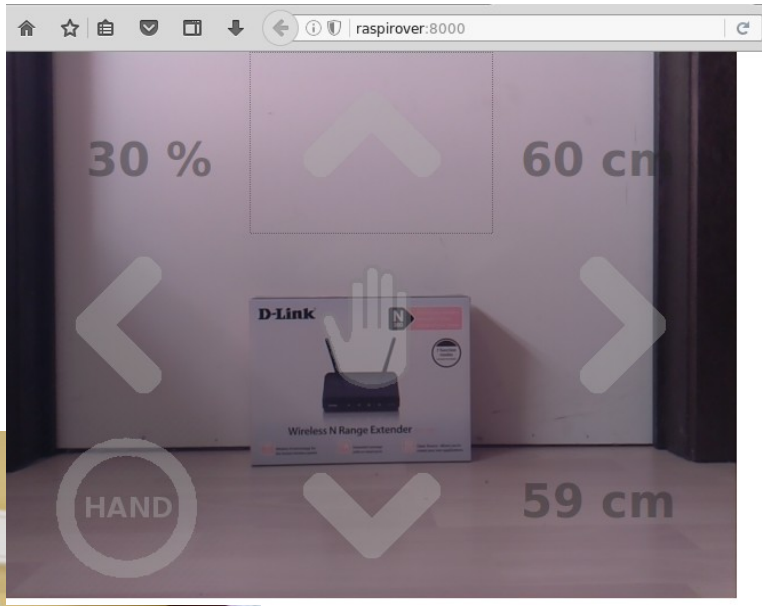
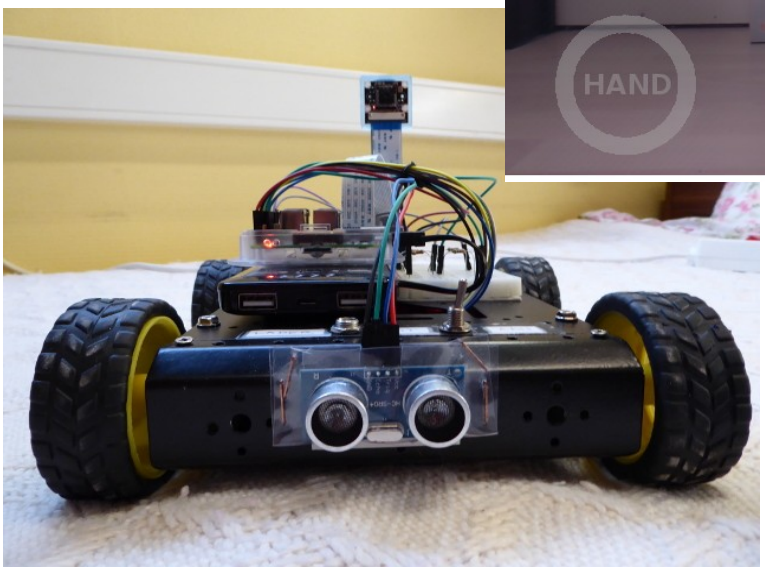
Eindproef cursus Raspberry Pi :

Effevee's Rover





Alle documentatie en code van dit project kan je hier downloaden. De code is ruim voorzien van commentaar en uitleg. Mocht je nog met vragen zitten, stuur dan gerust een email naar effevee AT gmail DOT com.

Enjoy,

Effevee.

Inhoudsopgave

Beschrijving

Effevee's rover is een robotvoertuig dat op afstand bestuurd wordt en ondertussen beelden streamt van de onboard Pi camera. De aanwezige afstandssensoren zorgt ervoor dat botsingen met obstakels worden vermeden. De stuursoftware is geschreven in python en wordt bediend via een web interface. De Raspberry Pi is voorzien van een wifi module zodat we vanaf iedere pc, tablet of smartphone met een verbinding op het netwerk het voertuig draadloos kunnen besturen. In principe kan je zelfs via port forwarding op je router het voertuig besturen over het internet. Als uitbreiding is er ook een autonavigatie modus van het robotvoertuig toegevoegd.

Hardware

Controller

• Raspberry Pi 2	44,99 €
• EW-7811Un Edimax Wifi usb module	9,79 €
• Pi NoIR camera module	26,73 $
• Medion MD84610 powerbank 5V	14,99 €

Robotvoertuig

• 4 wiel voertuig chassis met 4 gelijkstroommotoren	37,99 $
• motorstuurmodule L298N	6,88 $
• NiMH 9,6V 1300mAh accupack	27,99 €
• Voltcraft snellader	24,99 €
• afstandssensoren HC-SR04	2x 3,70 $
• klein breadboard	7,95 €

Allerlei

• Digitale multimeter	
• Top Craft Soldeerboutset en soldeertin	9,99 €
• Kableerdraad 0,2 mm^2	9,99 €
• Jumperkabels v/v en m/v	2x 1,84 $
• Velcro en/of Tesa bevestigingsband	3,62 €
• Afstandhouders	
• Schroeven en moeren M3	

Controller

Raspberry Pi 2

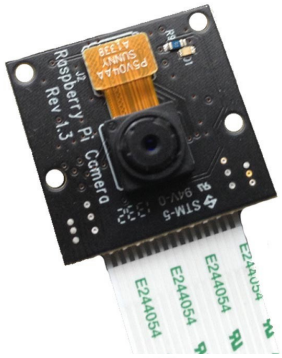
Ik gebruik uiteraard de vertrouwde Raspberry Pi 2 als controller voor dit project. Andere modellen Raspberry Pi kunnen uiteraard ook gebruikt worden.

Het robotvoertuig zal bestuurd worden door een Python script dat via de GPIO pinnen communiceert met de motorsturing en de afstandssensoren.

Wifi module

Ik gebruik de EW-7811Un Edimax wifi module om draadloze communicatie met het robotvoertuig mogelijk te maken.

Video streaming



Voor de video streaming maak ik gebruik van de Pi NoIR camera module die rechtstreeks op het Rpi bordje wordt aangesloten met het kabellint.

Voeding

Ik maak gebruik van een Medion MD 84610 powerbank van Aldi om de RPi2 te voeden.
Iedere stabiele 5V voeding voldoet uiteraard ook.

Robotvoertuig

4 wiel voertuig chassis

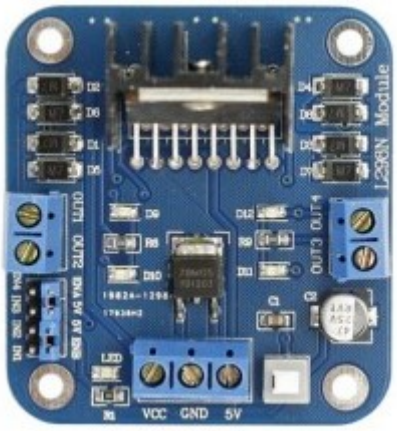
Dit chassis wordt onder andere verkocht door Sainsmart. Het bevat de volgende onderdelen :

Monteer het voertuig volgens deze instructies op de website met dit als resultaat :

Motorsturing

Stuurmodule L298N

Deze module wordt onder andere verkocht door Sainsmart.



De L298N (gemonteerd op de grote koelvin) is een zogenaamde H-bridge. Die worden veel gebruikt om motoren aan te sturen.

Deze module kan dienen voor zowel het sturen van 2 stappenmotoren (servomotoren) als van 2 gewone DC motoren. In dit project worden de linker- en rechter DC motoren parallel geschakeld op de module.

Voor het aansturen van een stappenmotor moet de jumper over de enable pin en de 5V pin geplaatst worden.

Voor het aansturen van een DC motor kan je een PWM signaal op de enable pin plaatsen om de draaisnelheid van de motoren aan te passen.

De draairichting van de motoren wordt bepaald door de signalen die op de In aansluitingen staan. De combinatie LAAG – HOOG laat de motor vooruit draaien; de combinatie HOOG – LAAG achteruit; de combinatie LAAG – LAAG stopt de motor.

Wanneer een externe voeding (>5V) is aangesloten op VCC, dan kan je met de 5V aansluiting op de module de Raspberry Pi voeden. **Uit praktijktesten is echter gebleken dat deze voeding niet stabiel genoeg is en willekeurige reboots van de Rpi veroorzaakt**. We voorzien dus een aparte voeding (Medion Powerbank) voor onze Pi.

L298N aansluitingen

Pin L298N	Functie
Out1	Motor A aansluiting +
Out2	Motor A aansluiting -
Out3	Motor B aansluiting +
Out4	Motor B aansluiting -
VCC	Voedingsspanning (5V – 35V)
GND	Massa
5V	5V output als VCC reeds spanning heeft > 5V **(niet betrouwbaar)**
EnA	Motor A : met jumper = stappenmotor – zonder = DC motor snelheid via PWM
EnB	Motor B : met jumper = stappenmotor – zonder = DC motor snelheid via PWM
In1, In2	Motor A draairichting : vooruit = LAAG-HOOG / achteruit = HOOG-LAAG
In3, In4	Motor B draairichting : vooruit = LAAG-HOOG / achteruit = HOOG-LAAG

Voeding

Accupack

De voeding die ik voor de aandrijving van de rover gebruik is een 9,6V/1300 mAh NiMH accupack die je onder andere bij Conrad kan aanschaffen.

Deze voor gemonteerde accupack bestaat uit 8 NiMH cellen met een vermogen van 1300 mAh en een spanning van 9,6V. Het geheel is gemonteerd met een mini Tamiya stekker aansluiting.

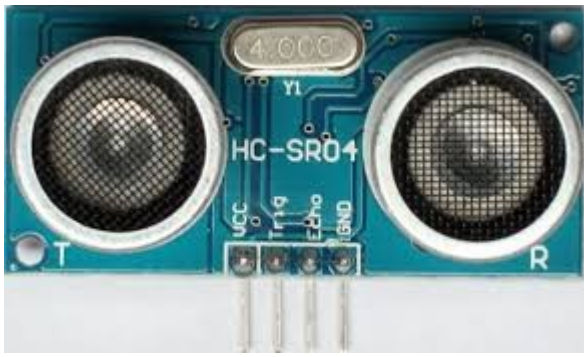
Snellader

Ik heb er ook een Volcraft 230V snellader bijgekocht. Deze is geschikt voor NiCd en NiMH accupacks van 5 tot 8 cellen met een maximum vermogen tot 4300 mAh en heeft een instelbare laadstroom van 1A of 2A.

Dit is de ideale snellader voor racingpacks met Tamiya-stekkers. De betrouwbare Delta-V-uitschakeling garandeert een maximum lading zonder dat de accu wordt overladen. De laadtoestand wordt mbv 3 LED's aangegeven.

Afstandssensoren

Afstandssensor HC-SR04



De HC-SR04 kan je onder andere kopen bij Sainsmart.

De sensor werkt volgens hetzelfde principe als de sonar van een duikboot. De sensor heeft een bereik van 2 tot 400 cm.

De Trigger pin wordt gedurende minstens 10 µs hoog gezet. Hierdoor worden 8 ultrasone geluidsignalen van 40 kHz uitgezonden vanaf de linker sensor (T van Transmit) die werkt als een mini luidspreker. De geluidsignalen worden door obstakels gedeeltelijk teruggekaatst en opgevangen door de rechter sensor (R van Recieve). Hierdoor wordt de Echo pin een tijd hoog evenredig met de afstand tot het obstakel.

Geluidsignalen hebben een snelheid van 340 m/s in lucht. De tijd dat de echo puls hoog is, is equivalent met 2 x de afstand tot het obstakel. De formule om de afstand te berekenen wordt dus :

Afstand (in cm) = Snelheid (34000 cm/s) x Echo tijd (in s) / 2

HC-SR04 aansluitingen

Pin HC-SR04	Functie
VCC	Voeding +5V
Trigger	Output
Echo	Input
GND	Voeding -

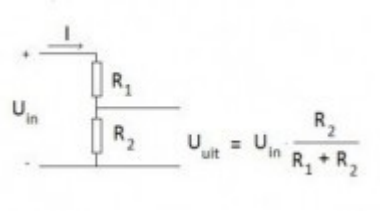
Robotvoertuig

Schakeling

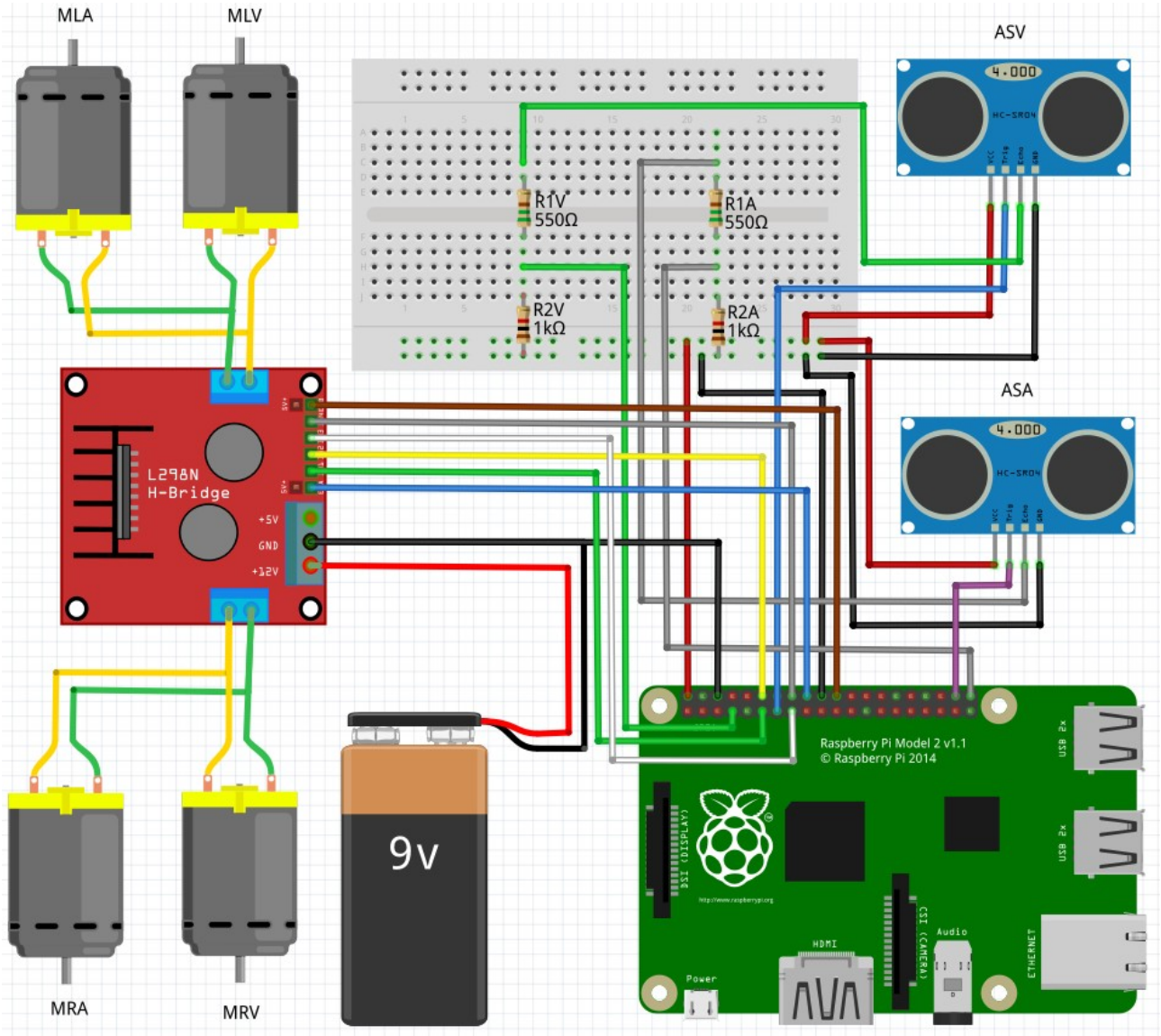
We verbinden de verschillende onderdelen als volgt met elkaar :

Van		Jumper	Naar		
Onderdeel	**Pin**	**kleur**	**Onderdeel**	**GPIO**	**Fysiek#**
L298N stuurmodule	OUT1	zwart	Motoren links		-
	OUT2	geel/rood			+
	OUT3	groen/blauw	Motoren rechts		+
	OUT4	zwart			-
	VCC	rood	Accu 9,6V		+
	GND	zwart			-
		zwart	Breadboard -		
	IN1	groen	RPi2	17	11
	IN2	geel		18	12
	IN3	wit		22	15
	IN4	grijs		23	16
	ENA	blauw		24	18
	ENB	bruin		25	22
Breadboard	-	zwart	RPi2	GND	6
	+	rood		5V	2
HC-SR04 Voor	VCC	rood	Breadboard +		
	Trig	blauw	RPi2	27	13
	Echo [°]	groen		4	7
	GND	zwart	Breadboard -		
HC-SR04 Achter	VCC	rood	Breadboard +		
	Trig	purper	RPi2	20	38
	Echo[°]	grijs		21	40
	GND	zwart	Breadboard -		

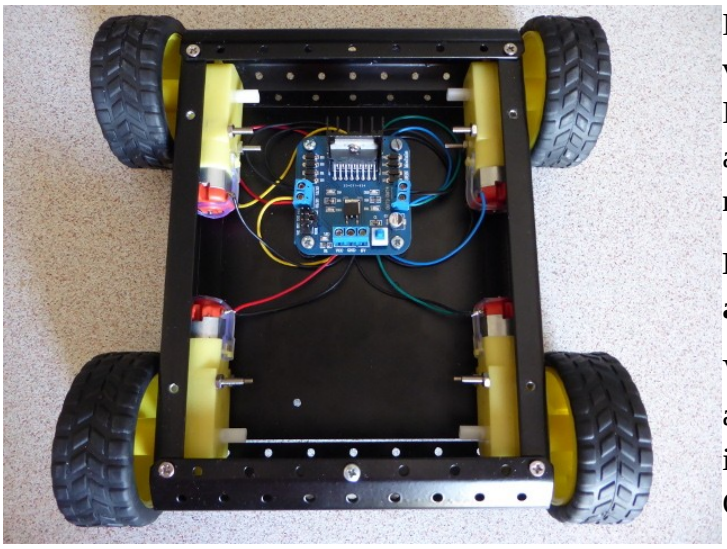
[°] De afstandssensoren HC-SR04 worden gevoed door 5V van de Rpi. Het signaal op de echo pin van de sensor moet uitgelezen worden via een GPIO pin. Er mag op die pinnen maximaal 3,3V komen. We maken dus gebruik van een spanningsdeler om de 5V omlaag te brengen naar 3,3V.



Hier heb ik gebruik gemaakt van weerstanden :
R1 = 550 KΩ
R2 = 1 KΩ.
Hierdoor wordt de U_{uit} = 5 * (1000/(550+1000)) = 3,2 V



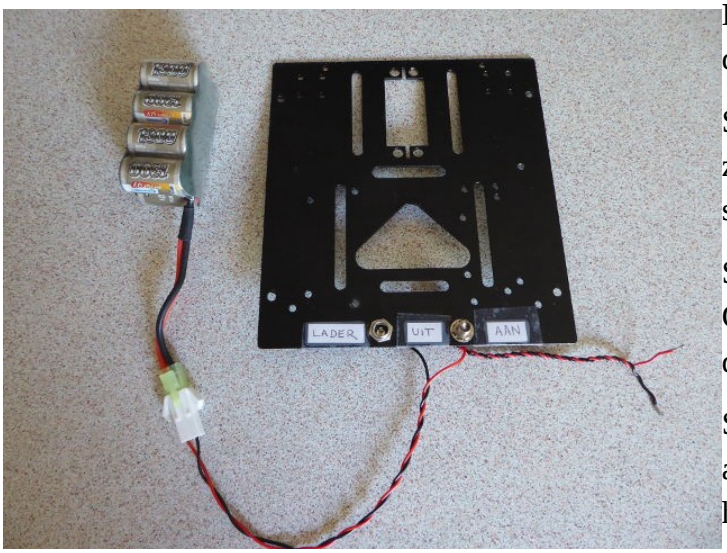
Montage



De L298N stuurmodule is op de onderplaat van het robotvoertuig chassis gemonteerd. De linker motoren zijn verbonden met de aansluitingen OUT1 en OUT2; de rechter motoren met OUT3 en OUT4.

Let op met de polariteit van de aansluitingen !

Verbind 6 jumper kabels met de andere aansluitingen van de L298N. Die verbind ik straks doorheen de bovenplaat met de GPIO pinnen van de Rpi.

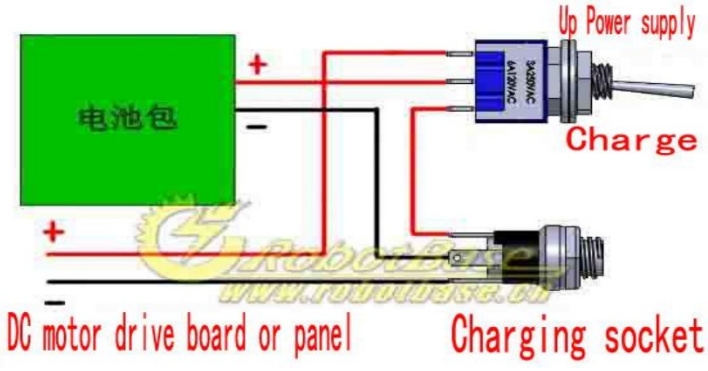


De aan/uit schakelaar en de connector voor de lader zijn in de bovenplaat gemonteerd.

Sluit de rode kabel aan op VCC en de zwarte op GND van de L298N stuurmodule.

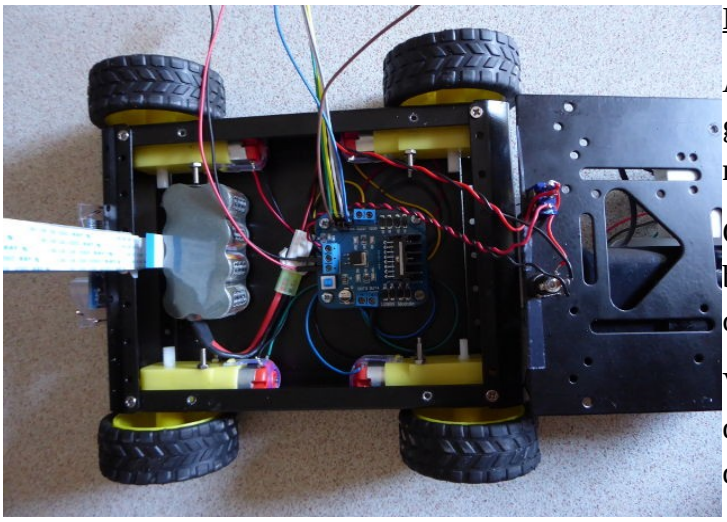
Sluit ook een zwarte jumper kabel aan op GND . Die verbind ik straks met GND op de Rpi om een gesloten kring te maken.

Sluit de Tamiya stekker aan op de accupack en bevestig deze met velcro in het chassis.



Dit is het bedradingsschema voor de verbinding van accupack, aan/uit schakelaar, oplaadconnector en L298N stuurmodule.

Test alles grondig na met je digitale multimeter vooraleer je de accupack en de stuurmodule aansluit.

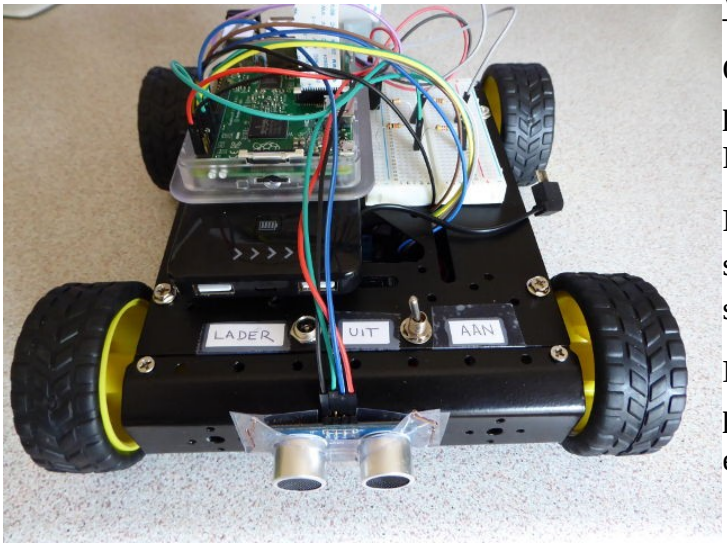


Binnenzicht :

Alle aansluitingen met de L298N zijn gemaakt en de jumper kabels voor de motorsturing zijn geplaatst.

Geleid de jumper kabels doorheen de bovenplaat en schroef deze vast op het chassis.

Verbind de jumper kabels van de L298N en de HC-SR04 sensoren met de Rpi volgens de schakeling tabel.

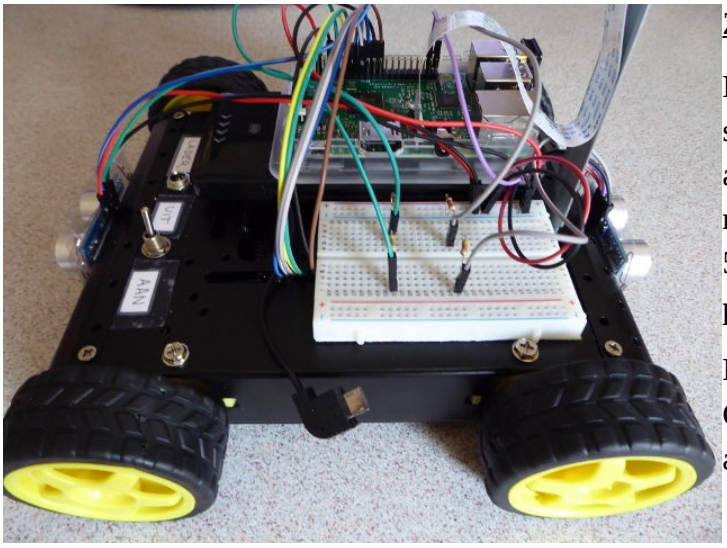


Vooraanzicht :

Op de bovenplaat heb ik links de Medion powerbank met daar bovenop de Raspberry Pi bevestigd.

Rechts is een klein breadboard voor de spanningsdelers van de HC-SR04 afstand sensoren.

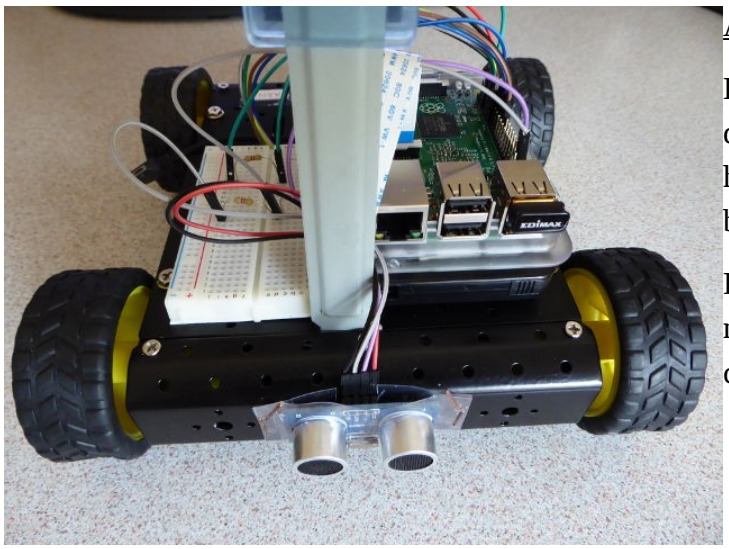
De HC-SR04 heb ik met een stuk harde plastic en Tesa band bevestigd op de voor- en achterzijde van de chassis.



Zijaanzicht :

Hier zie je beter het breadboard met de spanningsdelers van de voorste en achterste afstand sensoren. De spanningsdelers zijn nodig omdat de HC-SR04 enkel werken op 5V en de Rpi maximaal 3,3V op zijn gpio pinnen mag krijgen.

Bemerk ook de lint connector van de Pi Camera die rechtstreeks op de Rpi is aangesloten.



Achteraanzicht :

Bemerk rechts de Rpi met zijn Edimax usb draadloze wifi module. In het midden de houder voor de Pi Camera en links het breadbord met de spanningsdelers.

De achterste HC-SR04 is op dezelfde manier bevestigd op de achterzijde van het chassis.

Software

Effevee's rover project draait op een Raspberry Pi met alleen open source software.

Operating systeem

Raspbian

Voor dit project gebruik ik Raspbian. Dit is een Debian Linux Wheezy gebaseerd operating systeem dat speciaal is aangepast voor de Raspberry Pi hardware.

Het is echter meer dan een operating systeem : het bevat meer dan 35.000 gecompileerde software componenten voor elke denkbare toepassing zodat installatie van software zeer gebruiksvriendelijk is.

Raspian is niet gebonden aan de Raspberry Pi Foundation. Het is gemaakt door een klein team van ontwikkelaars die de Raspberry Pi hardware, het Debian Linux project en de educatieve doeleinden van de de Raspberry Pi Foundation genegen zijn.

Voor het installeren van Raspbian op het SD kaartje verwijs ik naar de installatie instructies op de website van Raspberry Pi.

Configuratie

We starten de configuratie op met :

$ sudo raspi-config

Aanpassingen :

- Expand Filesystem
- Internationalisation Options – Timezone & Keyboard
- Change User Password
- Enable Camera (!)
- Advanced Options – Hostname
- Advanced Options – Memory Split – 128 (nodig voor streaming !)
- Advanced Options – SSH aanzetten

Daarna rebooten.

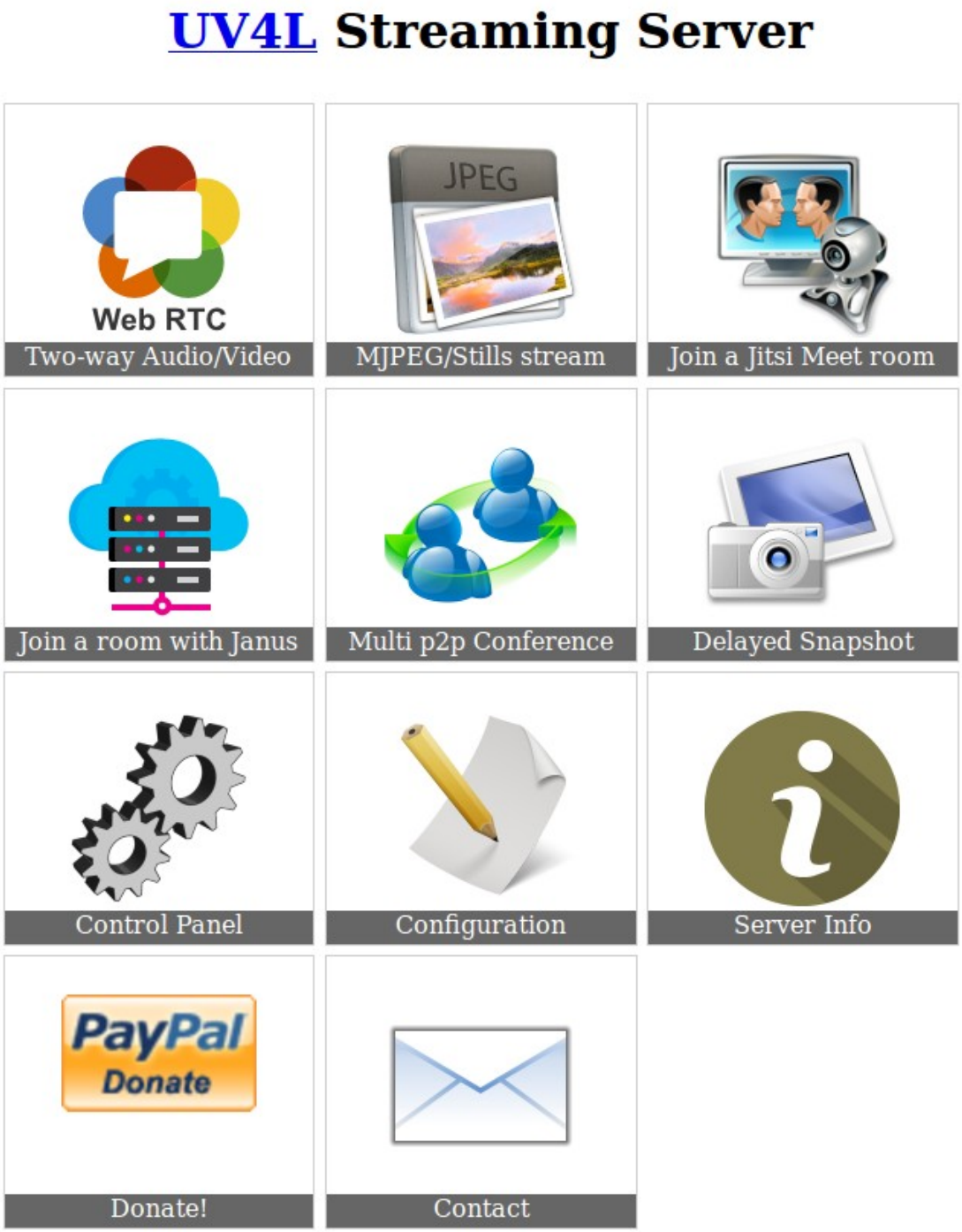
Video streaming server

UV4L

Voor dit project maak ik gebruik van UV4L die bestaat uit een aantal performante drivers voor diversen types camera (waaronder de Pi camera) en een krachtige streaming server.

Het is vooral van belang dat er tijdens het streamen niet te veel vertraging optreedt zodat de rover betrouwbaar kan bestuurd worden vanop afstand.

Voer de installatie op deze webpagina uit. Vergeet niet *uv4l-webrtc* te installeren. Surf na de installatie naar *http://<naam van je rpi>:8080*



Van hieruit kan je onder andere :

- de video stream afbeelden *(http://<naam van je rpi>:8080/stream)*
- het control panel oproepen *(http://<naam van je rpi>:8080/panel)*
- het configuratie bestand aanpassen *(http://<naam van je rpi>:8080/config)*
- een snapshot nemen *(http://<naam van je rpi>:8080/stream/snapshot.jpeg?delay_s=0)*
- de server info tonen *(http://<naam van je rpi>:8080/info)*
- ...

Configuratie

Als je bovenstaande installatie uitgevoerd hebt, dan is er een default configuratie bestand voor de Raspicam mee geïnstalleerd. We gaan de resolutie aanpassen naar 640x480 om die beter geschikt te maken voor de web interface op een default smartphone.

$ sudo nano /etc/uv4l/uv4l-raspicam.conf

We passen de *width* en *height* lijnen in dit configuratie bestand aan en slaan de wijzigingen op.

###################################
 # raspicam driver options
###################################
encoding = mjpeg
width = 640
height = 480
framerate = 30

Daarna herstarten we de service met :

$ sudo service uv4l_raspicam restart

We krijgen deze output waarin we de gebruikte driver en onze aangepaste resolutie zien :

<notice> [core] Trying driver 'raspicam' from built-in drivers...
<warning> [core] Driver 'raspicam' not found
<notice> [core] Trying driver 'raspicam' from external plug-in's...
<notice> [driver] ***Dual Raspicam Video4Linux2 Driver v1.9.35 built Mar 14 2016***
<notice> [driver] Selected format: ***640x480****, encoding: mjpeg, JPEG Video Capture*
<notice> [driver] Framerate max. 30 fps
<notice> [driver] ROI: 0, 0, 1, 1
<notice> [core] Device detected!
<notice> [core] Registering device node ***/dev/video0***

We testen tenslotte de aanpassing in onze browser :

http://<naam van je rpi>:8080/stream

Programmeertaal

Python

Python is ontwikkeld door Guido van Rossum, destijds verbonden aan het Centrum voor Wiskunde en Informatica in Amsterdam. Het heeft zijn naam te danken aan zijn favoriete televisieprogramma : Monty Python's Flying Circus.

Python is ontwikkeld met het oog op leesbare code. Hieruit vloeit haar "zuivere" stijl voort. Met weinig woorden kan men veel zeggen. Dit uit zich op verschillende manieren. Structuur wordt aangebracht door indentatie, of regelinspringing in plaats van bijvoorbeeld accolades uit C achtige talen. Statements (vergelijkbaar met zinnen uit gewone taal) worden simpelweg beëindigd door het eind van de regel. Variabelen krijgen geen typedeclaratie. Python maakt gebruik van duck-typing.

Een Python-programma wordt geschreven in een of meerdere tekstbestanden met de extensie .py die vervolgens uitgevoerd worden door de Python interpreter, die ze on the fly omzet naar uitvoerbare code.

Python wordt geleverd met een uitgebreide bibliotheek om van alles en nog wat standaard te kunnen bewerken. Het is erg eenvoudig om in Python herbruikbare code te schrijven. Doordat veel van de bibliotheken die mensen schrijven gratis aan anderen ter beschikking wordt gesteld, groeien de mogelijkheden van de bibliotheek voortdurend. Python wordt zo tot een programmeertaal die voor razendsnel ontwikkelen van een nieuwe applicatie kan worden gebruikt, zonder dat de daarbij geproduceerde code onleesbaar wordt.

Web interface

WebIOPi

Voor het maken van de web interface die communiceert met de Raspberry Pi van de rover heb ik gebruik gemaakt van WebIOPi – The Raspberry Pi Internet of Things Framework. Op de homepage vind je alle documentatie en tutorials terug om je te laten starten met dit interessante framework.

Deze software wordt op de Rpi geïnstalleerd en maakt het experimenteren met de GPIO pins een stuk handiger. Zie hierboven voor de belangrijkste onderdelen van deze software. Voor het ontwerp van de web UI zijn er een aantal voorgeprogrammeerde javascript instructies die het werk heel wat vereenvoudigen.

Installatie

Haal de laatste versie op van de website en pak deze uit in je Downloads folder.

$ cd Downloads
$ wget http://sourceforge.net/projects/webiopi/files/WebIOPi-0.7.1.tar.gz
$ tar xvzf WebIOPi-0.7.1.tar.gz

Opgelet : deze versie is enkel geschikt voor de originele Raspberry Pi met 26 header gpio pins. Voor de Raspberry Pi 2 is er een patch beschikbaar.

$ wget https://raw.githubusercontent.com/doublebind/raspi/master/webiopi-pi2bplus.patch
$ patch -p1 -i webiopi-pi2bplus.patch

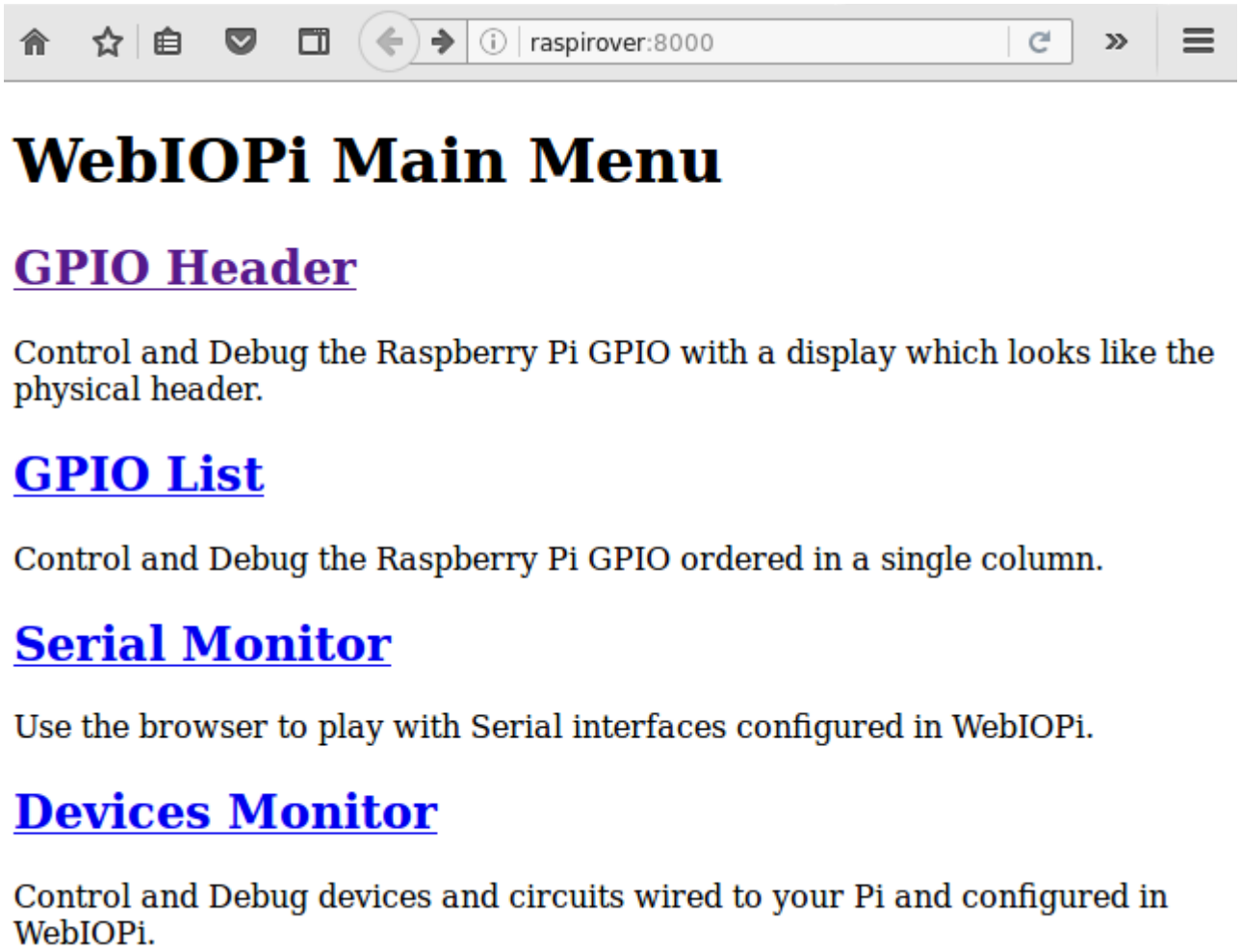
We gaan vervolgens verder met de installatie. Deze compileert de (aangepaste) source code en installeert ze.

$ sudo ./setup.sh

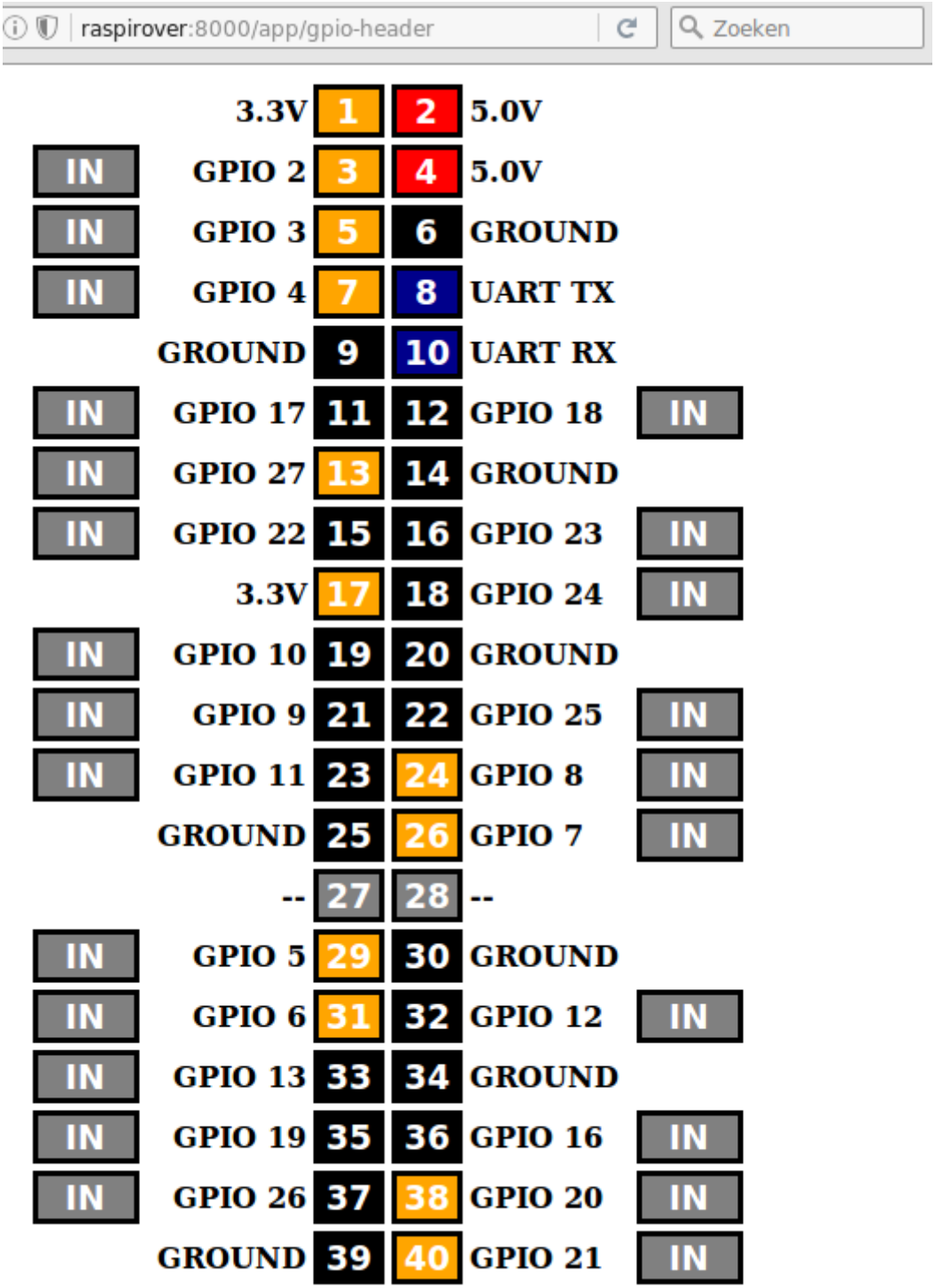
Na deze installatie testen we of alles werkt. We starten eerst de service.

$ sudo service webiopi start

Vervolgens surfen we naar *http://<naam van je Rpi>:8000*
en loggen in met de default user ***webiopi*** en paswoord ***raspberry***



We komen terecht op het hoofdmenu. Click op de link GPIO Header.



Je ziet de GPIO header pinnen. In het midden de fysieke pin nummers met daarnaast de GPIO naam en daarnaast het functie label of de pin IN (input) of OUT (output) is.

Door te klikken op dit functie label verwissel je tussen IN en OUT. Het signaal op de poort zelf zet je aan of af door op de fysieke pin nummer te klikken (oranje is aan, zwart is af)

Dit is een prima manier om interactief (zonder programma) je schakeling te testen.

Voor de veiligheid verander je best de default login en paswoord met :

$ sudo webiopi-passwd
WebIOPi passwd file generator
Enter Login: <nieuwe login>
Enter Password: <nieuw paswoord>
Confirm password: <nieuw paswoord>

Hash: e70c940a189251e9cd4515b3a1a6c6f02aa05c744a456ce360fe14bf2c5c0353
Saved to /etc/webiopi/passwd

Python

WebIOPi maakt gebruik van een eigen GPIO bibliotheek die met de software mee geïnstalleerd is. Deze is iets anders dan de WiringPi bibliotheek die we in de les gezien hebben maar komt op hetzelfde neer.

De werking van WebIOPi lijkt sterk op de Arduino programma's (sketches). Er zijn een aantal vaste functies beschikbaar :

1. *setup()* : deze functie wordt slechts éénmaal opgeroepen tijdens de starten van de WebIOPi service; ideaal dus om de gpio pinnen initialisatie te doen.
2. *loop()* : deze functie wordt voortdurend uitgevoerd als een oneindige lus zolang de WebIOPi service draait.. Zet hierin de functionaliteit van je script die steeds moet uitgevoerd worden.
3. *destroy()* : deze functie wordt aangeroepen tijdens het afsluiten van de WebIOPi service; ideaal om opruimingscode in te plaatsen.

De functies die je wil aanroepen in de web interface laat je vooraf gaan door de marker ***@webiopi.macro.***

HTML

De Javascript bibliotheek van WebIOPi bevat volgende interessante functies :

Javascript functie	**Uitleg**
webiopi()	Maak een WebIOPi instance.
ready(callback)	Voer functie callback uit bij WebIOPi klaar.
setFunction(gpio,func[,callback])	Zet gpio poort op (in|out|pwm) en roep callback op (optioneel).
digitalWrite(gpio,value[,callback])	Zet waarde (0|1) op gpio poort en roep callback op (optioneel).
digitalRead(gpio[,callback])	Lees waarde van gpio poort en roep callback op (optioneel).
toggleValue(gpio)	Wissel de waarde van de gpio poort.

Javascript functie	Uitleg
callMacro(macro [,args [,callback]])	Roep macro op van server met optioneel argumenten en callback functie.
outputSequence(gpio,period,sequence [,callback])	Output sequence met periode (ms) naar gpio poort en roep callback op (optioneel).
pulse(gpio [,callback])	Output een puls naar gpio poort en roep callback op (optioneel).
pulseRatio(gpio, ratio [,callback])	Output PWM duty cycle ratio naar gpio poort en roep callback op (optioneel)
pulseAngle(gpio, angle [,callback])	Output PWM angle naar gpio poort en roep callback op (optioneel).
createButton(id, label [,mousedown [,mouseup]])	Maak een gewone knop. Je kan functies voor mousedown en mouseup event optioneel meegeven.
createFunctionButton(gpio)	Maak een knop op de functie van de gpio poort te wisselen.
createGPIOButton(gpio,label)	Maak een knop die bij iedere klik de waarde van de gpio poort wisselt.
createMacroButton(id, label, marcro, args)	Maak een knop die een macro oproept op de server met argumenten.
createSequenceButton(id, label, gpio, period, sequence)	Maak een knop die sequence met periode (ms) stuurt naar gpio poort.
createRatioSlider(gpio, ratio)	Maak een slider die een PWM duty cucle ratio stuurt naar de gpio poort.
createAngleSlider(gpio, angel)	Maal een slider die een PWM angle stuurt naar de gpio poort (servomotoren).
setLabel(id, label)	Verandert het label van de knop.

Voor het maken van de rover web interface heb ik vooral gebruik gemaakt van de createMacroButton functie waarmee functies worden opgeroepen in mijn Python script.

Configuratie

Om WebIOPi te configureren om gebruik te maken van onze eigen webpagina en code voeren we het volgende uit :

$ sudo nano /etc/webiopi/config

In de sectie [SCRIPTS] voegen we een lijn toe die verwijst naar ons python script.

```
[SCRIPTS]
# Load custom scripts syntax :
# name = sourcefile
#   each sourcefile may have setup, loop and destroy functions and macros
#myscript = /home/pi/webiopi/examples/scripts/macros/script.py
myproject = /home/pi/effevees_rover/python/effevees_rover.py
```

In de sectie [HTTP] voegen we een lijn toe die verwijst naar de plaats van onze html pagina. Standaard wordt index.html afgebeeld.

```
[HTTP]
# HTTP Server configuration
enabled = true
port = 8000

# File containing sha256(base64("user:password"))
# Use webiopi-passwd command to generate it
passwd-file = /etc/webiopi/passwd

# Change login prompt message
prompt = "WebIOPi"

# Use doc-root to change default HTML and resource files location
#doc-root = /home/pi/webiopi/examples/scripts/macros
doc-root = /home/pi/effevees_rover/templates

# Use welcome-file to change the default "Welcome" file
#welcome-file = index.html
```

In de sectie [REST] wijzigen we enkele lijnen die de web interface beperken tot de gebruikte gpio pinnen.

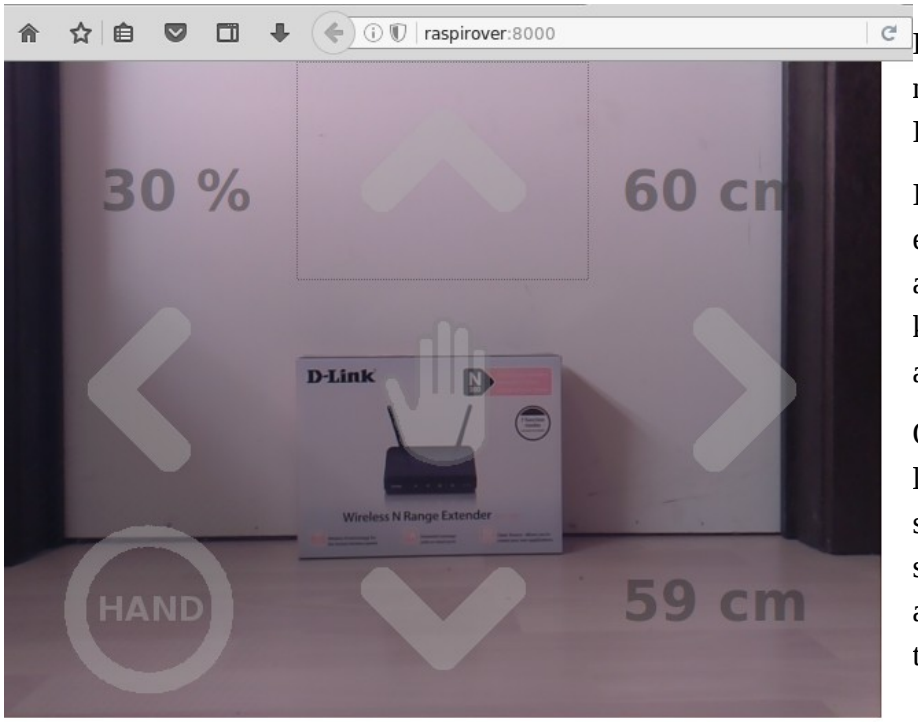
```
[REST]
# By default, REST API allows to GET/POST on all GPIOs
# Use gpio-export to limit GPIO available through REST API
gpio-export = 17,18,22,23,24,25,27,4,20,21

# Uncomment to forbid changing GPIO values
gpio-post-value = true

# Uncomment to forbid changing GPIO functions
gpio-post-function = false

# Uncomment to disable automatic device mapping
#device-mapping = false
```

Tenslotte zorgen we er nog voor dat de WebIOPi service automatisch start bij het booten van de Rpi.

$ sudo update-rc.d webiopi defaults

Rover interface

Handbediening

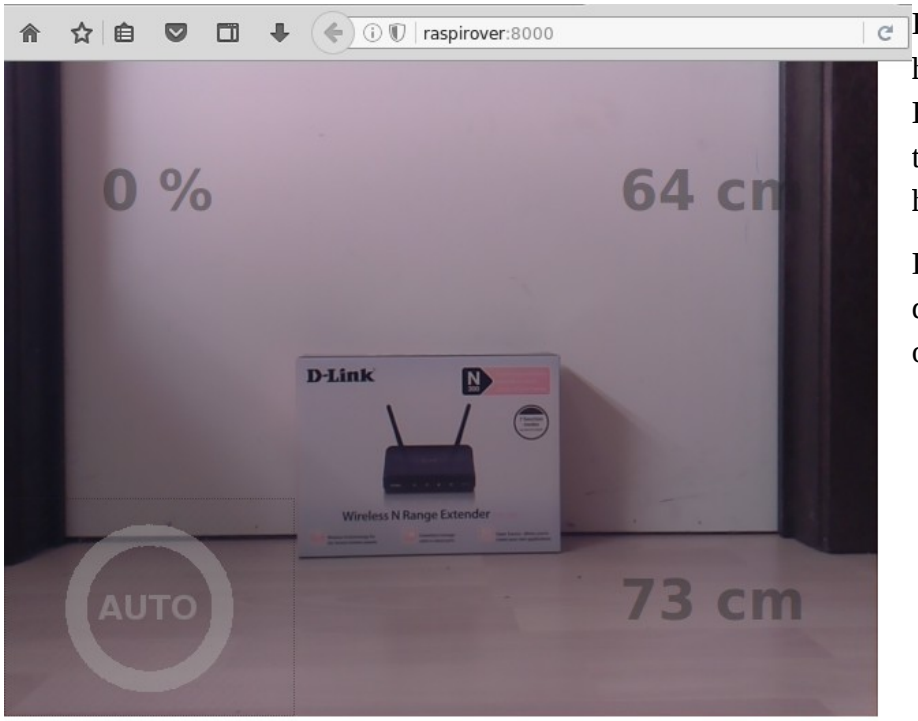


De interface is opgebouwd met de videostream van de Pi camera als achtergrond.

Ik heb die ruimte verdeeld in een 3x3 grid. Op de centrale assen staan de bedienings-knoppen : vooruit, stoppen, achteruit, links en rechts.

Op de hoeken vind je vanaf linksboven in wijzerzin : snelheid (%), afstand voorste sensor (cm), afstand achterste sensor (cm) en de toggle knop hand / auto.

Autonavigatie

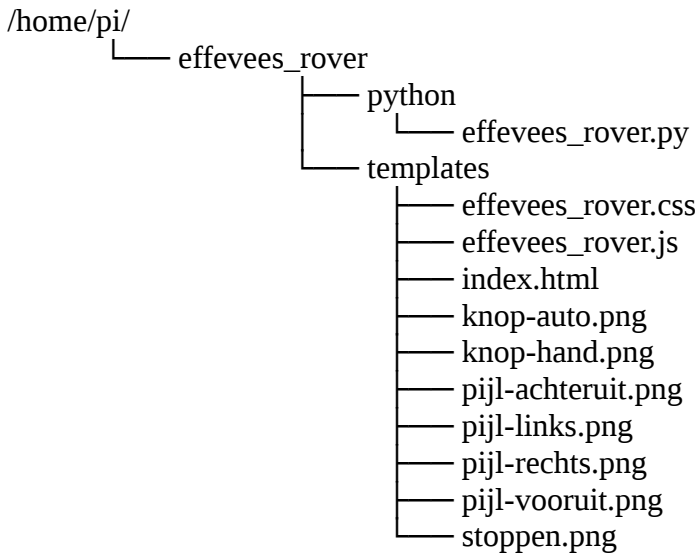


Hier zijn de knoppen voor de handbediening uitgeschakeld Enkel de toggle knop om terug over te gaan naar handbediening is zichtbaar.

De snelheid en afstanden van de sensoren worden uiteraard ook nog afgebeeld.

Source code

Voor dit project werd de volgende folder structuur opgezet :



index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="height=480px,width=640px user-scalable=yes">
<title>Effevee's Rover</title>
    <script type="text/javascript" src="/webiopi.js"></script>
    <script type="text/javascript" src="/effevees_rover.js"></script>
    <link rel="stylesheet" type="text/css" href="/effevees_rover.css">
</head>

<body>
<div id="controls">
    <img src="http://raspirover.local:8080/stream/video.mjpeg" width="640"
height="480">
    <input type="text" id="snelheid"></br>
    <input type="text" id="afstandvoor"></br>
    <input type="text" id="afstandachter"></br>
</div>
</body>
</html>
```

effevees_rover.js

```
// dit wordt uitgevoerd bij de initialisatie van WebIOPi
webiopi().ready(function() {

    // Maak de macro knop vooruit
    // bij het klikken wordt de python code Vooruit aangeroepen
    var knop_vooruit = webiopi().createMacroButton("idBtnVooruit", "",
"Vooruit");

    // Maak de macro knop stoppen
    // bij het klikken wordt de python code Stoppen aangeroepen
    var knop_stoppen = webiopi().createMacroButton("idBtnStoppen", "",
"Stoppen");

    // Maak de macro knop achteruit
    // bij het klikken wordt de python code Achteruit aangeroepen
    var knop_achteruit = webiopi().createMacroButton("idBtnAchteruit", "",
"Achteruit");

    // Maak de macro knop links
    // bij het klikken wordt de python code Links aangeroepen
    var knop_links = webiopi().createMacroButton("idBtnLinks", "", "Links");

    // Maak de macro knop rechts
    // bij het klikken wordt de python code Rechts aangeroepen
    var knop_rechts = webiopi().createMacroButton("idBtnRechts", "", "Rechts");

    // Maak de knop hand/auto
    // bij het klikken wordt de python code ToggleModus aangeroepen
    // de callback functie updateModus update de class van de knop
    var knop_modus = webiopi().createButton("idBtnModus", "", function() {
        webiopi().callMacro("ToggleModus", [], updateModus);
    });

    // Voeg de knoppen toe aan het HTML element met ID="controls" mbv jQuery
    $("#controls").append(knop_vooruit);
    $("#controls").append(knop_stoppen);
    $("#controls").append(knop_achteruit);
    $("#controls").append(knop_links);
    $("#controls").append(knop_rechts);
    $("#controls").append(knop_modus);

    // Start een timer om iedere seconde de UI te updaten
    setInterval(LeesWaarden,1000);

});

// deze functie roept de python code WaardenUI aan om de waarden voor de UI op
te halen
// de callback functie updateUI zorgt voor de eigenlijke update van de UI
function LeesWaarden(){
   webiopi().callMacro("WaardenUI", [], updateUI);
}

// deze functie wordt opgeroepen als callback van LeesWaarden (timer)
// de returnwaarden worden gebruikt om de UI te updaten met de nieuwe waarden
```

```
var updateUI = function(macro, args, response) {
    // returnwaarden zijn als tekst doorgegeven gescheiden door ";"
    var waarden = response.split(";");
    $("#snelheid").val(waarden[0]+" %");
    $("#afstandvoor").val(waarden[1]+" cm");
    $("#afstandachter").val(waarden[2]+" cm");
    $("#automodus").val(waarden[3]);
};

// deze functie wordt opgeroepen bij de toggle van hand naar auto of omgekeerd
// de returnwaarde wordt gebruikt om de class van de besturingsknoppen aan te
passen
// waardoor we via css de juiste knop kunnen afbeelden in de UI
var updateModus = function(macro, args, response) {
    var waarden = response.split(";");
    // update class voor knoppen
    if (waarden[3] == "True") {
        document.getElementById("idBtnVooruit").className = "auto";
        document.getElementById("idBtnStoppen").className = "auto";
        document.getElementById("idBtnAchteruit").className = "auto";
        document.getElementById("idBtnLinks").className = "auto";
        document.getElementById("idBtnRechts").className = "auto";
        document.getElementById("idBtnModus").className = "auto";
    } else {
        document.getElementById("idBtnVooruit").className = "hand";
        document.getElementById("idBtnStoppen").className = "hand";
        document.getElementById("idBtnAchteruit").className = "hand";
        document.getElementById("idBtnLinks").className = "hand";
        document.getElementById("idBtnRechts").className = "hand";
        document.getElementById("idBtnModus").className = "hand";
    }
}
```

effevees_rover.css

```
/* controle paneel */
#controls {
    position:absolute;
    top:0px;
    left:0px;
    width:640px;
    height:480px;
    border:0px solid #000000;
}

/* knop vooruit */
#idBtnVooruit {
    position:absolute;
    top:0px;
    left:214px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(pijl-vooruit.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* knop stoppen */
#idBtnStoppen {
    position:absolute;
    top:160px;
    left:214px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(stoppen.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* knop achteruit */
#idBtnAchteruit {
    position:absolute;
    top:320px;
    left:214px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(pijl-achteruit.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* knop links */
#idBtnLinks {
    position:absolute;
```

```
    top:160px;
    left:0px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(pijl-links.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* knop rechts */
#idBtnRechts {
    position:absolute;
    top:160px;
    left:426px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(pijl-rechts.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* toggle modus handbediening */
#idBtnModus.Default,
#idBtnModus.hand {
    position:absolute;
    top:320px;
    left:0px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(knop-hand.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* toggle modus auto navigatie */
#idBtnModus.auto {
    position:absolute;
    top:320px;
    left:0px;
    width:213px;
    height:160px;
    opacity:0.4;
    z-index: 20;
    background: url(knop-auto.png) no-repeat;
    background-position: center center;
    border:0px solid #000000;
}

/* knoppen minder transparant maken bij hover */
#idBtnVooruit:hover,
#idBtnStoppen:hover,
#idBtnAchteruit:hover,
```

```
#idBtnLinks:hover,
#idBtnRechts:hover,
#idBtnModus:hover {
    opacity:0.8;
}

/* knoppen verbergen bij auto navigatie */
#idBtnVooruit.auto,
#idBtnStoppen.auto,
#idBtnAchteruit.auto,
#idBtnLinks.auto,
#idBtnRechts.auto {
    visibility: hidden;
}

/* text input boxen voor afbeelden snelheid en afstanden */
input[type=text] {
    background: transparent;
    border: none;
    width: 150px;
    font-size: 40px;
    font-weight: bold;
    font-color: #afaeae;
}

/* snelheid */
#snelheid {
    position: absolute;
    top:70px;
    left:70px;
    opacity:0.4;
    z-index: 20;
}

/* afstand voor */
#afstandvoor {
    position: absolute;
    top:70px;
    left:450px;
    opacity:0.4;
    z-index: 20;
}

/* afstand achter */
#afstandachter {
    position: absolute;
    top:370px;
    left:450px;
    opacity:0.4;
    z-index: 20;
}
```

effevees_rover.py

```
#!/usr/bin/python

## Frank Vergote
## email effevee@gmail.com
## WebIOPi rover versie 5
## 06/05/2016

################################################################################
# Bibliotheken
################################################################################

import webiopi
import time
import threading
import logging
import random

# gebruik de gpio lib van WebIOPi voor de interactie met de gpio pinnen
gpio = webiopi.GPIO

################################################################################
# Constanten en variabelen
################################################################################

# GPIO poorten voor sturen motoren via L298N motor module
# -------------------------------------------------------
# Motor             LINKS               RECHTS
# GPIO          IN1     IN2         IN3     IN4
# -------------------------------------------------------
# Vooruit       low     high        low     high
# Stoppen       low     low         low     low
# Achteruit     high    low         high    low
# -------------------------------------------------------
L298N_IN1 = 17          # richting motoren links
L298N_IN2 = 18          # richting motoren links
L298N_ENA = 24          # snelheid motoren links (PWM)

L298N_IN3 = 22          # richting motoren rechts
L298N_IN4 = 23          # richting motoren rechts
L298N_ENB = 25          # snelheid motoren rechts (PWM)

# commando's motoren
VOORUIT = 1
STOPPEN = 2
ACHTERUIT = 3
LINKS = 4
RECHTS = 5

# snelheden motoren
minSnelheid = 0         # alle snelheden in procenten
maxSnelheid = 100
stapSnelheid = 10
snelheid = minSnelheid
Commando = STOPPEN
laatsteCommando = STOPPEN
```

```
# modus besturing rover
autoModus = False       # auto navigatie boolean
autoSnelheid = 30       # snelheid in % bij auto navigatie
kwartDraai = 2          # tijd in sec voor een kwartdraai (90°)

# GPIO poorten afstandssensoren HC-SR04 voor en achter
ASV_TRIG = 27           # trigger afstandssensor voor
ASV_ECHO = 4            # echo afstandssensor voor

ASA_TRIG = 20           # trigger afstandssensor achter
ASA_ECHO = 21           # echo afstandssensor achter

ASV = 1                 # afstandssensor voor
ASA = 2                 # afstandssensor achter

# berekenen afstanden
PULSE = 0.00001         # trigger pulse 10 micro sec
SNELHEID_GELUID = 34000 # geluidsnelheid in cm/sec

# afstanden
minAfstand = 50         # minimum afstand tot obstakel in cm
maxAfstand = 200        # maximum afstand tot obstakel in cm

afstandVoor = 200       # variabele afstand voorste sensor
afstandAchter = 200     # variabele afstand achterste sensor


##############################################################################
# Setup wordt 1 maal uitgevoerd tijdens het starten van de WebIOPi service
# hier doen we de initialisatie van de GPIO pinnen
##############################################################################

def setup():

    # Zet debug aan
    #webiopi.setDebug()

    # setup GPIO poorten L298N
    gpio.setFunction(L298N_IN1,gpio.OUT)
    gpio.setFunction(L298N_IN2,gpio.OUT)
    gpio.setFunction(L298N_ENA,gpio.PWM)
    gpio.setFunction(L298N_IN3,gpio.OUT)
    gpio.setFunction(L298N_IN4,gpio.OUT)
    gpio.setFunction(L298N_ENB,gpio.PWM)

    # setup GPIO poorten HC-SR04
    gpio.setFunction(ASV_TRIG,gpio.OUT)
    gpio.setFunction(ASV_ECHO,gpio.IN)
    gpio.setFunction(ASA_TRIG,gpio.OUT)
    gpio.setFunction(ASA_ECHO,gpio.IN)

    # init GPIO poorten L298N
    gpio.digitalWrite(L298N_IN1,gpio.LOW)
    gpio.digitalWrite(L298N_IN2,gpio.LOW)
    gpio.pwmWrite(L298N_ENA,snelheid/100)
    gpio.digitalWrite(L298N_IN3,gpio.LOW)
    gpio.digitalWrite(L298N_IN4,gpio.LOW)
```

```
    gpio.pwmWrite(L298N_ENB,snelheid/100)

    # init GPIO poorten HC-SR04
    gpio.digitalWrite(ASV_TRIG,gpio.LOW)
    gpio.digitalWrite(ASA_TRIG,gpio.LOW)

    # thread opzetten voor afstandsmetingen en botsingsdetectie
    thread_metingen = threading.Thread(target = MeetAfstanden)
    thread_metingen.setDaemon(True)
    thread_metingen.start()

    # thread opzetten voor auto navigatie
    thread_navigatie = threading.Thread(target = Navigeer)
    thread_navigatie.setDaemon(True)
    thread_navigatie.start()


################################################################################
# Loop wordt voortdurend herhaald zolang de WebIOPi service draait
# ik gebruik dit niet omdat loop enkel een single thread ondersteunt.
################################################################################

# def loop():


################################################################################
# Destroy wordt uitgevoerd bij het stoppen van de WebIOPi service
# dit wordt gebruikt voor de cleanup van de GPIO poorten.
################################################################################

def destroy():

    # reset GPIO poorten
    gpio.digitalWrite(L298N_IN1,gpio.LOW)
    gpio.digitalWrite(L298N_IN2,gpio.LOW)
    gpio.digitalWrite(L298N_IN3,gpio.LOW)
    gpio.digitalWrite(L298N_IN4,gpio.LOW)
    gpio.digitalWrite(ASV_TRIG,gpio.LOW)
    gpio.digitalWrite(ASA_TRIG,gpio.LOW)

    # setup GPIO poorten
    gpio.setFunction(L298N_IN1,gpio.IN)
    gpio.setFunction(L298N_IN2,gpio.IN)
    gpio.disablePWM(L298N_ENA)
    gpio.setFunction(L298N_IN3,gpio.IN)
    gpio.setFunction(L298N_IN4,gpio.IN)
    gpio.disablePWM(L298N_ENB)
    gpio.setFunction(ASV_TRIG,gpio.IN)
    gpio.setFunction(ASV_ECHO,gpio.IN)
    gpio.setFunction(ASA_TRIG,gpio.IN)
    gpio.setFunction(ASA_ECHO,gpio.IN)



################################################################################
# ZetSnelheid bepaalt de snelheid van de rover :
# manuele modus : verhoogt de huidige snelheid met stapSnelheid tot maxSnelheid
#                 bij veranderen van richting reset de snelheid
```

```
# auto modus : gebruikt steeds een vaste snelheid -> autoSnelheid
################################################################################

def ZetSnelheid(Commando):
    global snelheid, laatsteCommando

    # snelheid op 0 zetten bij wijziging richting
    if Commando != laatsteCommando:
        snelheid = 0
        gpio.pwmWrite(L298N_ENA,snelheid/100)
        gpio.pwmWrite(L298N_ENB,snelheid/100)
        laatsteCommando = Commando

    # instellen snelheid
    if autoModus:
        # auto modus -> autoSnelheid
        snelheid = autoSnelheid
    else:
        # manuele modus -> snelheid verhogen met stapSnelheid
        snelheid += stapSnelheid

        # begrenzen tot maxSnelheid
        if snelheid > maxSnelheid:
            snelheid = maxSnelheid

    return snelheid


################################################################################
# Afstand bepaalt de afstand (cm) tot een obstakel met de HC-SR04 sensoren
# de tijd dat de echo puls hoog is evenredig met 2x de afstand tot het obstakel
################################################################################

def Afstand(sensor):

    # sensor variabelen
    if sensor == ASV:
        TRIG = ASV_TRIG
        ECHO = ASV_ECHO
    elif sensor == ASA:
        TRIG = ASA_TRIG
        ECHO = ASA_ECHO
    else:
        # ongeldige sensor
        print('Ongeldige sensor!')
        return maxAfstand

    # initialisatie
    afstand = 0

    # als echo nu reeds hoog is, dan ongeldige meting
    if gpio.digitalRead(ECHO):
        return maxAfstand

    # zet trigger laag en wacht 50 msec om sensor te stabiliseren
    gpio.digitalWrite(TRIG, gpio.LOW)
    time.sleep(0.05)
```

```
    # zet trigger hoog gedurende 10 micro sec
    gpio.digitalWrite(TRIG,gpio.HIGH)
    time.sleep(PULSE)
    gpio.digitalWrite(TRIG,gpio.LOW)

    # initialisatie tijden
    start, stop = time.time(), time.time()

    # wachten op echo hoog voor maximum 20 msec
    # indien langer, dan afbreken met ongeldige meting
    while not gpio.digitalRead(ECHO):
        start = time.time()
        if start - stop > 0.02:
            return maxAfstand

    # wachten op echo laag voor maximum 20 msec
    # indien langer, dan afbreken met ongeldige meting
    while gpio.digitalRead(ECHO):
        stop = time.time()
        if stop - start > 0.02:
            return maxAfstand

    # berekenen afstand
    delta = stop - start
    afstand = delta * SNELHEID_GELUID / 2.0

    return round(afstand, 0)


##############################################################################
# MeetAfstanden draait in een aparte thread. De afstanden van beide sensoren
# worden iedere 250 msec gemeten en bewaard in globale variabelen.
# In manuele stuurmodus wordt de rover gestopt bij risico op botsing.
##############################################################################
def MeetAfstanden():
    global afstandVoor, afstandAchter

    while True:

        # initialiseer de lijsten voor de afstandsmetingen
        av, aa = [], []

        # meet 5 afstanden voor iedere sensor en voeg ze toe in de lijsten
        for i in range(5):
            av.append(Afstand(ASV))
            aa.append(Afstand(ASA))

        # verwijder de minimum en maximum meetwaarden uit de lijsten
        av.remove(min(av))
        av.remove(max(av))
        aa.remove(min(aa))
        aa.remove(max(aa))

        # bereken het gemiddelde van de restende meetwaarden uit de lijsten
        afstandVoor = round(sum(av)/len(av),0)
        afstandAchter = round(sum(aa)/len(aa),0)

        # debug
```

```
        #logging.debug("Richting: %s",laatsteCommando)
        #logging.debug("Afstand voor: %d cm", afstandVoor)
        #logging.debug("Afstand achter: %d cm", afstandAchter)

        # stoppen bij risico op botsing in manuele stuurmodus
        if not autoModus:
            if (laatsteCommando == VOORUIT) and (afstandVoor < minAfstand):
                Stoppen()
            if (laatsteCommando == ACHTERUIT) and (afstandAchter < minAfstand):
                Stoppen()

        # wacht even
        time.sleep(0.25)


##################################################################################
# WaardenUI wordt periodiek aangeroepen vanuit javascript (setInterval).
# De waarden dienen om de UI up te daten.
##################################################################################

@webiopi.macro
def WaardenUI():

    return "%d;%d;%d;%s" % (snelheid, afstandVoor, afstandAchter, autoModus)


##################################################################################
# ToggleModus verandert de status van de besturingsmodus.
# wordt aangeroepen bij het klikken op de modus knop van de UI
##################################################################################

@webiopi.macro
def ToggleModus():
    global autoModus

    # rover stoppen
    Stoppen()

    # modus besturing veranderen
    autoModus = not autoModus

    return WaardenUI()


##################################################################################
# Navigeer zorgt ervoor dat de rover automatisch zijn weg zoekt.
# Dit gebeurt in een aparte thread. We moeten namelijk terug kunnen
# overschakelen naar manuele modus via de UI
##################################################################################

def Navigeer():
    global snelheid

    while True:

        # enkel uitvoeren bij automatische navigatie
        while autoModus:
```

```
            # rover vooruit
            Vooruit()

            # botsing gevaar
            while afstandVoor < minAfstand:

                # rover stoppen
                Stoppen()

                # rover 2 sec achteruit
                if (afstandAchter > minAfstand):
                    Achteruit()
                    time.sleep(2)
                    Stoppen()

                # willekeurige kwartdraai
                if random.randint(1,10) <= 5:
                    Links()
                else:
                    Rechts()

            # even pauzeren
            time.sleep(0.5)


################################################################################
# Rover vooruit
# wordt aangeroepen bij het klikken op de vooruit knop van de UI
################################################################################

@webiopi.macro
def Vooruit():

    # snelheid instellen
    ZetSnelheid(VOORUIT)

    # linkermotoren vooruit
    gpio.digitalWrite(L298N_IN1,gpio.LOW)
    gpio.digitalWrite(L298N_IN2,gpio.HIGH)

    # rechtermotoren vooruit
    gpio.digitalWrite(L298N_IN3,gpio.LOW)
    gpio.digitalWrite(L298N_IN4,gpio.HIGH)

    # snelheid linker- en rechtermotoren
    gpio.pwmWrite(L298N_ENA,snelheid/100)
    gpio.pwmWrite(L298N_ENB,snelheid/100)


################################################################################
# Rover stoppen
# wordt aangeroepen bij het klikken op de stoppen knop van de UI
################################################################################

@webiopi.macro
def Stoppen():
    global snelheid
```

```
    # snelheid op 0 zetten
    snelheid = 0

    # linkermotoren stoppen
    gpio.digitalWrite(L298N_IN1,gpio.LOW)
    gpio.digitalWrite(L298N_IN2,gpio.LOW)

    # rechtermotoren stoppen
    gpio.digitalWrite(L298N_IN3,gpio.LOW)
    gpio.digitalWrite(L298N_IN4,gpio.LOW)

    # snelheid linker- en rechtermotoren
    gpio.pwmWrite(L298N_ENA,snelheid)
    gpio.pwmWrite(L298N_ENB,snelheid)


##################################################################################
# Rover achteruit
# wordt aangeroepen bij het klikken op de achteruit knop van de UI
##################################################################################

@webiopi.macro
def Achteruit():

    # snelheid instellen
    ZetSnelheid(ACHTERUIT)

    # linkermotoren achteruit
    gpio.digitalWrite(L298N_IN1,gpio.HIGH)
    gpio.digitalWrite(L298N_IN2,gpio.LOW)

    # rechtermotoren achteruit
    gpio.digitalWrite(L298N_IN3,gpio.HIGH)
    gpio.digitalWrite(L298N_IN4,gpio.LOW)

    # snelheid linker- en rechtermotoren
    gpio.pwmWrite(L298N_ENA,snelheid/100)
    gpio.pwmWrite(L298N_ENB,snelheid/100)


##################################################################################
# Rover links
# wordt aangeroepen bij het klikken op de links knop van de UI
# voert een kwartdraai (90°) uit; resultaat is afhankelijk van ondergrond
##################################################################################

@webiopi.macro
def Links():
    global snelheid

    # snelheid instellen
    ZetSnelheid(LINKS)

    # kwartdraai uitvoeren aan snelheid 40%
    snelheid = 40

    # linkermotoren achteruit
    gpio.digitalWrite(L298N_IN1,gpio.HIGH)
```

```
    gpio.digitalWrite(L298N_IN2,gpio.LOW)

    # rechtermotoren vooruit
    gpio.digitalWrite(L298N_IN3,gpio.LOW)
    gpio.digitalWrite(L298N_IN4,gpio.HIGH)

    # snelheid linker- en rechtermotoren
    gpio.pwmWrite(L298N_ENA,snelheid/100)
    gpio.pwmWrite(L298N_ENB,snelheid/100)

    # tijd voor kwartdraai en daarna stoppen
    time.sleep(kwartDraai)
    Stoppen()


##############################################################################
# Rover rechts
# wordt aangeroepen bij het klikken op de rechts knop van de UI
# voert een kwartdraai (90°) uit; resultaat is afhankelijk van ondergrond
##############################################################################

@webiopi.macro
def Rechts():
    global snelheid

    # snelheid instellen
    ZetSnelheid(RECHTS)

    # kwartdraai uitvoeren aan snelheid 40%
    snelheid = 40

    # linkermotoren vooruit
    gpio.digitalWrite(L298N_IN1,gpio.LOW)
    gpio.digitalWrite(L298N_IN2,gpio.HIGH)

    # rechtermotoren achteruit
    gpio.digitalWrite(L298N_IN3,gpio.HIGH)
    gpio.digitalWrite(L298N_IN4,gpio.LOW)

    # snelheid linker- en rechtermotoren
    gpio.pwmWrite(L298N_ENA,snelheid/100)
    gpio.pwmWrite(L298N_ENB,snelheid/100)

    # tijd voor kwartdraai en daarna stoppen
    time.sleep(kwartDraai)
    Stoppen()
```

Afbeeldingen

pijl-vooruit.png

pijl-achteruit.png

stoppen.png

pijl-links.png

pijl-rechts.png

knop-hand.png

knop-auto.png