezen, het is een keuze. Ook kan je er zelfs voor kiezen om meerdere lijnen te onderzoeken. Waarom opslag van de variabele in een long ? Omdat de Propellerchip hoofdzakelijk met Long's werkt. En, je hebt een "32-bit's sensor" wat meer is dan de meeste andere lijnvolger sensoren.

De opzet van het programma in de Propellerchip is zo dat ik in 1 Cog (processor) in assembly een routine heb staan die het videosignaal binnenhaalt en in een algemeen geheugenadres zet. Vanuit het hoofdprogramma (een andere Cog) wordt, indien gewenst, dit geheugenadres geraadpleegd en de data van de beeldlijn verder gebruikt terwijl op de "achtergrond" in de eerste Cog het videosignaal continu ververst wordt.



Resultaat

Als we de sensor over een lijn halen en het resultaat naar een terminal sturen krijgen we het volgende signaal. Dat ziet er best wel leuk uit. De gebruikte camera module kost rond de 10 Euro op E-bay.

De LM1881 een paar kwartjes en de LM311 ook zoiets. Maar met de laatste had ik een puntje. Het bestelde IC uit Verweggistan werkte NIET (!) Gelukkige had ik in mijn rommeldoos nog een echte LM311 van TI liggen en die werkte wel.

Ijkprocedure

In het programma heb ik een ijkprocedure gemaakt. Als deze uitgevoerd wordt biedt de Xicor-digitale potmeter in 32 stapjes van 0 – 3,3 volt een signaal aan aan de LM311. Dan wordt er in dit traject gekeken wanneer er een lijn herkend wordt en wanneer niet meer. Aan de hand van deze info wordt daarna de Xicor-digitale potmeter op de goede waarde ingesteld. Dit werkt heel leuk.

Conclusie

Als we aan een video camera denken, denken we in eerste instantie aan een compleet beeld wat we (met onze microcontroller) willen gaan bekijken. Dat vergt heel veel rekenkracht wat de meeste processoren die we als hobbyisten gebruiken niet hebben. Maar we kunnen natuurlijk ook "slechts" 1 lijn bekijken en die als bijvoorbeeld lijnsensor gebruiken. Ik heb hiermee een leuk werkende sensor gebouwd, die ook voor microcontrollers geschikt is, mits je snelheid weet te halen met de verwerking van de gegevens. De Propellerchip heeft daar totaal geen probleem mee dus dat was een heel groot voordeel. Op allerlei plaatsen heb ik wachtlussen in moeten bouwen om het goed werkend te krijgen. Het was een leuk project om me hier mee bezig te houden.

Abraham Vreugdenhil.

HCC!Robotica ig

HCC-Robotica is een interessegroep die zich bezig houdt met het ontwikkelen, ontwerpen, programmeren en bouwen van elektronica en mechatronica, toegepast op robots. Deze meer of minder intelligente en autonome robots en machines met verschillende sensoren, actuatoren, processoren en bewegende onderdelen worden onder andere ingezet bij de jaarlijkse georganiseerde Roborama wedstrijden. Wij komen elke eerste zaterdag van de maand bijeen in dorpshuis de Dissel te Hooglanderveen. Kennis delen, kennis vergaren, presentaties en workshops bijwonen zijn terugkerende activiteiten tijdens deze bijeenkomsten.

U bent van harte welkom!

HCC!ROBOTICA: workshop Arduino en I2C

Aankonding workshop Arduino en I2C

Twee jaar na de succesvolle workshop C, Arduino en Robots geven Karel en Joep opnieuw een workshop.

Dit keer is het onderwerp I2C, een handige interface die toegang geeft tot allerlei sensoren en andere randappatatuur.
Met de bekende mix van theorie en prakijk gebruik je verschillende type slaves en maak je zelfs je eigen slave op basis van een Arduino.

De workshop start in januari 2017 en zal tijdens vier achtereenvolgende bijeenkomsten (januari tm. april)
gehouden worden.

Locatie:
De Dissel waar ook de maandelijkse bijeenkomsten plaatvinden.
Hooglanderveen nabij Hoevelaken.

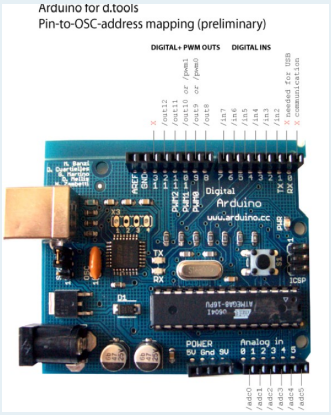
Tijd: 13:00 - 15:00

Discussiegroepen
HCCROBOTICA:
http://groups.google.nl/group/hcc_robotmc

Blogs
http://zotten.wordpress.com/
https://avretro.wordpress.com/
http://www.robotblog.nl/
Blog Huub van Niekerk

Een praktische introductie voor beginners die met een Arduino aan de slag willen.

http://colandino.nl/wp-content/uploads/pdf/Arduino-werkboek.pdf



HCC!Robotica ig

Dagelijks bestuur:
Voorzitter : Bert Berrevoets
Secretaris : Edith van Putten
Penningmeester : Joep Suijs

Het Kernledenbestand ziet er als volgt uit en zal het dagelijks bestuur ondersteunen:
Redactie : Zeno Otten
Website : Pim v. d. Bos
Techniek : Tim Woldring
Roborama : Bert Ruben
Public Relations : Rien van Harmelen
Externe Contacten : Ed Buzzi

Website: http://www.hccrobotica.nl