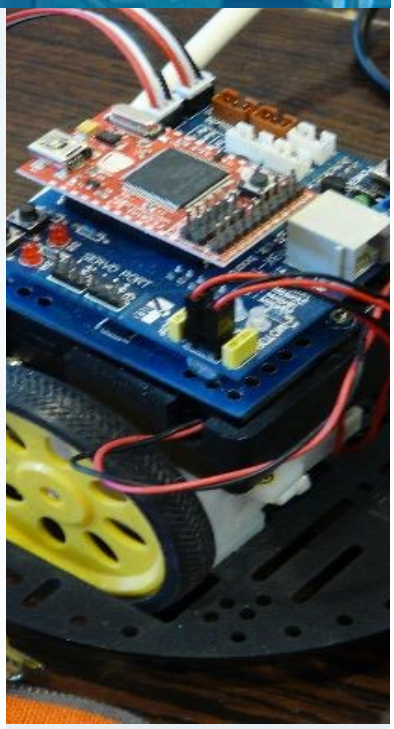
HCC ROBOTICA

Maart 2021
Jaargang 24
Nummer 1

ROBOBITS-91



VAN DE BESTUURSTAFEL

Beste lezer,

We zijn bezig met de voorbereidingen voor de ALV in mei. De vergadering zal virtueel zijn. Mededelingen hierover volgen binnenkort in de nieuwsbrief.

Zoals het er nu uit ziet zal het nog even duren voor we weer life bij elkaar kunnen komen. Wij hopen dat we na september weer kunnen oefenen en in november een RoboRama kunnen houden maar we zijn ook aan het kijken naar de mogelijkheden van een virtuele RoboRama.
Rob vd Ouderaa heeft al wat onderzoek gedaan en stelt voor om alvast te kijken naar https://www.cyberbotics.com/

We zijn samen aan het kijken naar de mogelijkheden en of we wat kunnen doen met het aanleveren van een speelveld en standaard robots. Rob gaat kijken of hij een Nederlandstalige handleiding kan maken. Hierover binnenkort meer.

Heb je zelf een idee over een virtuele RoboRama of andere vormen van het organiseren van een competitie laat het dan vooral weten.

Dan is er bij de '3D' in Delft een groep bezig met Otto robots, https://www.ottodiy.com/. Kunnen we hier als Robotica iets mee? Het kan zijn dat er vragen komen om te helpen met de programmering. En kunnen we bijvoorbeeld een wedstrijd of demonstratie houden tijdens de RoboRama? Graag jullie mening.

De virtuele bijeenkomsten worden vooral bezocht door een vaste groep. Hierbij een oproep, kom eens een keertje mee kijken. Link van de meeting wordt gepubliceerd in onze Google groups https://groups.google.com/g/hcc_robotmc

Veel plezier met je hobby en ik hoop dat de blokkade van het Suezkanaal de aanvoer uit China niet te veel vertraagd :-)

Met vriendelijke groet,
Wim

IN DIT NUMMER

Afwasrobot/wasser dekt de tafel

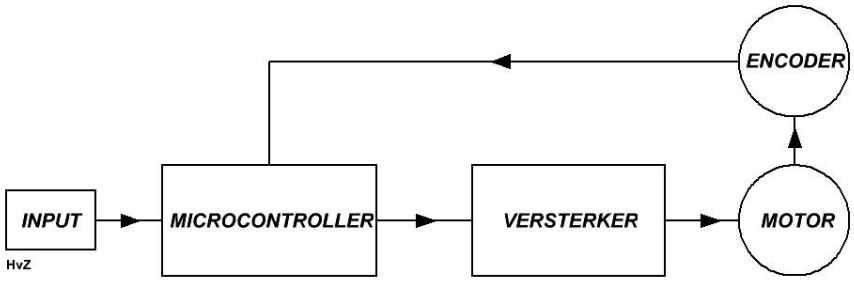
Samsung heeft meerdere nieuwe robots voor thuis gepresenteerd tijdens de techbeurs CES, waaronder een robotbutler die onder meer de vaatwasser kan inruimen!

Werken met servo's (deel 1) .

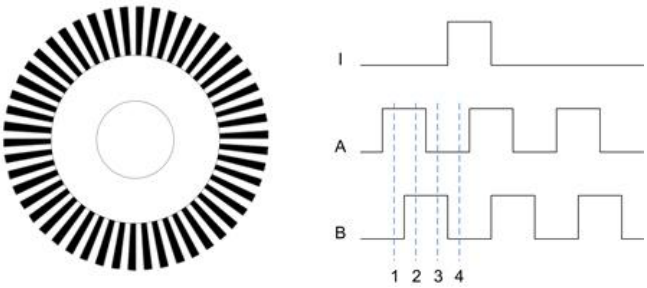
In deze Robobits gaan we aan de slag met servo's en de besturing ervan. In verband met de grootte van dit artikel van Harry van Z. Is het artikel verdeeld over drie Robobits. In Robobits92 en Robobits93 dus deel 2 en 3. Wil je alvast vooruitkijken dan kun je het artikel ook vinden op onze website.

We gaan eens kijken wat een servo nu eigenlijk is. Er zijn verschillende servo systemen, je hebt ze bij hydrauliek, pneumatiek en bij elektrische systemen heb je DC, en AC systemen. Ze komen eigenlijk allemaal op hetzelfde neer; het zijn allemaal teruggekoppelde systemen. Ik ga proberen om in begrijpelijke taal uit te leggen hoe zoiets werkt. Hieronder een blokschema van zo'n systeem.

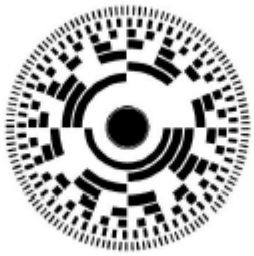
Zoals je in het blokschema kan zien bestaat zo'n systeem uit een controller, een versterker en een encoder. Bij de input wordt een waarde ingegeven waar de motor-as naar toeloopt. Dat signaal gaat van de controller naar de versterker, de versterker is een eindtrap die de motor aanstuurt. Aan de motor zit (in dit geval) een encoder gekoppeld, maar dat kan ook bijvoorbeeld een potmeter zijn, of een resolver, of een tachogenerator of een combinatie van de genoemde componenten. Een encoder geeft een X-aantal pulsen per omwenteling en kan naast het toerental ook de stand van de motor-as en de draairichting van de motor-as aangeven. Encoders heb je in een paar types. Er zijn optische encoders en magnetische encoders. De optische encoders bestaan uit een ronde doorzichtige schijf met streepjes erop en die draait tussen een lichtsluis. Je hebt ook schijfjes van metaal met gaatjes of met sleufjes erin.



Zo'n schijfje ziet er zo uit. Het signaal wat uit een encoder komt bestaat uit blokgolven. Vaak zijn dat drie signalen, maar soms ook twee. In het voorbeeld zijn het er drie, het A en B signaal en het zogenaamde index signaal. Het A en B signaal is 90 graden in fase verschoven ten opzichte van elkaar; op deze manier kan er gekeken worden of de motor links- of rechtsom draait. Het index signaal komt eenmaal per omwenteling voorbij. Je hebt trouwens twee types. Het type hierboven is een incrementele encoder, maar je hebt ook absolute encoders. De absolute encoder behoudt te allen tijde zijn waarde, dus ook na stroomuitval. De incrementele encoder heeft dat niet, bij stroomuitval zijn alle gegevens weg.



Hieronder een paar voorbeelden van zulke encoderschijven :



Zoals je kan zien verschillen ze nogal van patroon, dat heeft ook met het aantal posities te maken die de schijf afgeeft. Ik had het daarnet ook over een magnetische encoder, een magnetische encoder bestaat uit een schijf van magnetisch materiaal, voorzien van bijvoorbeeld tanden en een opnemer die naast of tegenover de schijf is geplaatst. De opnemer zet, als gevolg van het verdraaien van de tanden, optredende variaties in het magnetisch veld om in elektrische pulsen. De puls gever geeft een puls trein af waarbij de frequentie evenredig is aan het toerental van de motor-as, als deze tenminste direct op de motor-as gemonteerd is. Hij kan ook op een reductiekast gemonteerd worden. Het encodersignaal wordt terug gekoppeld naar de controller, de controller regelt dan het toerental en de positie van de motor-as. Op deze manier krijg je dus een gesloten regelkring. Het zijn dan ook complexe en dure regelingen.

Er zijn ook servomotoren waar de encoder, versterker en controller ingebouwd zijn in de servomotor.

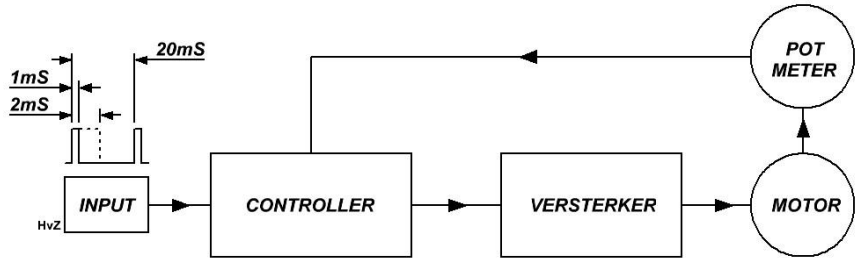
Hiernaast zie je een voorbeeld van zo'n servo systeem. Je ziet dat alles in één unit is ingebouwd, zo'n systeem heeft alleen een voeding nodig en een signaal om de positie door te geven. De positie doorgeven wordt vaak door een bus- systeem gedaan. Dit kan gebeuren door bijvoorbeeld Profibus, maar er zijn ook andere veldbus-systemen.

Je hebt natuurlijk ook servo systemen met een losse servo driver en motor. In de servo driver zit de controller en de eindversterker die de motor bestuurt. Zo'n systeem wordt vaak door een PLC of een PC aangestuurd. De systemen worden vaak in de machinebouw gebruikt voor het snel positioneren van assen. Voor bijvoorbeeld CNC machines zijn het uitermate geschikte systemen.

Hiernaast zie je zo'n losse servo driver.

Dit is wel een drie fase AC servo driver, maar deze heb je ook voor DC. Vaak hebben dit soort drivers ook nog wat I/O aan boord voor diverse functies, de I/O poorten kunnen vaak naar eigen inzicht geprogrammeerd worden. Vaak zit er tegenwoordig ook een Ethernet of Profibus aansluiting op om ze op afstand uit te lezen of te bedienen.

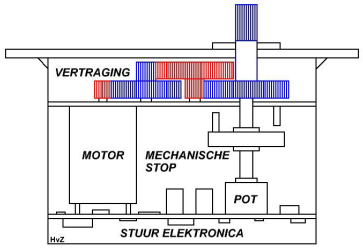
Dit zijn dure systemen en daar ga je niet zo snel mee experimenteren, maar gelukkig zijn er ook goedkope systemen. Die worden vaak in de modelbouw gebruikt en die noemen we dan ook modelbouw servo's. Dat zijn wel vaak systemen die maar een X-aantal graden versteld kunnen worden, maar ze vallen toch onder de noemer servo. De werking is dan ook een beetje hetzelfde, het zijn ook gesloten regelsystemen, net als de grotere broers. Zie voorbeeld hieronder.



Zoals je kan zien verschilt het blokschema niet zoveel van zijn grote broer. Het inputsignaal is wel anders, hier is het een PWM signaal met een vaste frequentie van 50Hz en een variabele puls breedte die varieert tussen de 1 en 2 milliseconden. Met een signaal van 1 milliseconden staat de servo bijvoorbeeld helemaal links en met een signaal van 2 milliseconden staat de servo helemaal rechts.

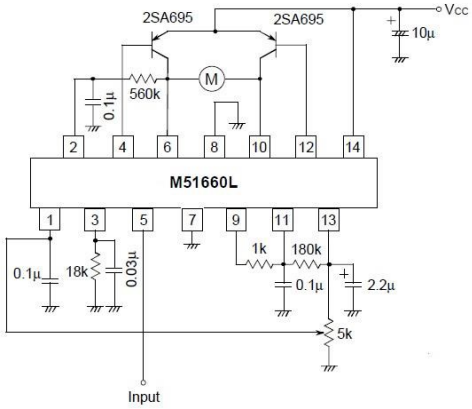
Als het signaal 1.5 milliseconden bedraagt staat de servo precies in de middenstand. De as van een modelbouw servo kan +/- zo'n 180 graden (verschilt per servo) verdraaien. Het inputsignaal is meestal afkomstig van een zender en ontvanger. Het zendersignaal wordt in de ontvanger omgezet naar een variabele puls breedte. De elektronica in de servo (tegenwoordig bij digitale servo's is dat een microcontroller) stuurt de eindversterker aan die op zijn beurt de motor bedient. Het terug- gekoppelde potmetersignaal zorgt ervoor dat de servo naar zijn gewenste positie loopt. Dus wat de encoder doet bij de industriële systemen, doet de potmeter bij de modelbouw servo's. Niets anders dus dan de gewenste positie vergelijken met de positie van de motor-as.

Hieronder zie je hoe zo'n modelbouw servo is opgebouwd.

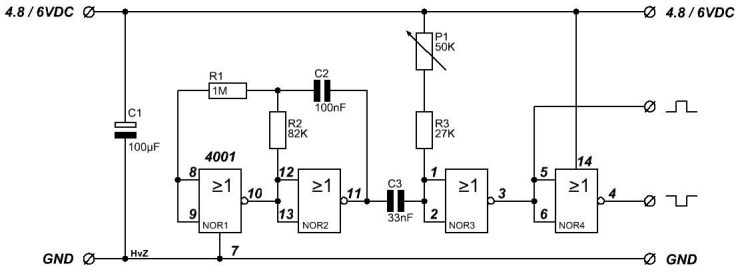


De servo bestaat uit een motor, een vertraging, de potmeter voor de positie bepaling, een mechanische stop (deze is belangrijk, want die beschermt de potmeter tegen te ver doordraaien) en niet te vergeten de besturingselektronica. De modelbouw servo heeft drie aansluitdraden, een plus en een min en de draad waar het PWM signaal op binnen komt.

Hiernaast één van de mogelijke schema's van de besturingselektronica. Op het schema kun je zien hoe zo'n besturing is opgebouwd. Het hart van de besturing is de M51660L, dit IC regelt alles. Het enige wat nodig is, zijn nog wat randcomponenten voor de juiste werking. Op pen 1 komt het potmetersignaal binnen, daar wordt de servo positie op ingelezen. Pen 2 en 3 zijn voor de timing. Pen 4 en 12 sturen de externe PNP transistoren aan. Pen 5 is het ingangssignaal dat bijvoorbeeld van de ontvanger komt. Op pen 6 en pen 10 zitten de interne transistoren, die samen met de externe transistoren de motor aansturen. Pin 9 is de error puls uitgang. Pen 11 de stretcher ingang.

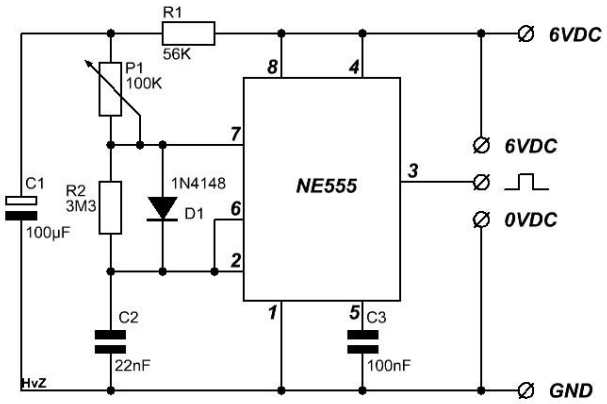


Pen 13 verzorgt de spanning voor de potmeter. Pen 7 en pen 8 zijn de GND aansluitingen en pen 14 is de voeding voor het IC. Zo heb je een idee hoe een modelbouw servo is opgebouwd. Aan de besturing in de servo kan je weinig veranderen, maar het besturen van een modelbouw servo kan op verschillende manieren. We hadden het al over de ontvanger die voor dit signaal zorgt, maar dat kan ook anders. Je kan bijvoorbeeld met een 4001 (een nor poort) een prima puls generator maken waar je dan de servo mee kan bedienen. Zie schema hieronder.



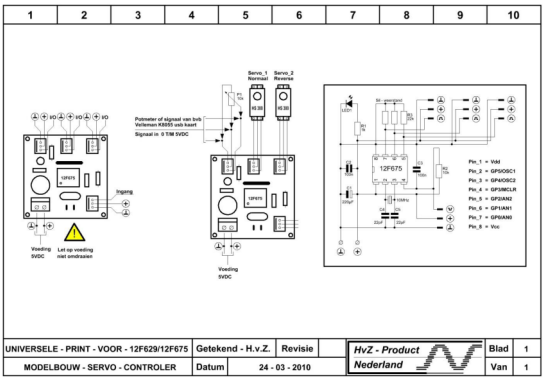
De schakeling kan gevoed worden met een spanning tussen de 4.8 en 6VDC. Er komen twee pulsen af, de ene is geïnverteerd ten opzichte van de ander. Dit kan makkelijk zijn omdat er servo's zijn die een geïnverteerd signaal nodig hebben. Zo kan je op een eenvoudige manier dus je servo bedienen.

En zo kunnen we met een NE555 ook een puls generator maken om een modelbouw servo te bedienen. Zie schema hiernaast.



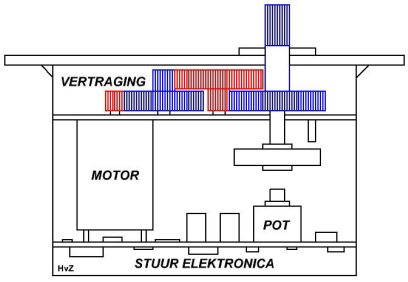
Het is ook mogelijk om het met een microcontroller te doen, met bijvoorbeeld een 12F629 of een 12F675 kan dat heel simpel gerealiseerd worden. Hieronder het schema en het programma wat in de controller staat voor het bedienen van de servo.

```
Device 12F675                   ; Processor type

Xtal 10                         ; Kristal 10 Mhz

Config WDT_OFF,_                ; WatchDog Timer uit
       PWRTE_ON,_               ; Power-up Timer Enable aan
       MCLRE_OFF,_              ; Externe Master Reset Enable uit
       HS_OSC                   ; X-tal groter dan 4MHz

All_Digital true                ; Alle poorten digitaal

Symbol servo_1 = GPIO.1         ; Servo uitgang_1
Symbol servo_2 = GPIO.2         ; Servo uitgang_2

Declare Adin_Res    = 8         ; Resolutie 8 bits
Declare Adin_Tad    = frc       ; Set rc osc
Declare Adin_Stime = 50         ; Sample tijd 5

Dim potmeter As Byte            ; Variabele waarde potmeter
Dim positie As Word             ; Variabele waarde positie
Clear                           ; wis geheugen
```

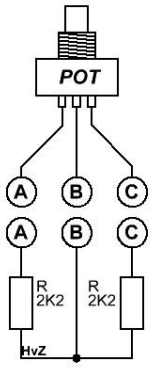



```
        ;543210                 ; Hulpregel
GPIO    = %000000               ; Maak poort laag
TRISIO = %001001                ; In en uitgangen

        ;76543210               ; Hulpregel adcon register
ADCON0 = %00000001              ; A/D control register

run:                            ; Run programma
    potmeter = ADIn 0           ; Potmeter op analoog_0
    positie = 850 + (potmeter*7) ; Berekening positie servo
    Servo servo_1,positie       ; Uitsturing positie servo_1
    Servo servo_2,positie       ; Uitsturing positie servo_2
    DelayMS 20                  ; Pauze 20 Ms
GoTo run                        ; Ga naar run programma

End                             ; Einde programma
```

*Zoals je kan zien stelt het programma niet veel voor, op de analoge ingang (ADIN_0) komt de potmeter die de servo gaat bedienen. De minimale positie van de servo is 850, dat is de waarde als de potmeter op nul staat. De maximale waarde is 850 + (255 X 7) = 850 + 1785 = 2635. Die waardes vallen buiten de 1 en 2 milliseconden, maar de ene servo pikt dat wel en de andere servo niet. Dat ligt dus aan de constructie van de servo. Ik heb een HITEC HS300 gebruikt en die haalt het wel. Dus als je gaat experimenteren kan je de waarde achter de positie aanpassen, plus potmeter * 7.*



We hebben nu gezien hoe je een servo heen en weer kan laten lopen, daar is hij ook voor bedoeld. Maar met een kleine aanpassing kan je er ook een mooie aandrijving mee maken die gewoon rondjes kan draaien, bijvoorbeeld om een robot mee aan te drijven. Wat we dan doen is de mechanische stop eruit halen en de potmeter loshalen van de servo-as. De potmeter zet je nu in de middenstand en die lijm je bijvoorbeeld vast. Je kan de potmeter ook vervangen door twee weerstanden van gelijke waarde. In het schema van de standaard besturingselektronica staat dat de weerstand van de potmeter 5K is. Dus je zou twee weerstanden van bijvoorbeeld 2K2 kunnen nemen, maar je kan ook de potmeter laten zitten. Hieronder zie je het voorbeeld, kijk wat er veranderd is ten opzichte van de standaard servo.



Als je de potmeter verwijdert en je zet de twee weerstanden zo neer als in het voorbeeld, dan krijg je dezelfde waarde als met de potmeter in de middenstand.

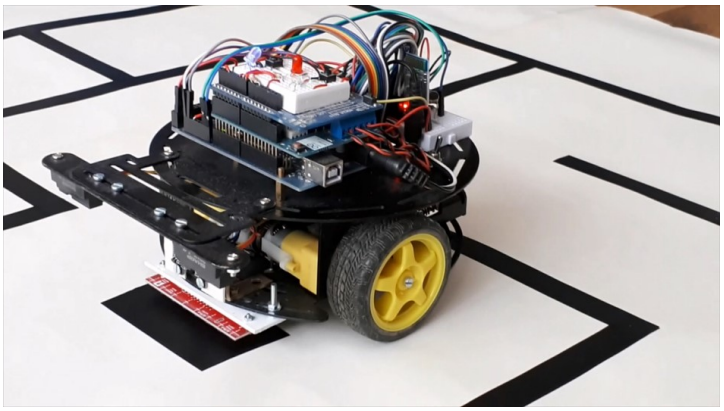
Ik heb nu 2K2 weerstanden genomen, maar dat kan ook een andere waarde zijn natuurlijk. Als je twijfelt kan je de waarde van de potmeter even opmeten, dan weet je het helemaal zeker. Meet de waarde tussen de twee buitenste pootjes en deel de waarde door twee. Als daar een waarde uitkomt waar geen weerstand voor is, dan kies je de waarde die er zo dicht mogelijk bij in de buurt komt. Voor de weerstanden kan je gewoon 0.6Watt types nemen. Ik zou wel 1% weerstanden nemen, dan heb je de kleinste afwijking.

Als je de weerstanden gemonteerd hebt kan je de servo nu gebruiken als aandrijving om bijvoorbeeld een robot te laten rijden. Je hebt er dan wel twee nodig natuurlijk.

(eventueel vooruit lezen via deze link of wachten tot volgende Robobits.)

Een doolhof oplossen met een robot.

In dit artikel wil ik mijn ervaringen met jullie delen met het bouwen van een doolhof robot. In de Corona periode, die mij net als iedereen in een huismus had veranderd, was ik op zoek naar een nieuw type uitdaging. Op de bijeenkomsten in de Dissel heb ik met bewondering en enige jaloezie gekeken wat enkele IG-leden konden met hun robots in een doolhof. Dat zag er erg intelligent uit, en het leek me wel wat om dat ook te proberen.

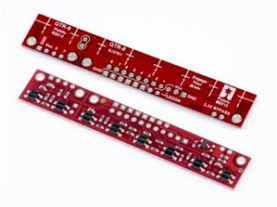


Weten jullie trouwens dat de Robotica IG een officieel Doolhof reglement kent (https://robotica.hcc.nl/downloads/file/110-regelementen.html)? Dat reglement beschrijft onder andere dat het labyrint bestaat uit een reeks van zwarte lijnen op een wit veld. Alle lijnen liggen op een (denkbeeldig) raster van 150 mm.

Maar waar te beginnen?

Als eerste ging ik maar eens na wat voor technieken ik hierbij zou kunnen inzetten die ik al kende:

- *Een rijdende robot bouwen, en omgaan met motor-encodersignalen*
- *Via bluetooth sensordata en interne informatie op afstand uitlezen met een HC-05 bluetooth module*
- *Uitlezen van een lijnvolgsensor*



Ik deed aan de Lockdown Challenges mee met een Turtle robotplatform van DFRobot, voorzien van 6v motoren met redelijk bruikbare 6V motoren met encoders (… pulsen per omwenteling). Deze is wel iets te groot om mee te dingen in een officiële competitie, maar dat mag de pret niet drukken. Het voordeel van deze robot, die ik TurboTurtle heb gedoopt, was wel dat hij al beschikte over een Pololu QTR-8a 8-voudige array van lijnvolgsensoren.

Alles wat mij te doen stond was het maken van een doolhofprogramma.

Als programmeeromgeving bleef ik bij de voor mij vertrouwde Arduino IDE. Sommigen missen hierin de geavanceerdere functies die andere ontwikkelomgevingen je kunnen bieden, maar de Arduino omgeving biedt veel libraries, ik ken er de weg in en ik heb er veel code in die ik kan hergebruiken.

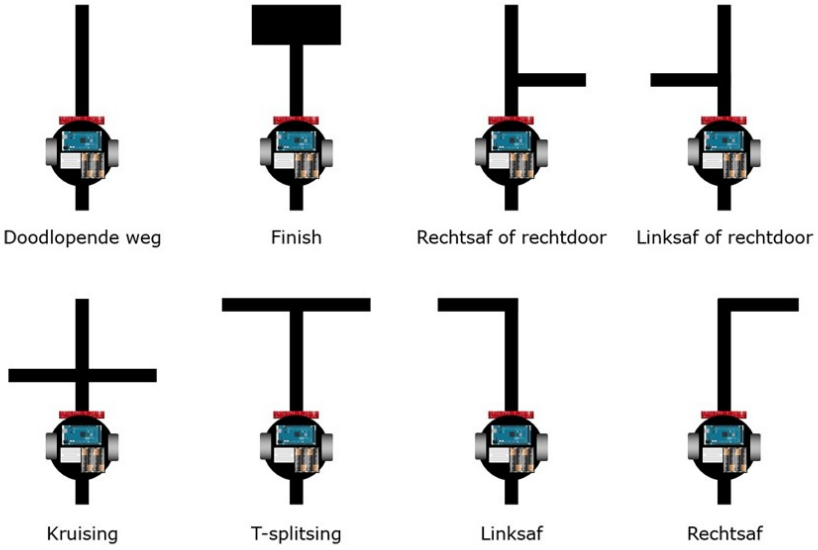
Vervolgens ging ik mij verdiepen in het algoritme om een doolhof op te lossen. En daar kwam de HCC Robotica-groep natuurlijk heel goed van pas. Al snel wezen vriendelijke helpers mij op de linkerhandregel: als je je hand steeds tegen de linkerwand houdt, zul je uiteindelijk het hele doolhof bezoeken.
Google op 'linkerhandregel doolhof' of 'left hand rule maze' levert een heleboel leuke en nuttige voorbeelden op. De linkerhandregel heeft wel een beperking: De oplossing werkt alleen voor 'eenvoudig verbonden' doolhoven. Dat is de benaming voor een doolhof waarbij alle muren met elkaar zijn verbonden.
In een doolhof dat bestaat uit zwarte lijnen komt dit erop neer dat er geen lussen in het doolhof voorkomen. Maar dat geldt in de praktijk voor een heleboel doolhoven.

Het oplossen van een doolhof bestaat in de basis uit 2 stappen. De eerste stap is de verkenning van het doolhof om de locatie van de finish te vinden. Hierbij volgt TurboTurtle de linkerhandregel. In de tweede stap wordt TurboTurtle op de oorspronkelijke beginpositie gezet, waarna hij zelfstandig de kortste weg door het doolhof naar de finish kiest.

.

HET PROGRAMMA VOOR FASE 1: VERKENNING VAN HET DOOLHOF

Om de robot een lijnendoolhof te laten rijden , moet je natuurlijk ook met je robot een lijn kunnen volgen. Daarvoor besloot ik om een PID-regeling toe te passen. Op zich al weer een kleine challenge, want die had ik nog nooit echt gebruikt (behalve als kant-en-klaar demonstratie program-maatje). Dat had makkelijker gekund, want een rechte lijn volgen zou best eenvoudiger kunnen dan met PID. Hoe die programmacode eruit ziet is misschien iets voor een ander artikel. De volgende uitdaging is het afhandelen van de punten waarop de weg verandert. Als je daar over nadenkt zul je ontdekken dat er acht mogelijke vormen voor een knooppunt zijn:



Hoe bepaalt TurboTurtle met welk knooppunt hij te maken heeft?:

De eerste detectiestap die de robot doet is of de weg is doodgelopen. Het is logisch om als eerste op deze situatie te testen omdat hij meteen eenduidig te herkennen is. De lijnvolgsensor registreert namelijk ineens 0 hits. En de actie als je een doodlopende weg detecteert is ook duidelijk: Stop en keer om.

Voor de overige situaties zal de robot wat vervolgonderzoek moeten doen. Want zie je bijvoorbeeld een aftakking naar rechts, dan kan je misschien ook nog rechtdoor op dat punt. En zie je zowel een aftakking naar links als naar rechts, dan kan dat zowel een kruising als een T-splitsing zijn.

Als de testroutine het type knooppunt heeft bepaald dan reageert de robot volgens het volgende schema, wat in feite de uitwerking van de linkerhandregel is:

Knooppunt	Code voor de situatie	Actie
Doodlopende weg	*B*	*Keer om*
Finish	*EOM*	*Stop en initieer Fase 2*
Rechtsaf of rechtdoor	*RS*	*Rij rechtdoor*
Linksaf of rechtdoor	*LS*	*Sla linksaf*
Kruising	*LRS*	*Sla linksaf*
T-splitsing	*T*	*Sla linksaf*
Linksaf	*L*	*Sla linksaf*
Rechtsaf	*R*	*Sla rechtsaf*

De betekenis van de lettercodes:
L = Links
R = Rechts
B = Back (rechtsomkeert)
S = Straight (rechtdoor)
T = T-splising
EOM = End Of Maze (einde doolhof)

Hoe vertaalt zich dit dan in programmacode?
Laat ik beginnen met te zeggen dat de programmacode van TurboTurtle werkt volgens het principe van een State machine. De meeste tijd verkeert de robot in de State 10: volg de lijn. Tijden het volgen van de lijn test de robot voortdurend of er zich een verandering voordoet:

1. Loopt het pad dood? Ga naar de State 180 voor rechtsomkeert maken
2. Doet er zich een aftakking voor? Bepaal dan het type kruising
2. Heb je het einde van het doolhof bereikt? Stop en stap over naar programmafase 2 (State 300)
4. Als zich een aftakking voordoet, kun je dan rechtdoor? (situatie RS) Behoud de huidige State, en blijf gewoon rechtdoor rijden.

Eindconditie State 10:

```
if (hits == 0) {                                    // test op doodlopende weg
  Motors(0,0);
  Serial3.println(F("  State 10: Doodlopende weg, keer om!"));
  State = 180;
} else if(sensorValues[1] > QTRZWART || sensorValues[NUM_SENSORS-2] > QTRZWART) {
                                                    // test op afslag, splitsing of kruising
  Richting = RichtingStore.read();
  TestJ_uitslag = testJunction(sensorValues);   // bepaal type kruising
  if (TestJ_uitslag == JNC_EOM) {
    State = 300;                                  // Einde doolhof bereikt
  } else if (TestJ_uitslag != JNC_RS) {
    State = 11;
  }
}
```

De resterende testen, die in de functie testJunction() zijn geïmplementeerd, verlopen dan als volgt:

Als het knooppunt dus niet een doodlopende weg is , gaat de testfunctie lopen die het type kruising bepaalt en retourneert. Die test verloopt in de volgende stappen:

1.kijk met de lijnsensor of er links en rechts afslagen zijn,
2. rijd een stukje rechtuit,
3. kijk met de lijnsensor of het pad rechtdoor mogelijk is.
4. Bepaal d.m.v. de verzamelde informatie het type kruising en retourneer die waarde.

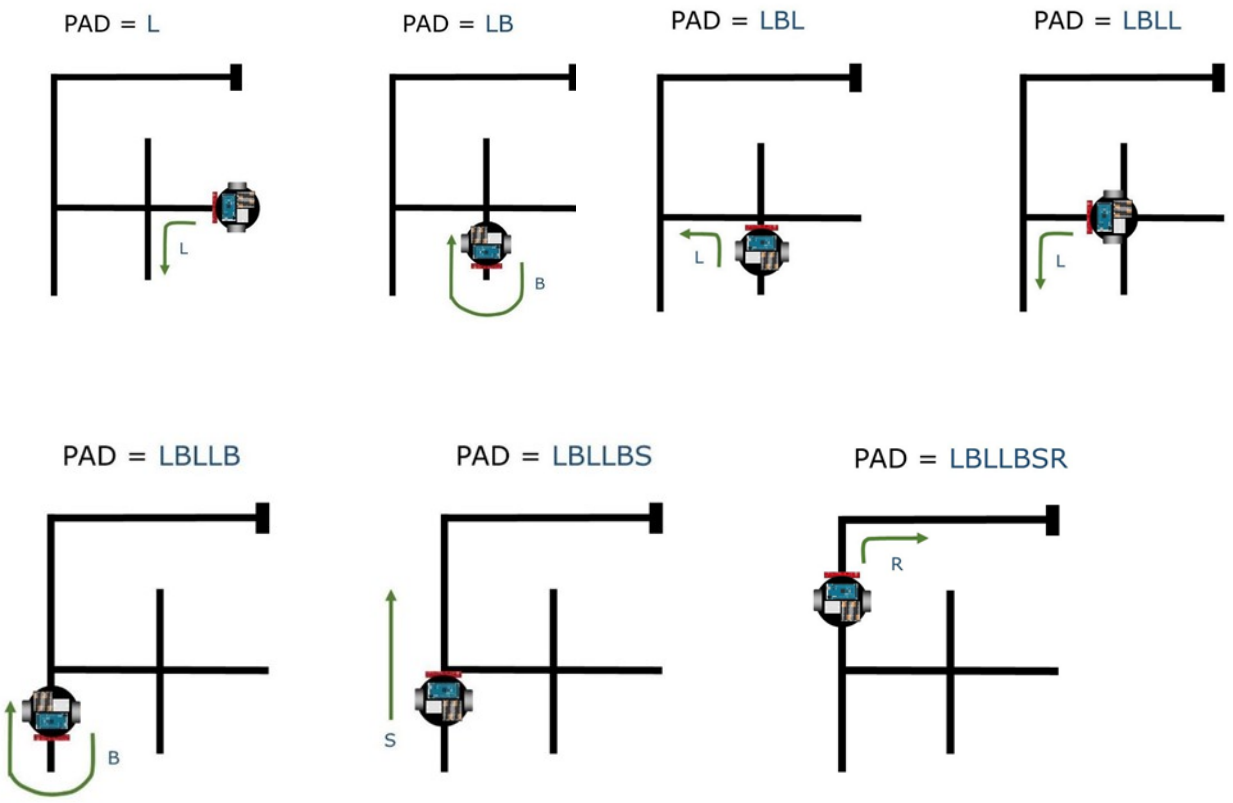
Is het niet een van de reeds genoemde gevallen, dan weet je dat je in ieder geval een afslag links of rechtsaf zult moeten nemen. Daarom wordt eerst een 'tussen'state 11 uitgevoerd om een klein stukje rechtdoor te rijden om goed uit te komen bij het maken van de draai.

De State Machine kiest vervolgens aan de hand van de geretourneerde waarde met behulp van een switch-case statement de volgende State.
Eindconditie State 11:

```
switch (TestJ_uitslag) {
  case JNC_LS  :
  case JNC_LRS :
  case JNC_T   :
  case JNC_L   :
    {
      State = 90;                           // 90 graden linksaf
      break;
    }
  case JNC_R   :
    {
      State = 100;                          // 90 graden rechtsaf
      break;
```

VOORBEELD WERKING LINKERHAND ALGORITME

Als je dit algoritme toepast op een eenvoudig voorbeeld-doolhof levert dat het volgende resultaat op:



OK, nu de robot dit voor elkaar heeft hebben we een pad. Het pad wordt in het geheugen opgeslagen door op iedere kruising de actie op te slaan die de robot neemt. Daarvoor maken we de volgende variabele (array) aan:

```
char path[100] = " ";
```

We declareren ook 2 index variabelen voor gebruik bij de array:

```
unsigned char pathLength = 0; // de lengte van het pad
int pathIndex = 0;      // index om een specifiek element van de array te kunnen benaderen.
```

Als we bovenstaand voorbeeld uitvoeren eindigen we met:

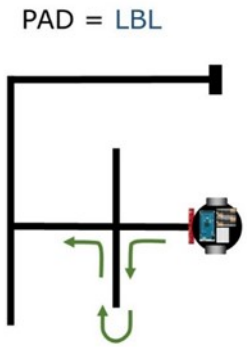
path = [LBLLBSR]
en pathLengh = 7

FASE 2: HET OPTIMALISEREN VAN HET PAD

De volgende uitdaging is het algoritme om dit pad te vereenvoudigen tot een korter pad.

Laten we kijken naar de eerste foute afslag die we namen. Een "B" (voor "Back") betekent dat de robot rechtsomkeert maakte - hij had dus een doodlopende afslag genomen. We kunnen het pad dus optimaliseren door op een of andere wijze die "B" door iets anders te vervangen.

Neem de eerste 3 acties in het pad van het voorbeeld. Die zijn "LBL". In totaal ziet dat er als volgt uit:



In plaats van linksaf te slaan, rechtsomkeert te maken en weer linksaf te slaan, had de robot dus beter meteen rechtdoor kunnen gaan. Dus kunnen we stellen dat LBL => S. Deze vervanging is wat de robot gebruikt om het pad te verkorten. De hele lijst van substituties ziet er als volgt uit:

LBL → S

LBR → B

LBS → R

RBL → B

SBL → R

SBS → B

LBL → S

Als je zin hebt, kun je ze op papier uittekenen om te zien hoe ze werken.

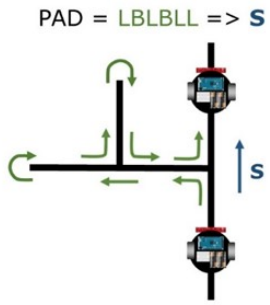
Voor het codevoorbeeld, kun je de volgende webpagina raadplegen:
https://create.arduino.cc/projecthub/mjrobot/maze-solver-robot-using-artificial-intelligence-4318cf

Hoe vertaalt zich dit dan weer in Arduino C-code? Daarvoor bevat het programma een functie die verantwoordelijk is voor de conversies. Hoewel het robotprogramma eromheen eigen ontwikkeling is, kon ik deze functie letterlijk overnemen van het codevoorbeeld. Daarin zul je niet direct de letteromzettingen terug zien komen. De bedenker heeft namelijk een slimme manier verzonnen om de conversies te doen. Hij vertaalde de draaien links, rechts en rechtsomkeert naar een hoek in graden van 90, 270 resp. 180. Als je die optelt modulo 360 rolt de gewenste draaihoek er uit, die zich weer terugvertaalt naar L, R, of B.

```
void simplifyPath()
{
  // only simplify the path if the second-to
-last turn was a 'B'
  if(pathLength < 3 || path[pathLength-2] !=
'B')
    return;

  int totalAngle = 0;
  int i;
  for(i=1;i<=3;i++)
  {
    switch(path[pathLength-i])
    {
    case 'R':
      totalAngle += 90;
      break;
    case 'L':
      totalAngle += 270;
      break;
    case 'B':
      totalAngle += 180;
      break;
    }
  }

  // Get the angle as a number between 0 and
360 degrees.
  totalAngle = totalAngle % 360;

  // Replace all of those turns with a sin-
gle one.
  switch(totalAngle)
  {
  case 0:
    path[pathLength - 3] = 'S';
    break;
  case 90:
    path[pathLength - 3] = 'R';
    break;
  case 180:
    path[pathLength - 3] = 'B';
    break;
  case 270:
    path[pathLength - 3] = 'L';
    break;
  }

  // The path is now two steps shorter.
  pathLength -= 2;
}
```

Het is natuurlijk prima mogelijk om een eigen versie van simplifyPath() implementeren die hetzelfde doet op basis van het herkennen van de letterreeksen. Ik was daar echter te lui voor ;-).

Wanneer wordt de functie simplifyPath() nu in het programma aangeroepen? Als de robot het doolhof verkent, kijkt hij bij elk knooppunt of hij het pad kan optimaliseren, door te testen of de laatste drie gevonden knooppunten voldoen aan één van deze substitutieregels. De werking van het algoritme kun je volgen aan de hand van het volgende stukje doolhof:



Het gewenste eindresultaat van de optimalisatie is natuurlijk dat de robot de gehele aftakking negeert en bij het eerste knooppunt rechtdoor gaat. Het pad evolueert als volgt bij toepassen van het algoritme:

L
LS
LSB
LSBL → simplifyPath() → LR
LRB
LRBL → simplifyPath() → LB
LBL → simplifyPath() → S

En voilà, het resultaat is bereikt. Als dit eenmaal geprogrammeerd is kan de robot ook heel ingewikkeld ogende doolhoven aan. Als er maar geen lussen in voorkomen.
Het uitvoeren van Fase 2 is dan in feite slechts het volgen van het geoptimaliseerde pad.

Tot slot van dit artikel een link naar een Youtube-filmpje van mijn doolhofrobot in actie:

https://youtu.be/qSSniuzVxpg

Ik hoop dat jullie het interessant om vonden dit te lezen. Ik moet er wel bij zeggen dat het er op papier heel logisch uit ziet, maar in de praktijk bleek er nog aardig wat 'tuning' bij te komen kijken om de robot succesvol bij de finish te laten aankomen. Meer over die praktijkervaringen in een volgend artikel.

Februari 2021, Ewoud Hüttner

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!

Agenda

Samen met Expo Houten plant HCC dit jaar twee keer een Kennisdag. Zet 26 juni en 9 oktober alvast in je agenda. De Kennisdag op 1 mei a.s. gaat helaas niet door.

Onze maandelijkse HCC!Robotica bijeenkomsten in de Dissel in Hooglanderveen gaan voorlopig ook niet door. We zullen moeten afwachten hoe alles zich ontwikkelt..

Discussiegroepen
HCCROBOTICA:

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

Blogs
http://zotten.wordpress.com/

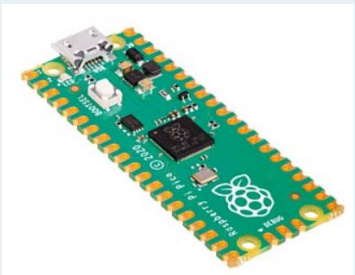
https://avretro.wordpress.com/
http://www.robotblog.nl/

Facebook

HCC!Robotica ook op Facebook.

Gewoon om te laten weten, dat wij ook op Facebook actief zijn.

Raspberry Pi Pico



De Raspberry Pico is een klein, snel en veelzijdig bord gebouwd op basis van een RP2040.
Ontworpen door Raspberry Pi, is de RP2040 voorzien van een dual-core Arm Cortex-M0 processor met 264 KB interne RAM en ondersteuning voor maximaal 16 MB off-chip Flash.
Ondersteuning van I2C, SPI en **micro-python**. Programmeren van een microcontroller is nog nooit zo eenvoudig geweest!

HCC!Robotica ig

Dagelijks bestuur:
Voorzitter : Wim de Boer
Secretaris : Edith van Putten
Penningmeester : Ed Buzzi

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

Website: http://www.hccrobotica.nl