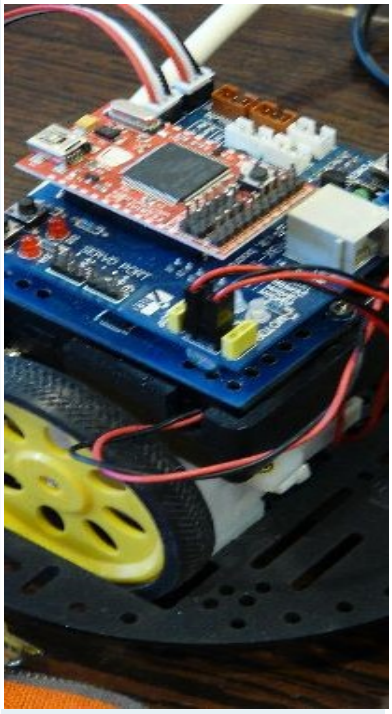
HCC ROBOTICA

September 2015
Jaargang 18
Nummer 3



ROBOBITS-70

VAN DE BESTUURSTAFEL

Beste lezer,

Op het moment van schrijven is de zomer al weer bijna helemaal voorbij. Dit betekent dat de wedstrijden er weer aan komen.

Dus is het tijd om de robots nog even goed te testen en een poetsbeurt te geven. En dan als het zover is hopen dat de robot doet waarvoor je hem geprogrammeerd hebt. En vergeet vooral niet om de batterijen op te laden!

Ik hoop dit jaar weer veel deelnemers te zien. En ik hoop natuurlijk veel publiek.

Maar wat ik ook wil doen is om met z'n alle te brainstormen om misschien een nieuw onderdeel voor de robotwedstrijden er bij te bedenken voor volgend jaar. Bijvoorbeeld wie het snelste een doolhof kan oplossen, of iet dergelijks. Het Sumo worstelen is misschien ook nog steeds een optie als er tenminste mensen zijn die dit soort robotten willen bouwen en uit te testen tegen tegenstanders. Want alleen sumoën* is niet leuk he.

Natuurlijk als iemand ideeën heeft, is het altijd mogelijk deze aan het bestuur voor te stellen en gaan we natuurlijk hier naar kijken hoe we dat kunnen vormgeven. Want ten slotte moet het voor iedereen leuk en aantrekkelijk worden/zijn om met de wedstrijden mee te kunnen doen.

Ik ga in ieder geval het internet afspeuren naar leuke ideeën om inspiratie op te doen.
Ik zou zeggen tot de wedstrijden allemaal.

Met vriendelijke groet,

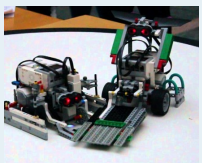
Bert Berrevoets.

(* is sumoën eigenlijk wel een woord?)

IN DIT NUMMER

SUMO worstel ideeën

Sumo (相撲 Sumō) of sumoworstelen is een traditionele Japanse worstelsport die meestal wordt beoefend door zeer zwaarlijvige mannen. De worstelpartij vindt plaats in een cirkelvormig deel van een kleibodem en gaat gepaard met vele rituelen. De Japanners beschouwen Sumo als een moderne krijgskunst; het is daarbij een populaire sport die vaak op televisie uitgezonden wordt.



RETRO ARTIKEL: Leren programmeren met RoboMind

Introductie
Ik ben Arvid Halma en studeer momenteel Kunstmatige Intelligentie aan de Universiteit van Amsterdam.
Deze richting houd zich bezig met de vraag hoe robots en computers zelf problemen kunnen oplossen en advies kunnen geven in moeilijke situaties.

Vanwege deze achtergrond ben ik met name geïnteresseerd geraakt hoe je een robot zo ver krijgt dat hij zelf weet wat hij moet doen, zonder dat ik zelf stiekem alles hoef in te voeren. Ten eerste is dat vaak heel saai werk, maar vaak weet je zelfs niet in welke situatie de robot terecht zal komen.

Denk bijvoorbeeld aan de Mars Explorer van NASA die in een totaal onbekend terrein interessante monsters moet nemen zonder ergens tegenaan te botsen.

In zo'n situatie moet je iets slims bedenken en de instructies zó kiezen dat de robot er het beste van maakt op het moment dat hij er is. Het definiëren van de instructies is programmeren. In dit artikel zal ik degenen die geen ervaring hebben met programmeren proberen een indruk te geven waarom dit eigenlijk het leukste/belangrijkste/uitdagendste onderdeel is bij het werken met computers en robots en hoe het programma RoboMind je bij deze introductie van dienst kan zijn.

Programmeren? Gewoon doen!
Over programmeren kan je hoop lezen, maar eigenlijk moet je het gewoon doen. Dan begin je pas te begrijpen hoe je robots en computers kunt laten doen wat je wilt en hoe soms ingewikkelde zaken eenvoudig kunnen worden opgelost.

Om snel aan de slag te kunnen is er een nieuwe en eenvoudige programmeertaal gemaakt, Robo, zodat je snel aan de slag kunt zonder eerst een tijd in de boeken te duiken. Het is speciaal gemaakt om op simpele wijze een robotje te programmeren dat wordt gesimuleerd in RoboMind.

Een puzzel op Mars
Zei ik net dat je nog niets van programmeren hoeft af te weten? Bij deze meteen maar een lastig probleem en met een lastig programma als oplossing. Onze Mars Explorer is namelijk zojuist geland en toevallig blijkt deze planeet bewoond te zijn. Helemaal vlekkeloos ging dat niet, want de robot is midden in een door de inboorlingen gebouwd doolhof verzeild geraakt. Zonder dat we een plattegrond of ook maar iets weten, komen we toch graag heelhuids bij het eindpunt aan. Dit eindpunt is te herkennen aan een groen baken. Allemaal niet erg realistisch, maar hoe zou je dit aanpakken?

'Zei ik net dat je nog niets van programmeren hoeft af te weten? Bij deze meteen maar een lastig probleem en met een lastig programma als oplossing.'

Een oplossing
Je kunt verschillende strategieën bedenken, en een daarvan is de volgende: volg altijd de muur aan de rechterkant. Anders gezegd:
ga op een recht stukje rechtdoor, maar zodra je rechtsaf kunt moet je dat doen. Als je tegen een muur opbotst ga je links met de bocht mee. Door altijd langs de rechter muur te lopen kom je gegarandeerd bij het einde. Wiskundigen kunnen aantonen dat dit echt altijd werkt, ook al klinkt dit misschien ongeloofwaardig (die zijn overigens vaker met dit soort leuke dingen bezig dan je denkt, maar dat terzijde).

Het bijbehorende programma
Deze kennis kunnen we gebruiken om onze robot te helpen. De bovenstaande uitleg zal de robot niet direct begrijpen, daarvoor moeten we preciezer uitspellen wat er moet gebeuren. Hiernaast zie je een programma dat de robot wel kan uitvoeren.Je hoeft niet alles meteen te begrijpen, daarvoor staat er een uitgebreide introductie online, maar je kunt het redelijk ontcijferen met de volgende uitleg.

We beginnen voor het gemak op regel 3, en hebben als doel om maar een klein stapje in de goede richting te doen.
We kijken eerst hoe de situatie ervoor staat.

Als er rechts ten opzichte van de robot een obstakel (muur) is, moeten we kijken of we een stapje vooruit kunnen doen (5). Als het voor vrij is gaan we ook echt een stapje vooruit (7) en anders was er zowel rechts als voor een muur en gaan we maar naar links (11). Tot dusver hebben we alleen het geval uitgewerkt waarbij we rechts een muur zagen. Als er rechts echter geen obstakel was, zoals op regel 3 is gecontroleerd, doen we iets anders. We zorgen dat we de muur rechts weer tegenkomen door naar rechts te draaien en een stapje vooruit te gaan (16, 17).

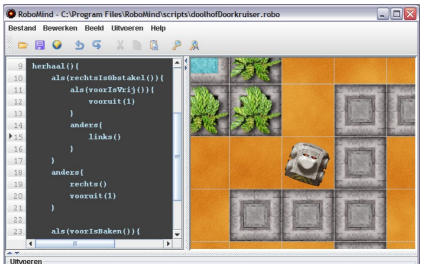
Nu we zeker weten dat we stapje in goede richting hebben gedaan door de muur een stukje te volgen kunnen we kijken of we al bij het eindpunt zijn. Dit eindpunt was herkenbaar aan een baken. Dus alleen als het baken recht voor je staat kun je dit oppakken (22) en mag het programma helemaal stoppen (23). Goed. We hebben nu voor elke mogelijke situatie netjes opgeschreven wat er moet gebeuren.
Door telkens maar zo'n heel klein stapje te blijven herhalen kom je vanzelf bij het eindpunt en zal de robot bij het einde komen en stoppen.

```
1   herhaal()
2   {
3       als(rechtsIsObstakel())
4       {
5           als(voorIsVrij())
6           {
7               vooruit(1)
8           }
9           anders
10          {
11              links()
12          }
13      }
14      anders
15      {
16          rechts()
17          vooruit(1)
18      }
19
20      als(voorIsBaken())
21      {
22          pakOp()
23          einde
24      }
25  }
```

Daarom stoppen we de regels 3 t/m 24 in een zogenaamd herhaal(){...} blok, dat er voor zorgt dat deze regels net zo lang worden uitgevoerd totdat einde wordt tegengekomen.

Maar...
Hoe weet je nou welke woorden de robot wel en niet begrijpt? Tot nu toe is er geen touw aan vast te knopen! Dat klopt, wat de robot wel en niet begrijpt moet er domweg bij worden verteld, en dat heb ik van te voren niet gedaan. In de handleiding staat dit daarom wel netjes van te voren verteld. Sommige onderdelen komen echter altijd weer voor, soms met een iets andere benaming. Zo heb je altijd wel herhalingen, herhaal(){...}, en altijd wel de mogelijkheid om zogenaamde conditionele uitspraken te doen, als(...){...}anders{...}.

Waarom is het programma zo vreemd opgemaakt? Het inspringen van regels, ofwel de indentatie, doet er voor de robot niet toe. Het is bedoeld voor ons zelf om beter te zien wat bij elkaar hoort. Nu kun je bijvoorbeeld in een keer zien dat regel 4 t/m 13 hoort bij als(rechtsIsObstakel()). Voor de robot zelf hoef je slechts te zorgen dat je de namen goed schrijft en dat de haakjes goed staan.



Programmeren is dus helemaal niet eenvoudig, je moet maar net op de oplossing komen! Dat is voor een deel wel waar. Het blijft altijd gepuzzel om een idee goed te vertalen naar een programma. Gelukkig kan je goede oplossingen vaak hergebruiken en zijn er vaak een hoop verschillende manieren om het aan te pakken en op te schrijven.

'...Please, try this at home ...'

Please, try this at home!
Probeer dit programma eens uit in RoboMind, dat je gratis kunt downloaden van www.robomind.net. Je hoeft het voorgaande programma niet over te typen, want het is er standaard bijgeleverd.
Als je RoboMind hebt geïnstalleerd en opgestart doe je het volgende:
1. Ga naar Bestand > Open en selecteer doolhofDoorkruiser.robo. Hiermee heb je de doolhof-code geladen die net besproken is.
2. Ga naar Bestand > Open kaart en selecteer maze1.map. De robot zit nu in een andere omgeving, namelijk een doolhof.
3. Druk onderin het scherm op uitvoeren om het script uit te voeren. De robot zal nu stapje voor stapje dichter bij zijn doel komen en uiteindelijk stoppen.

Om een beter overzicht te krijgen kun je het beeld nog uitzoomen via Beeld > Zoom uit.

Je programmeercarrière uitgestippeld
Als je het fijne wilt weten van RoboMind raad ik je aan gewoon de introductie eens van de website te downloaden en er mee aan de slag te gaan. Nu je dit artikel hebt gelezen zou dat goed te doen moeten zijn, omdat het daar allemaal veel rustiger wordt opgebouwd. Ik denk dat het een aardige manier is om kennis te maken met het vakgebied.

Als je denkt dat programmeren inderdaad wel wat voor je is, kan je een programmeertaal leren waar je veel meer mee kan dan alleen deze simulatie robot programmeren. Er zijn honderden talen die je dan kan overwegen met elk hun eigen voordelen en eigenaardigheden. Programmeurs treden echter nog al eens op als regelrechte fundamentalisten en verkondigen hun geloof in een bepaalde taal alsof hun leven er van afhangt. Ik durf slechts voorzichtig Python (www.python.org) of Java (java.sun.com) aan te raden.

Tegen deze tijd is het boek Gödel, Escher, Bach van Douglas R. Hofstadter een klassieker die bijna instantane verlichting brengt over bijna elk denkbaar nerd-onderwerp (hierover durf ik minder voorzichtig te zijn).
Mocht je nou al een doorgewinterde programmeur zijn èn nog steeds dit artikel lezen, dan raad ik je aan eens inspiratie op te doen met Haskell (www.haskell.org) dankzij de elegante en veelbelovende andere aanpak.
Wanneer je deze gereedschappen op orde hebt, hoef je vervolgens alleen nog maar echte kunstmatige intelligentie uit te vinden!... want mij lukt dat nog niet zo goed.

buzz

Gelezen op Slashdot :

Buzz is een nieuwe open source programmeer-taal die toelaat om interacties tussen individu-ele robots te modelleren.

Bekijk ook het filmpje van een simulatie.

https://github.com/MISTLab/Buzz

En dan te bedenken dat ik al last heb met 'Heen en Weer met opties' :-)

Met vriendelijke groeten,
Patrick.

University of Reading

COURSE:

Begin Robotics

Robots today are roving Mars, collecting data in dangerous environments, hoovering our floors, lifting patients in hospital, building cars and entertaining us in films. And, if you share Bill Gates and Stephen Hawking's world view, the super intelligent ones may one day bring about the end of the human race.

<lees hier meer

ROBOT MAGAZINE

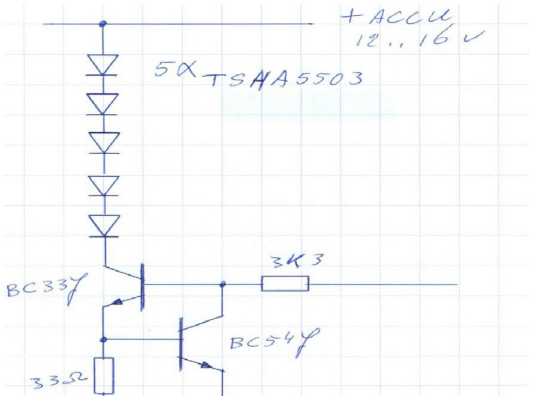
Welcome to our weekly round-up of robot news from our website, www.botmag.com!

LIJNVOLGEN

Lijnvolgen simpel?? ja, maar dat hoeft niet.

Ik zie bij de bijeenkomsten in Hooglanderveen steeds mensen worstelen met lijnvolgen. Graag wil ik mijn bedenkingen en ervaringen delen, misschien dat iemand er zijn voordeel mee kan doen.

Als eerste de sensor, daar komen de gegevens vandaan. Ik heb gekozen voor een sensor met vier IR foto transistor's, nl. BPW40 van eoo-bv.nl (nee geen aandelen).

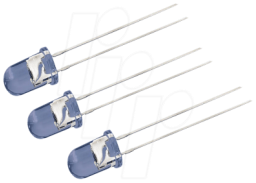


Die staan tussen vijf IR led's (eoo : TSHA5503) op een rij.

Zo dat iedere fotodiode links en rechts een led heeft, met de bedoeling dat ze allemaal evenveel licht krijgen. De IR led's staan alle vijf in serie

De sensor zit 10mm boven de vloer, dit mag zelfs nog iets meer zijn. En 12cm voor de hartlijn van de wielen. De foto transistors staan 15mm hart op hart. De sensor wordt analoog ingele-zen.

```
int lijn_center(void)
{
int LL = read_adc(4);
int L = read_adc(5);
int R = read_adc(6);
int RR = read_adc(7);
int ret = ((LL * 2) + L ) (
R + (2 * RR));
return ret;
}
```

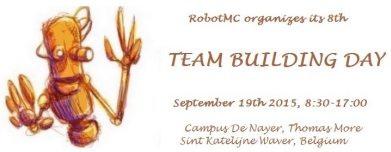


In theorie is return waarde dan 0 met lijn in het midden, maar ook zonder lijn is de return waarde 0. De praktijk is anders omdat er verschillen zitten in de IR led's en de foto transistors. Met gevolg dat, met de lijn in het midden, de waarde geen 0 is. Met kalibreren van de sensor lees ik de sensor zonder lijn, het getal wat ik dan krijg inverteer ik en tel dat bij de uitkomst van de sensor op.

Als ik zonder lijn bv. 80 krijg, tel ik 80 bij de sensorwaarde op. Dan krijg ik zonder lijn, of de lijn in het midden, wel een 0 als sensoruitlezing. Bij het meten van de sensor zag ik dat de waarde in het begin niet stabiel is. Na inschakelen is de sensorwaarde bij mij pas na 3 minuten een redelijk stabiele waarde. Ik denk dat het met opwarmen van de elektronica te maken heeft. Hier dus rekening mee houden bij een wedstrijd of demonstratie, robot op tijd aan zetten.

Dan de regeling, een PD regeling, eerst proportioneel instellen met de D factor op 0.

RobotMC: TEAM BUILDING DAY 2015

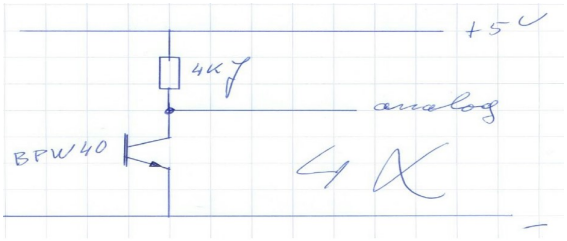


In 2015 was het thema 'kwaliteitscontrole'.
De bedoeling was om 1, 2 en 3 wielen helemaal te inspecteren op fouten.

Voor resultaten en verslag ga naar de website van RobotRMC

Vervolg lijnvolgen

```
int Kp = 3,
Kd = 0,
snh = 50; // Gewenste snelheid
fout = lijn_center();
D = fout vorige_fout;
vorige_fout = fout;
correctie = ((Kp * fout) + (Kd * D))>>3;
motors(Gsnh correctie, Gsnh + correctie);
```



Zoals je ziet schuif ik correctie 3 bits naar rechts, dat is delen door 8. Kp is dan 3/8 dit is 0,375 8 bit micro controllers zijn niet goed in float getal berekeningen. Dus ik probeer zoveel mogelijk integers te gebruiken, en voor delen naar rechts schuiven. En vermenigvuldigen naar links schuiven, daar zijn ze dan weer wel goed in. Als de stapjes voor Kp en Kd kleiner moeten, dan schuif ik 4 bits. Kp wordt dan 6/16 is ook weer 0,375, maar nu kunnen kleiner stapjes gemaakt worden. Terug naar de regeling.

Bij een lage snelheid steeds de Kp waarde ophogen totdat de robot begint te slingeren op de lijn. Dan de Kd waarde ophogen totdat de robot mooi stabiel de lijn volgt. Dan de gewenste snelheid verhogen. Kd laten staan en de Kp aanpassen totdat de robot net niet slingert. Kd aanpassen. Dan de snelheid verhogen enz. enz. Totdat het fout gaat, robot vliegt uit de bocht. Maximum snelheid bereikt, of toch niet? Op deze manier van corrigeren blijft de gemiddelde snelheid hetzelfde als de gewenste snelheid.

Op een recht stuk en in de bochten. De opdracht is "zo snel mogelijk een lijn volgen". Maar in een bocht kun je niet zo hard als een recht stuk. Dus in een bocht afremmen, maar hoeveel en wie bepaald wanneer. Op een club bijeenkomst maakte iemand de opmerking "en als je alleen het binnenwiel afremt". Daar heb ik over nagedacht en een en ander geprobeerd. Wat gebeurd er als je het binnenwiel afremt en het buitenwiel de gewenste snelheid laat draaien.

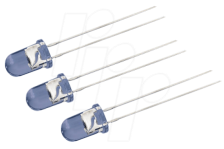
Stel de bocht is zo scherp dat het binnenwiel stil staat, met het buitenwiel op de gewenste snelheid geeft dat een gemiddelde snelheid van de halve gewenste snelheid. Met een niet zo scherpe bocht draait het binnenwiel wel en krijg je dus een hogere gemiddelde snelheid. Op een recht stuk draaien beide wielen op de gewenste snelheid, gemiddelde snelheid is dan gewenste snelheid. DIT IS PRECIES WAT WE NODIG HEBBEN. Bedankt voor de opmerking.

Op een recht stuk op volle snelheid en hoe scherper de bocht hoe lager de snelheid. Zo we zijn er, of nog steeds niet? Hoe werkt een lijnsensor eigenlijk?

Lijnsensor op 0 lijn in het midden, bij een afwijking wordt er gecorrigeerd dan krijg je dus een evenwicht situatie. Lijn uit het midden, wielen draaien met verschillende snelheid. Zoals in een bocht, dan ook een evenwicht situatie. Nu snap ik ook waarom je geen I factor kunt gebruiken met lijnvolgen. In een bocht hebben we een evenwicht situatie als de sensor niet op 0 staat, de I factor blijft dan corrigeren terwijl het niet nodig is. Maar als in een bocht de lijn uit het midden is gebruiken we daar maar een klein gedeelte van de lijnsensor. Vandaar dat het in een bocht eerder fout gaat dan op een recht stuk. We kunnen de sensor met een software foefje wat breder maken. We maken 2 variabelen aan Lbs en Rbs, Links buiten sensor en Rechts buiten sensor. Mijn lijnsensor gaat van 250 naar +250 nu maak ik Lbs 1 als de sensor kleiner is dan 240 en weer 0 als hij groter is dan 235 en Rbs wordt 1 als sensor groter is dan 240 en weer 0 als hij kleiner is dan 235 en een variabele bijsturen die zetten we even op 25 de regel in lijn_center()

```
int ret = ((LL * 2) + L ) (
R + (2 * RR));
wordt dan
int ret = ( (Lbs * bijsturen) + (LL * 2) + L ) (
R + (2 * RR) + (Rbs * bijsturen) );
```

Maar dit is geen mooie regeling, het is op het laatste moment een ruk aan het stuur om weer op de lijn te komen. Je moet wel experimenteren met de getallen wanneer 1 en 0 maken en hoeveelheid bijsturen. Het gewicht van de robot heeft veel invloed hoe hierop wordt gereageerd. Als de robot helemaal van de lijn is worden de variabelen ook op 0 gezet. Om dat op te vangen moeten de twee binnenste sensors ook digitaal ingelezen worden. En daar dan de variabelen mee op 0 zetten. Je zou ook nog kunnen kijken hoelang Lbs of Rbs 1 zijn, als dat te lang duurt kun je ook nog de gewenste snelheid verlagen totdat Lbs en Rbs weer 0 zijn.



Dan nog meer experimenteren met de lijnsensor. hoeveel stroom moet er eigenlijk door de IR led's, kan ik ook teveel / te weinig licht hebben grafieken maken voor een beter overzicht wat er gebeurd. er is dus genoeg te doen om te verbeteren. je kunt het zo mummie maken als je zelf wilt.

PS: Jan Heynen heeft een mooi verhaal over de lijnsensor op RoboWiki gezet
http://wiki.robotmc.org/index.php?title=Lijnvolgen
PPS: een mummie is ingewikkeld

hcc robotica

Bezoek ook eens onze website:
www.hccrobotica.nl

RoboRama

De jaarlijkse HCC robotica Roborama wedstrijden komen er weer aan. Op **zaterdag 7 november 2015** gaan de robots weer ten strijde in de bak.

De wedstrijd bestaat in ieder geval uit volgende disciplines :

- Heen en weer
- Lijnvolgen
- T-Tijd
- Blikken zoeken

De regelementen voor de wedstrijden zijn hier te vinden.

Het aantal sumo-robots is in de loop der jaren afgenomen er zijn er waarschijnlijk niet genoeg voor een serieuze wedstrijd. Mocht dit het geval zijn dan wordt het een demonstratie sport zodat er volgend jaar misschien wel weer genoeg deelnemers zijn.

RobotMC - de club van ROBOTica en Micro Controller enthousiastelingen

Campus De Nayer in Sint Katelijne Waver Belgie

Datum	Plaats
17 oktober 2015	Thomas More
22 november 2015	Thomas More
Zondag!	WETENSCHAPSDAG
19 december 2015	Thomas More
16 januari 2016	Thomas More
20 februari 2016	Thomas More

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

Agenda HCC!Robotica 2015

- 3 oktober Maandelijkse bijeenkomst in de Dissel
- 7 november Roborama wedstrijd in de Dissel
- 5 december Maandelijkse bijeenkomst in de Dissel

Agenda HCC!Robotica 2016

- 2 januari Maandelijkse bijeenkomst in de Dissel
- 13 februari Maandelijkse bijeenkomst in de Dissel LET-OP een week later in verband met carnaval.

Overige activiteiten:

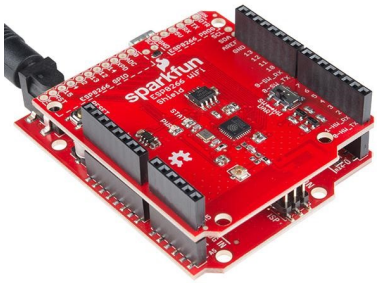
- 3-4 Oktober weekend van de wetenschap http://www.hetweekendvandewetenschap.nl/deelnemers-2015/
- 25 OKTOBER 2015 JUBILEUM ROBOTRACE CCFZ Bergen op Zoom http://www.robotics-ccfz.nl/Race2015.html

Discussiegroepen
Maken en delen met groepen :
HCCROBOTICA:
http://groups.google.nl/group/hcc_robotmc

Blogs
http://zotten.wordpress.com/
http://waarisdievanjou?
http://www.robotblog.nl/

WiFi Shield voor #Arduino

De SparkFun ESP8266 WiFi Shield is een Arduino compatible shield voor de ESP8266 WiFi chip. De ESP8266 WiFi Shield is voorzien van de AT-command firmware, zodat het bestuurd kan worden via de seriële interface. Alle I/O's van de ESP8266 zijn beschikbaar en kan met commando's benaderd worden.



HCC!Robotica ig

Dagelijks bestuur:
Voorzitter : Bert Berrevoets
Secretaris : Edith van Putten
Penningmeester : Raoul Boerlage

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