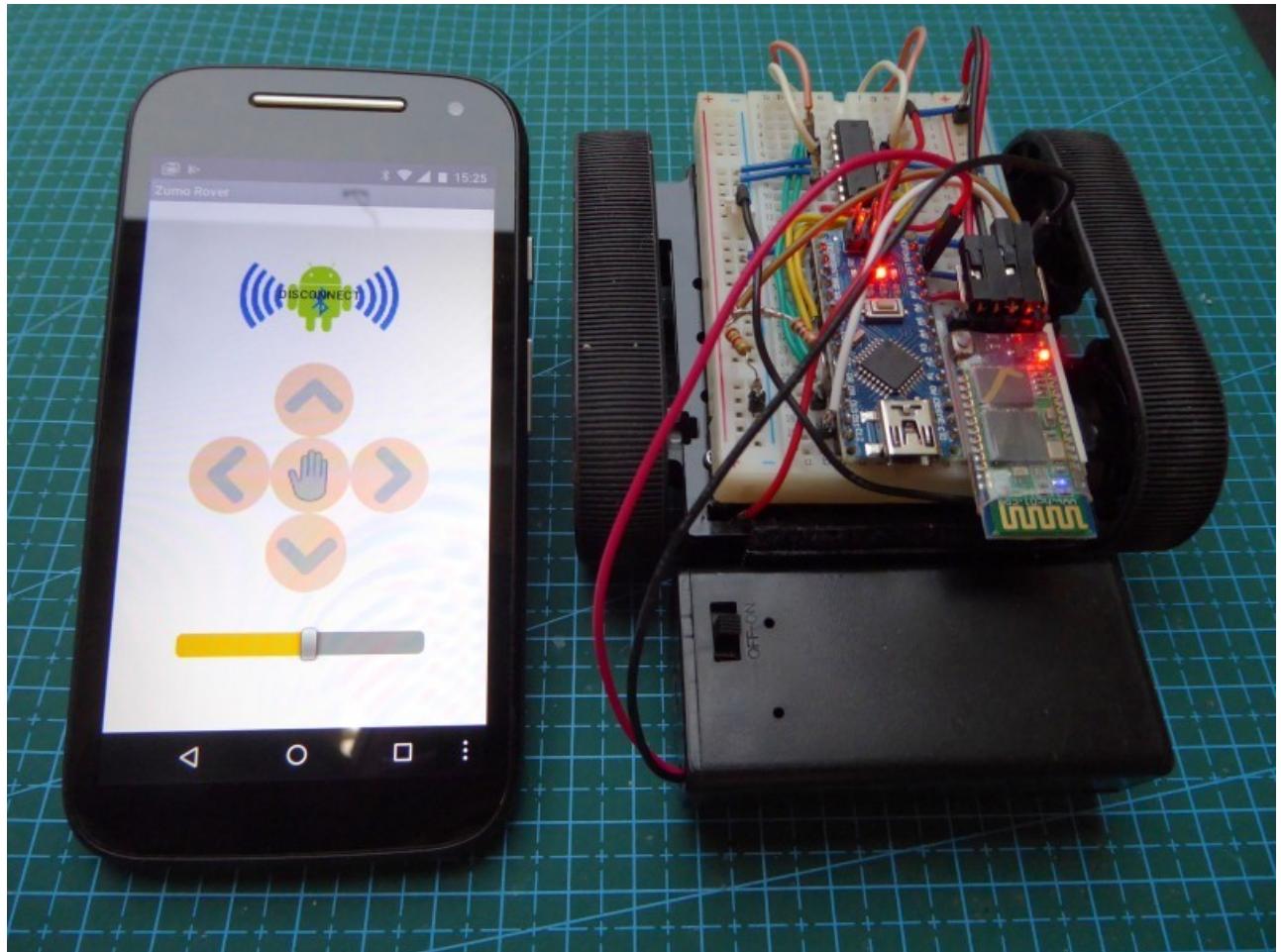


Zumo Rover



Dit document en de bijhorende source code kan je [hier](#) downloaden.

Enjoy ! Effevee :-)

Inhoudsopgave

Beschrijving.....	3
Hardware.....	3
Overzicht.....	3
Robotvoertuig.....	3
Microcontroller.....	3
Allerlei.....	3

Gereedschap.....	3
Pololu Zumo.....	4
Specificaties.....	4
Motoren.....	4
Arduino Nano.....	4
Specificaties.....	4
Pin layout.....	6
HC-05 Bluetooth module.....	7
Bluetooth.....	7
Specificaties.....	7
Pin layout.....	7
L293DNE dubbele H-brug motor controller.....	8
H-brug.....	8
Specificaties.....	8
Pin layout.....	8
9V PP3 batterijhouder.....	9
Zumo Rover besturing.....	10
Fritzing schema.....	10
Schakeling.....	11
Zumo Rover assemblage.....	12
Zumo chassis.....	12
Batterij contacten.....	12
Motoren.....	13
Bovenplaat.....	13
Wielen.....	14
Rupsbanden.....	14
Batterijen.....	15
Batterijdeksel.....	15
Rover niet geassembleerd.....	16
Rover geassembleerd.....	16
Software.....	18
MIT App Inventor code.....	18
Designer.....	18
Blocks Bluetooth.....	19
Blocks Besturing.....	20
Arduino sketch.....	21
Bibliotheken.....	21
Constanten.....	21
Variabelen.....	22
Setup.....	22
Loop.....	23
Functies.....	24

Beschrijving

Als eindproject voor deze cursus heb ik gekozen voor een robotvoertuig op basis van een Pololu Zumo chassis dat met een Arduino Nano via Bluetooth bestuurd wordt. De commando's worden vanaf een Android smartphone of tablet doorgestuurd via een app die ik met [MIT App Inventor](#) heb gemaakt. Het chassis worden aangedreven met een L293DNE dubbele H-brug om de 2 motoren onafhankelijk van elkaar voor- of achteruit te laten draaien. Hierdoor kan het robotvoertuig zowel vooruit en achteruit rijden, naar links of rechts draaien. De snelheid is variabel in te stellen via een slider in de app.

Hardware

Overzicht

Robotvoertuig

- 1x [Pololu Zumo chassis](#)
- 2x [Micro Metal Gearmotor 50:1](#)
- 2x [ceramische condensator 0,1µF \(104\)](#)
- 4x NiMH AA/HR6 oplaadbare batterijen 1.2V

Microcontroller

- 1x [Arduino Nano](#)
- 1x [HC-05 ZS-040 Bluetooth module](#)
- 1x [L293DNE dubbele H-brug motor controller](#)
- 1x [9V PP3 batterijhouder met schakelaar](#)
- 1x 9V PP3 batterij

Allerlei

- 1x [breadboard 400 pins](#)
- 1x [weerstand 1K en 2K](#)
- [jumperkabels](#)
- Velcro klitteband

Gereedschap

- Soldeerbout & soldeer
- Lijmpistool
- Schroevendraaier set

Pololu Zumo

Specificaties

Het Pololu Zumo chassis is een klein robot platform voorzien van rupsbanden. De afmetingen zijn ongeveer 10cm op 9cm.

Het hoofdgedeelte is samengesteld uit zwarte ABS en biedt plaats voor 4 AA batterijen en houders voor 2 Micro Metal Gear motoren. Het wordt geleverd met 2 silicone rupsbanden, 2 aandrijf- en 2 passieve tandwielen, een acryl bevestigingsplaat en montagemateriaal.



Motoren

Ik heb het chassis uitgerust met 2 micro metal gear high power motoren :

- verhouding 50:1
- maximum toerental : 625 rpm (6V)
- maximum snelheid : 130 cm/s (6V)
- torsie : 1,5 kg/cm
- maximum stroom : 1,6 A



www.pololu.com

Arduino Nano

Specificaties

Wegens de kleine afmetingen van het rupsvoertuig is voor de besturing gekozen voor een Arduino Nano.

De [Arduino Nano V3](#) is heel gelijkaardig aan de UNO maar heeft een veel kleinere afmeting. Hij kan bovendien makkelijk op een breadboard geprikt worden. Een half breadboard met 400 pins past trouwens perfect op het zumo chassis.

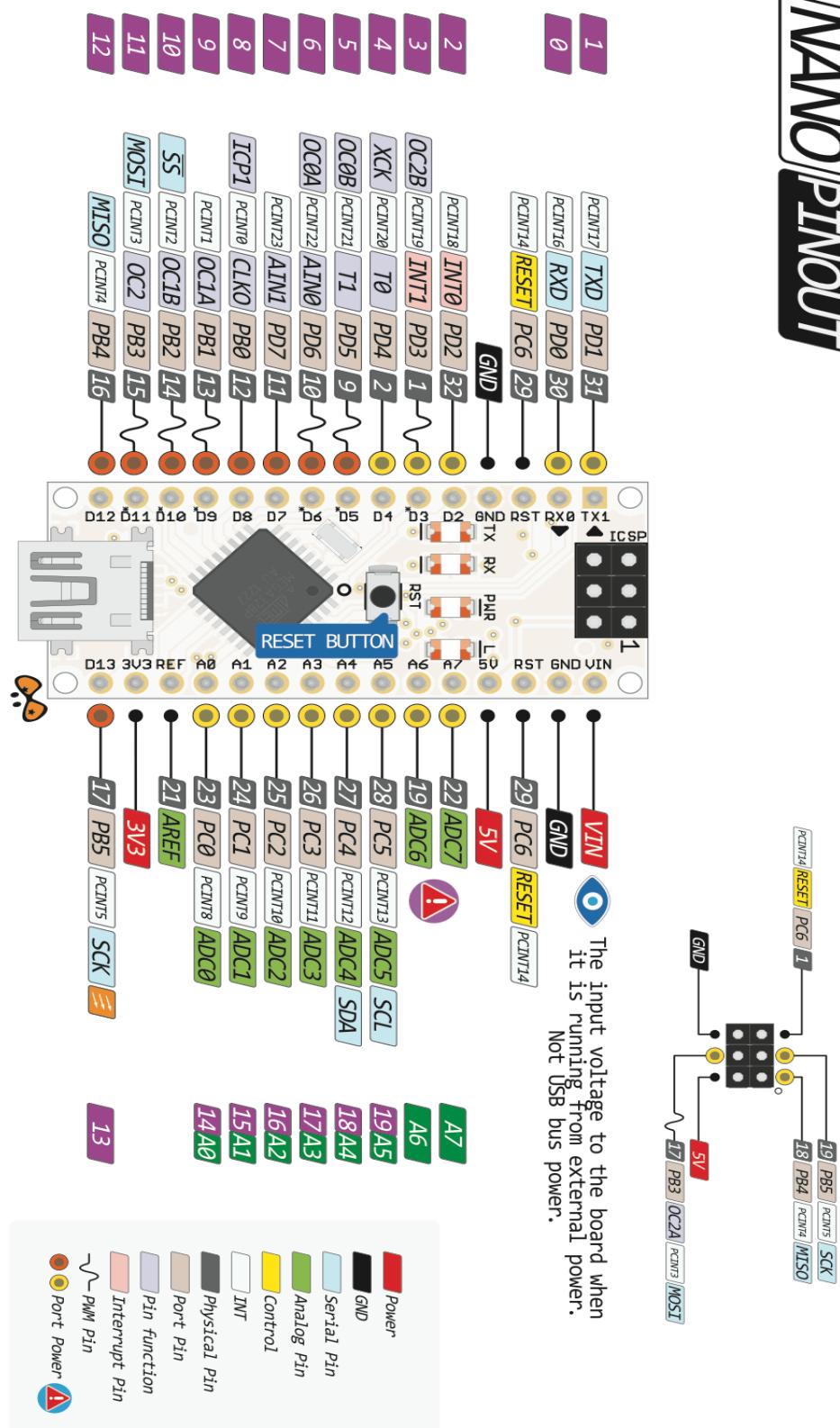


De volgende tabel vergelijkt de kenmerken van de Nano V3 met de UNO R3 :

Specificatie	Nano V3	Uno R3
Processor	ATmega328P	ATmega328P
Spanning pinnen	5 V	5 V
Voedingsspanning (VIN)	7-9 V	7-12 V
Kloksnelheid	16 MHz	16 MHz
Analoge poorten In/Out	8/0	6/0
Digitale poorten IO/PWM	14/6	14/6
EEPROM	1 KB	1 KB
SRAM	2 KB	2 KB
Flash	32 KB	32 KB
USB connector	Mini USB-B	USB-B
UART	1	1

Pin layout

NANO PINOUT



! Absolute MAX per pin 40mA recommended 20mA

! Absolute MAX 200mA for entire package

! Analog exclusively Pins

The power sum for each pin's group should not exceed 100mA



www.bq.com
19 AUG 2014
ver 3 rev 1

HC-05 Bluetooth module

Bluetooth

Bluetooth is een open standaard voor draadloze verbindingen tussen apparaten op korte afstand. Het normale bereik is 1 tot 10m, maar dat kan opgevoerd worden tot 100 m. BT werkt op de 2.45 GHz band volgens het ‘point to multipoint’ principe. Dit betekent dat 1 BT bron kan verbinden met meerdere BT ontvangers.

Om de verbinding tot stand te brengen moeten de apparaten gepaird worden. Hierbij moet een pincode van de BT ontvanger ingegeven worden. Om praktische redenen doen we dat met de Android smartphone (master). Vanaf dan kan BT communicatie plaatsvinden met de slave.

Specificaties

Ik gebruik de ZS-040 versie van de populaire [HC-05](#)

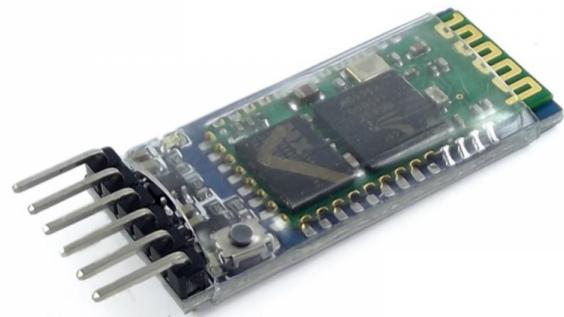
Bluetooth module. Deze kan werken in 2 modes :

- Slave : kan enkel connecties toelaten van andere BT apparaten.
- Master : kan zelf een BT connectie opzetten.

Voor dit project wordt de module in Slave mode gebruikt; de smartphone of tablet zal de BT connectie initiëren.

Op de module zit een EGBT-045MS Bluetooth chip die werkt op 3.3V. Daarnaast zit er ook een 3.3V spanning regulator die de ingangsspanning (3.6 – 6V) omzet naar de vereiste 3.3V.

De pincode voor de pairing is 1234.

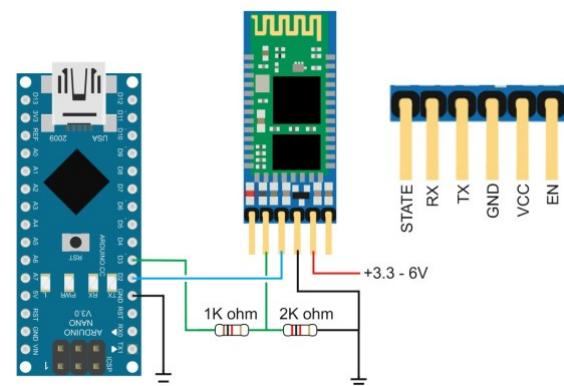


Pin layout

Doordat de Arduino Nano 5V op zijn pinnen zet bij een hoog signaal, moeten we gebruik maken van een spanningsdeler voor de RXD pin van de BT module. Die kunnen we eenvoudig maken met een weerstand van 1K en 2K. Twee derden van 5V is namelijk 3.3V. De 3.3V van de TXD pin van de BT module wordt door de Nano als een hoog signaal aanzien.

Pin HC-05	Functie
STATE	Status (Hoog=verbonden)
RXD	Ontvang seriële data (3.3V via spanningsdeler!!!)
TXD	Verzend seriële data (3.3V)
GND	Massa
VCC	Voedingsspanning (3.6-6V)
EN	Enable AT commando mode (Hoog)

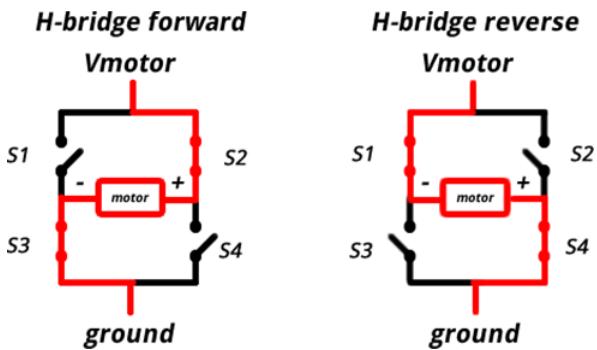
HC-05 BASIC SET UP



L293DNE dubbele H-brug motor controller

H-brug

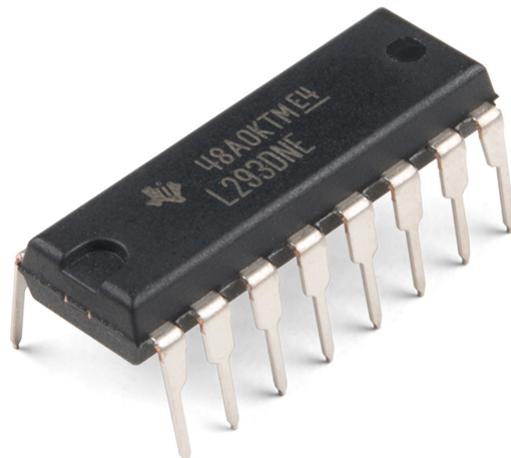
Om de polariteit van een gelijkstroommotor te regelen maken we gebruik van een H-brug. Het principe is eenvoudig : door de schakelaars S1-S4 en S2-S3 open of gesloten te maken zal de stroomrichting doorheen de motor wijzigen en de draairichting uiteraard ook. Er zijn verschillende motor controller modules beschikbaar op basis van deze dubbele H-brug controller chips, compleet met heat sink en connectoren voor motoren, sturing en voeding. Ik gebruik echter het basis L293DNE IC om het geheel zo compact mogelijk te houden.



Specificaties

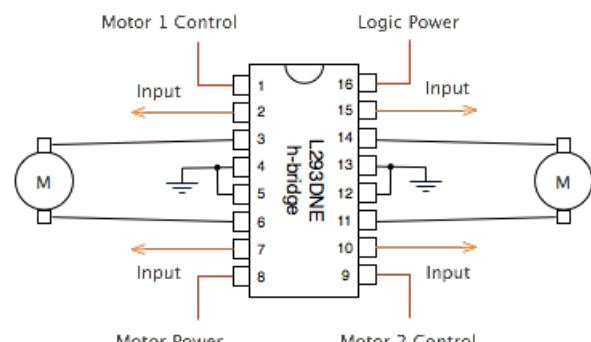
De [L293DNE](#) van Texas Instruments is een PDIP16 IC die een dubbele H-brug bevat om 2 gelijkstroommotoren onafhankelijk van elkaar te besturen.

- Voeding motoren : 4.5 – 36V
- Voeding IC : 5V
- Output stroom : 600mA
- Output stroom piek : 1.2A
- Clamp diodes voor inductieve stroomonderdrukking.
- Interne ESD (elektrostatische ontlading) bescherming
- Ruisonderdrukking op inputs



Pin layout

#	Pin	Functie
1	1,2EN	Controle motor 1
2	1A	Sturing 1 motor 1
3	1Y	Terminal 1 motor 1
4		Heat sink & GND
5		
6	2Y	Terminal 2 motor 1
7	2A	Sturing 2 motor 1
8	VCC2	Voeding motoren (4.5-36V)
9	3,4EN	Controle motor 2
10	3A	Sturing 1 motor 2



#	Pin	Functie
11	3Y	Terminal 1 motor 2
12		Heat sink & GND
13		
14	4Y	Terminal 2 motor 2
15	4A	Sturing 2 motor 2
16	VCC1	Voeding IC (5V)

De snelheid van de motor wordt bepaald door de waarde op de controle pin (EN) van het IC. Hiervoor is dus een PWM pin van onze Arduino nodig. De draairichting van de motor wordt bepaald door de hoog/laag waarden op de sturing pinnen (A). Hiervoor zijn gewone digitale pinnen van de Arduino voldoende.

Een en ander wordt verduidelijkt in volgende tabel.

Controle pin	Sturing 1 pin	Sturing 2 pin	Functie
Hoog*	Laag	Hoog	Motor draait rechts
Hoog*	Hoog	Laag	Motor draait links
Hoog	Laag	Laag	Motor stopt snel (remt af)
Hoog	Hoog	Hoog	Motor stopt snel (remt af)
Laag	Hoog/Laag	Hoog/Laag	Motor stopt traag (loopt uit)

(*) Motorsnelheid is afhankelijk van de waarde op de PWM controle pin.

9V PP3 batterijhouder

De batterijhouder biedt plaats aan een 9V PP3 batterij en heeft een ingebouwde aan/uit schakelaar.

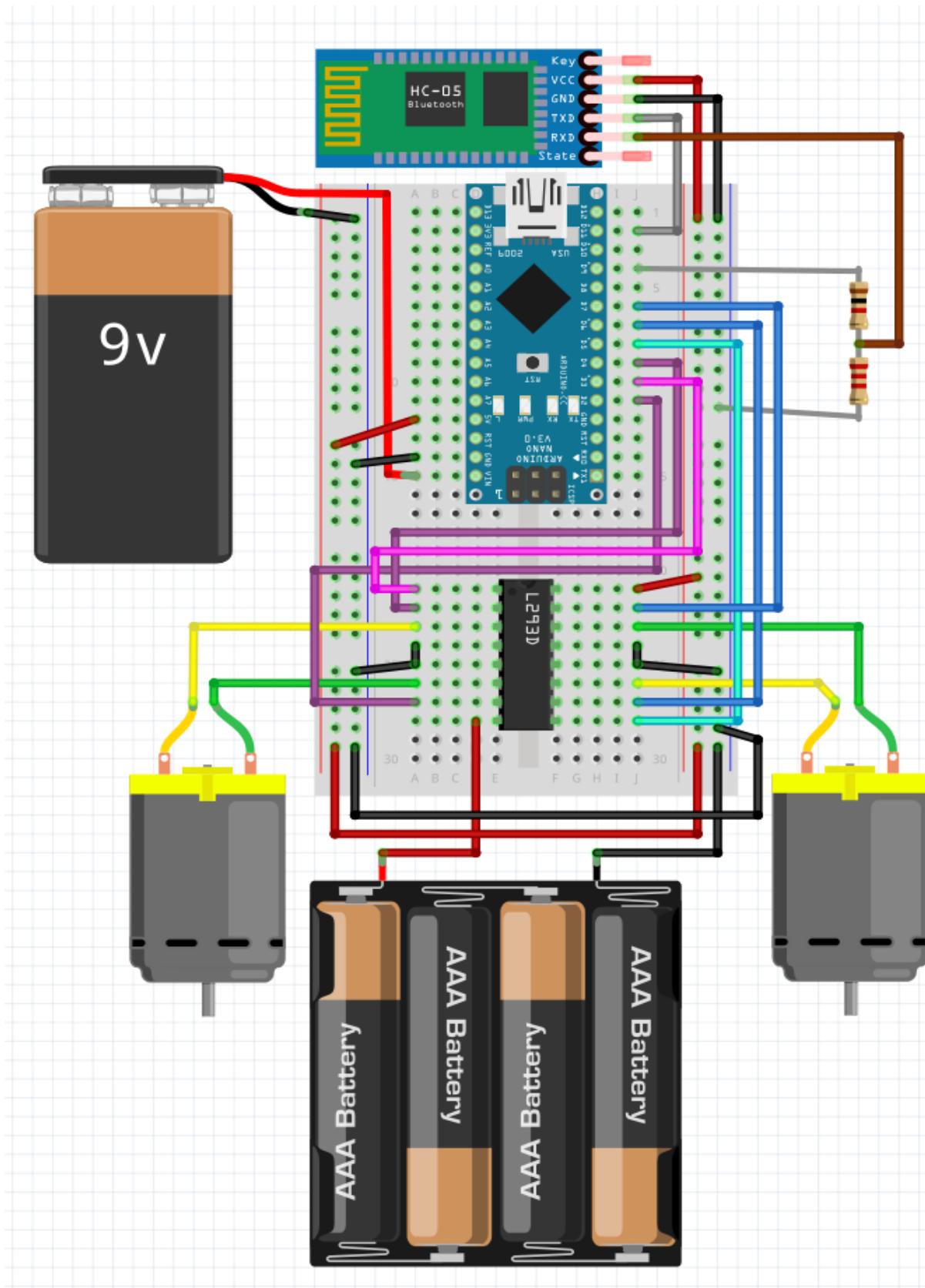
Hij dient voor de voeding van de Arduino Nano en wordt aangesloten op de VIN pin. Die is berekend op een ingangsspanning van 7 tot 9V.

De ingebouwde spanningsregelaar van de Arduino zet de voedingsspanning om naar de vereiste 5V en 3.3V voor de interne werking van de microcontroller.



Zumo Rover besturing

Fritzing schema



Schakeling

We verbinden de verschillende onderdelen als volgt met elkaar :

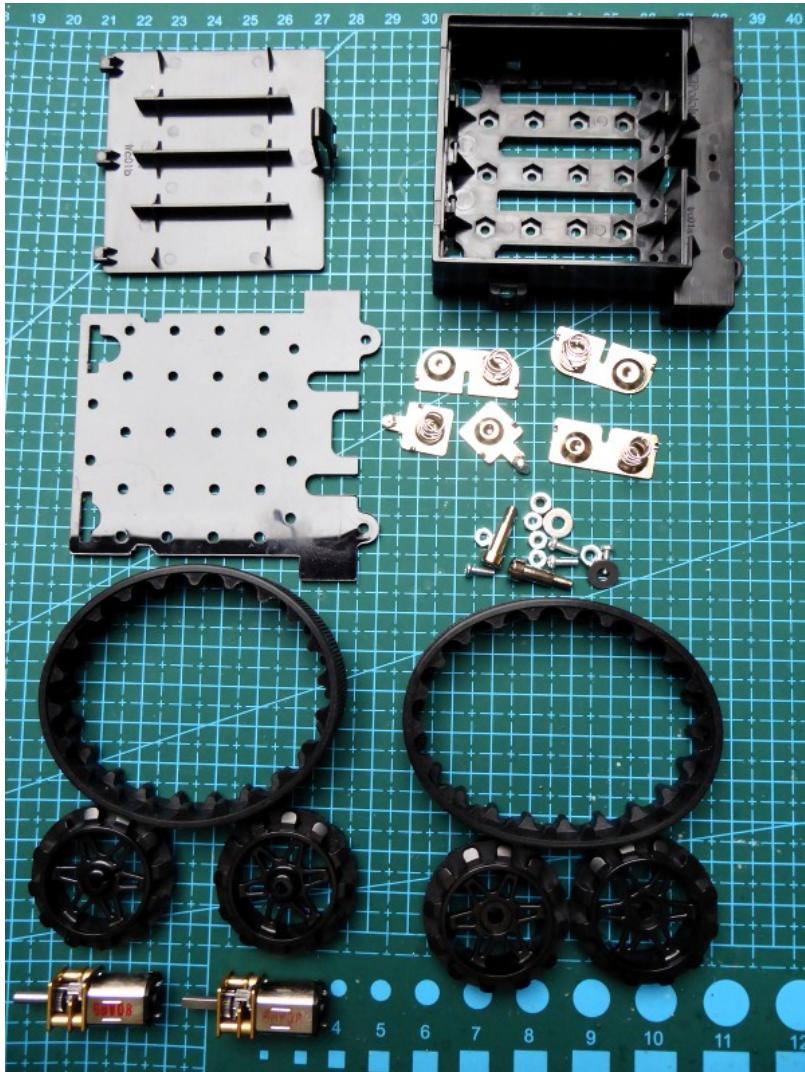
Van		Naar	
Onderdeel	Pin	Onderdeel	Pin
Arduino Nano	5V	Breadboard	+
	GND		-
9V PP3 batterij	+	Arduino Nano	Vin
	-	Breadboard	-
L293DNE motor controller	1	Arduino Nano	3
	2		4
	3	Motor links	+
	4	Breadboard	-
	5		
	6	Motor links	-
	7	Arduino Nano	2
	8	4x1.5V AA batterijen	+
	9	Arduino Nano	5
	10		6
	11	Motor rechts	+
	12	Breadboard	-
	13		
	14	Motor rechts	-
	15	Arduino Nano	7
	16	Breadboard	+
HC-05 Bluetooth	Vcc	Breadboard	+
	GND		-
	TxD	Arduino Nano	11
	RxD		9*

(*) via een spanningsdeler wordt de 5V van pin 9 omgezet naar 3.3V die geschikt is voor de RxD pin van de HC-05.

Opgelet : de massa's van de Arduino Nano, de 9V voeding en de Zumo voeding moeten met elkaar verbonden worden !

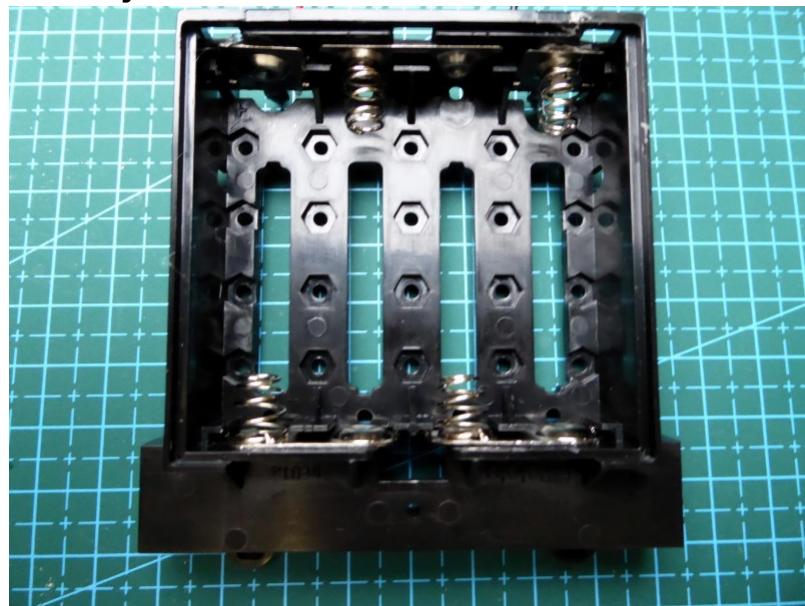
Zumo Rover assemblage

Zumo chassis



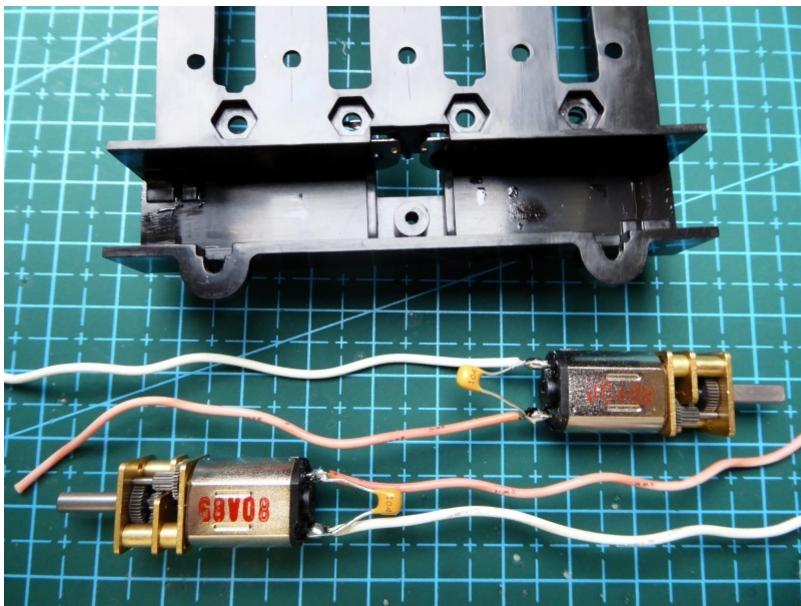
Het chassis wordt geleverd als aparte onderdelen. De officiële assemblage handleiding kan je [hier](#) vinden.

Batterij contacten



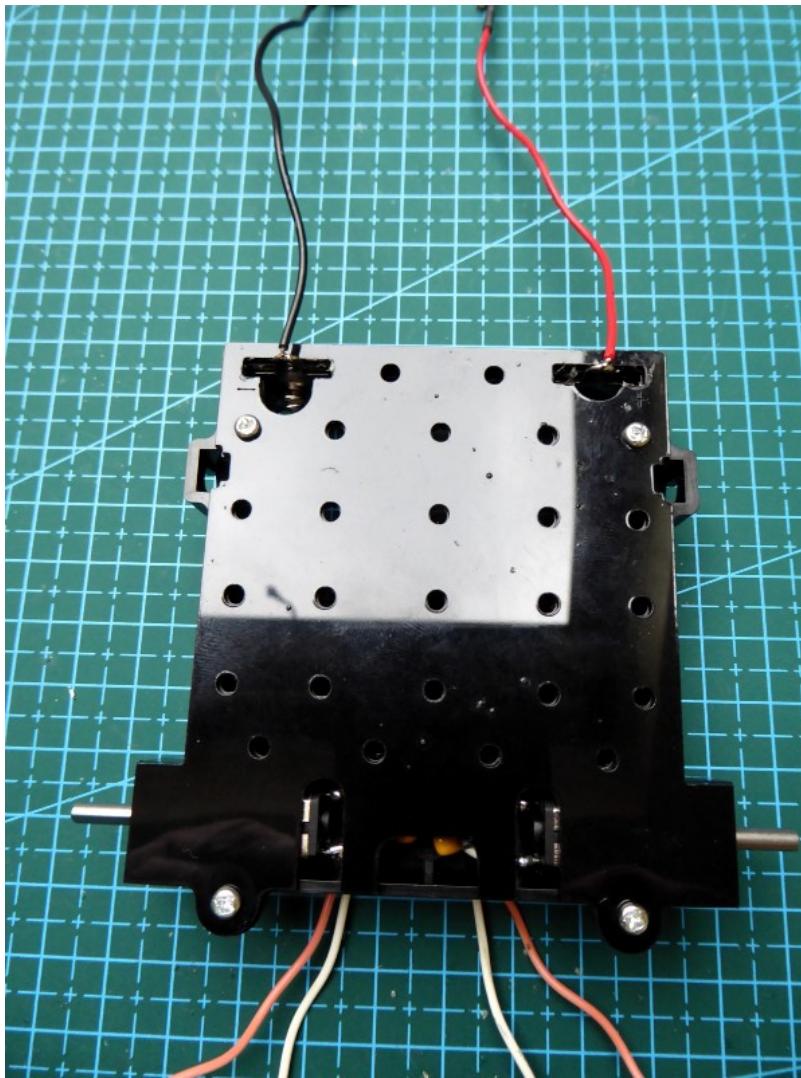
Het batterij compartiment na het plaatsen van de contacten. Kleef de positieve en negatieve batterij connector vast met je lijmpistool op het chassis.

Motoren



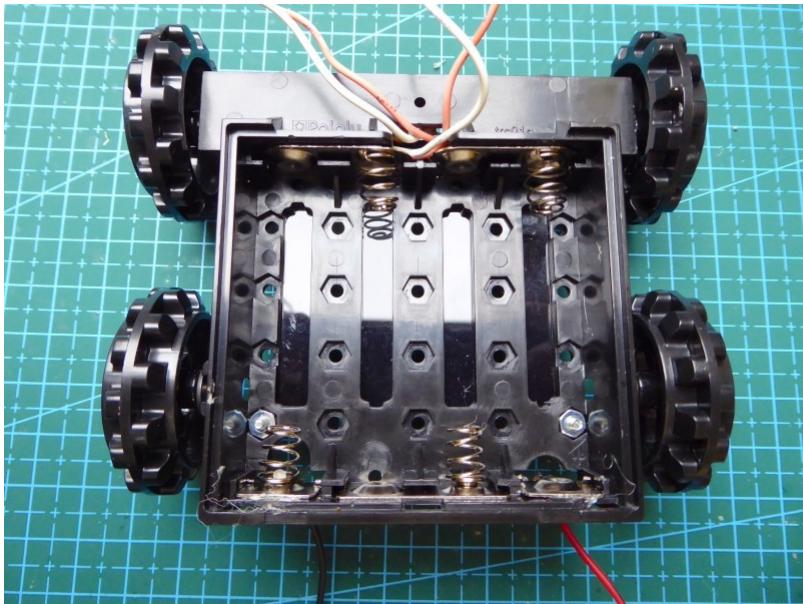
Om de elektrische storing van de motoren te reduceren wordt aangeraden om [een ceramische condensator van 100 pF \(0,1 µF\)](#) over de terminal contacten van de motoren te solderen. Dat is vooral belangrijk om de Arduino Nano en de Bluetooth module goed te laten werken.

Bovenplaat



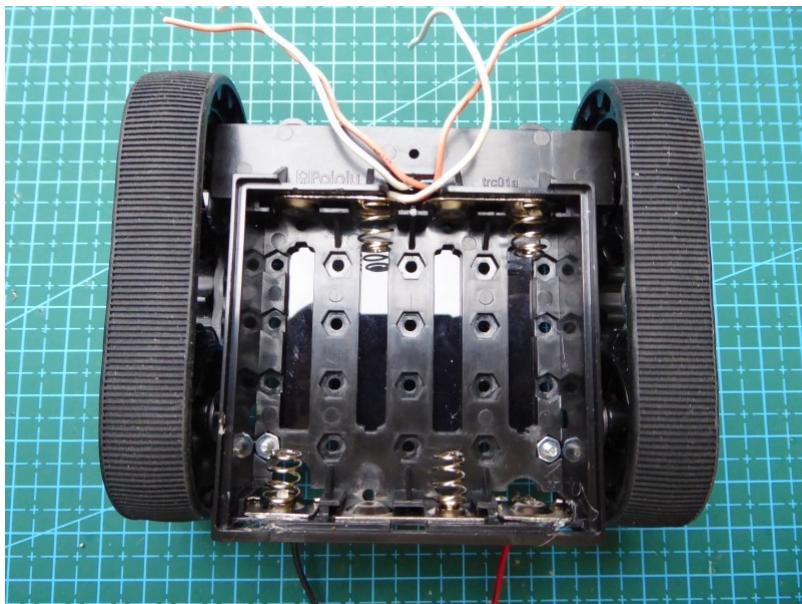
Na het plaatsen van de bovenplaat soldeer je de voedingskabeltjes op de batterij connectoren.

Wieren



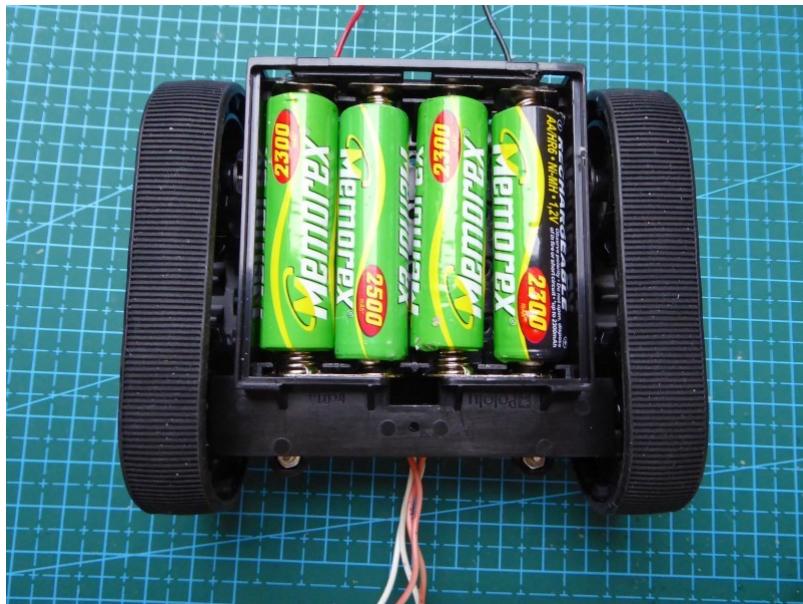
Het chassis na het monteren van de passieve wielen en de aandrijf wielen op de motoren.

Rupsbanden



De rupsbanden zijn nu over de wielen geplaatst.

Batterijen



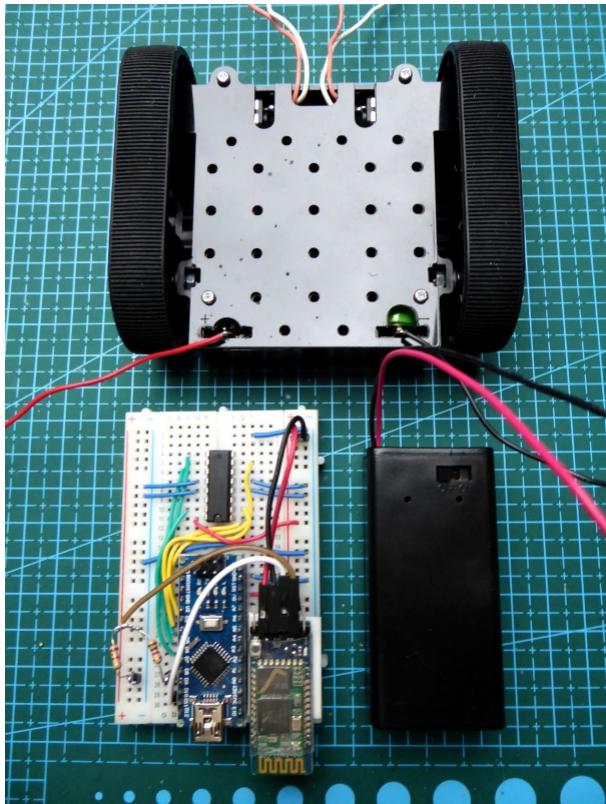
Plaats de batterijen in het batterij compartiment. Let op de juiste polariteit.

Batterijdeksel



Klik het batterijdeksel dicht.

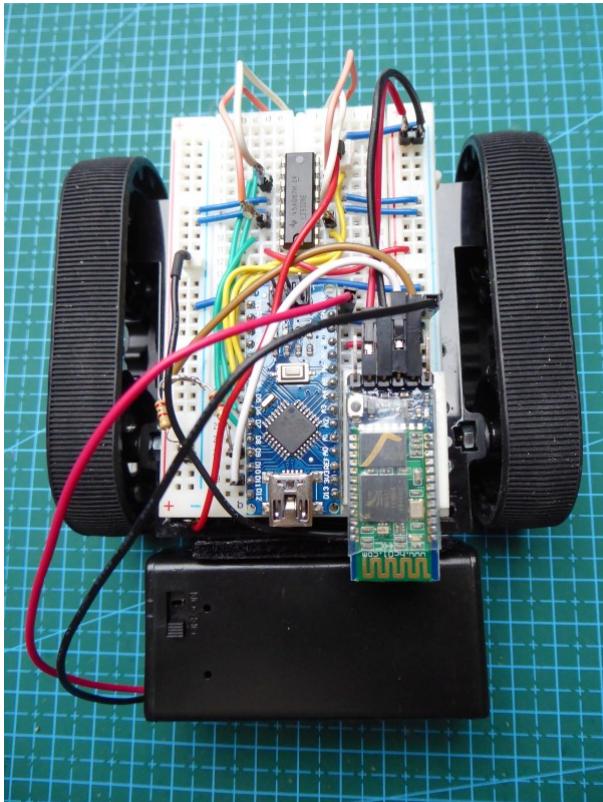
Rover niet geassembleerd



Op dit overzicht zie je alle onderdelen van het Zumo robotvoertuig :

- bovenaan het geassembleerde zumo chassis
- rechts onderaan de 9V batterij voeding voor de Arduino Nano
- links onderaan het breadboard met :
 - L293DNE dubbele H-brug motor controller
 - HC-05 Bluetooth module
 - Arduino Nano
 - Spanningsdeler voor de RxD pin van de HC-05 BT module

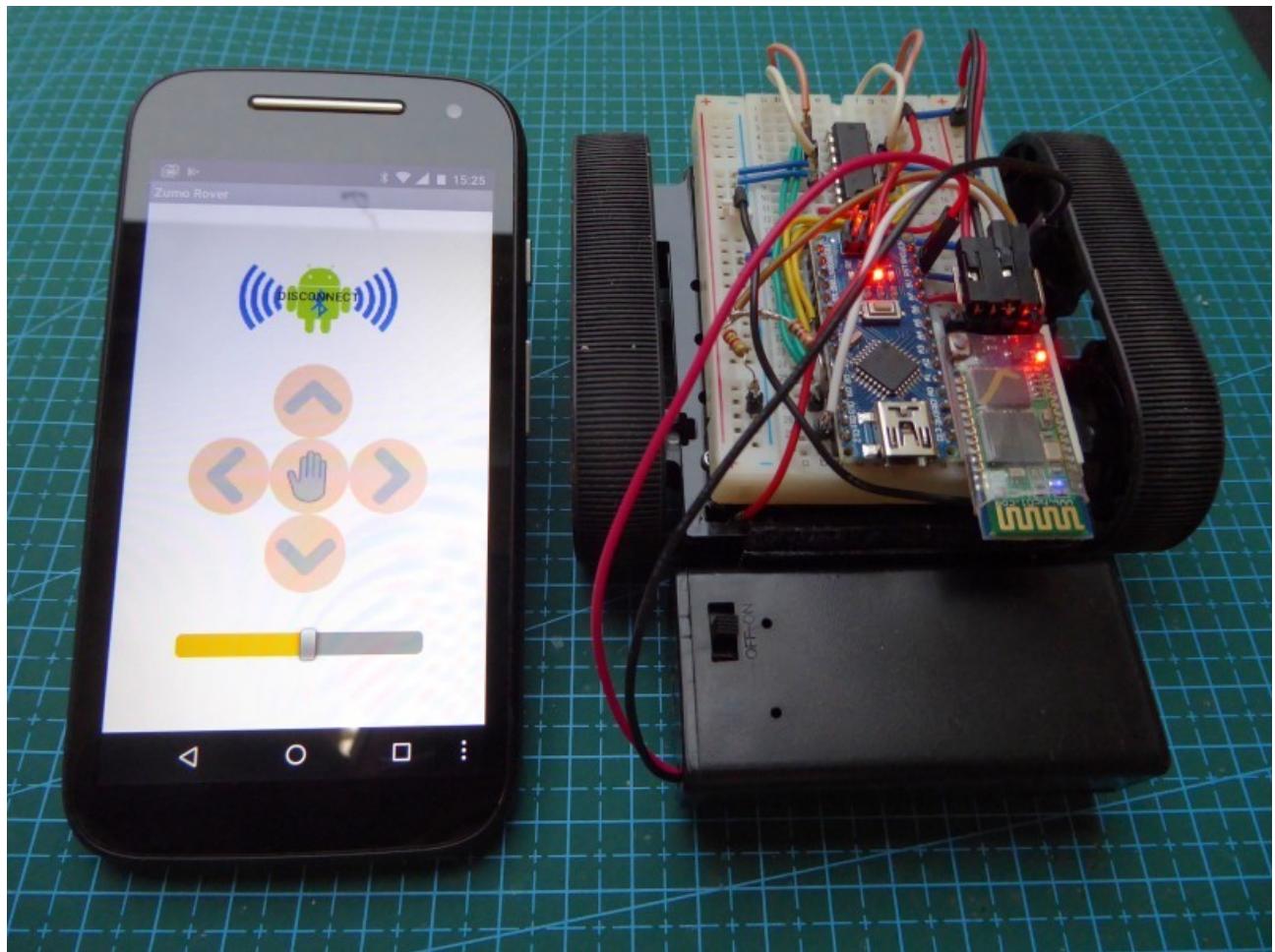
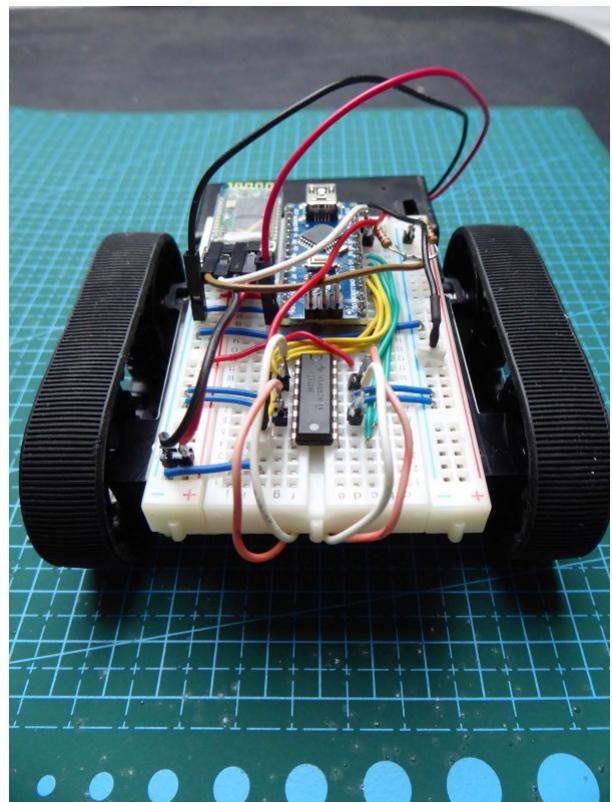
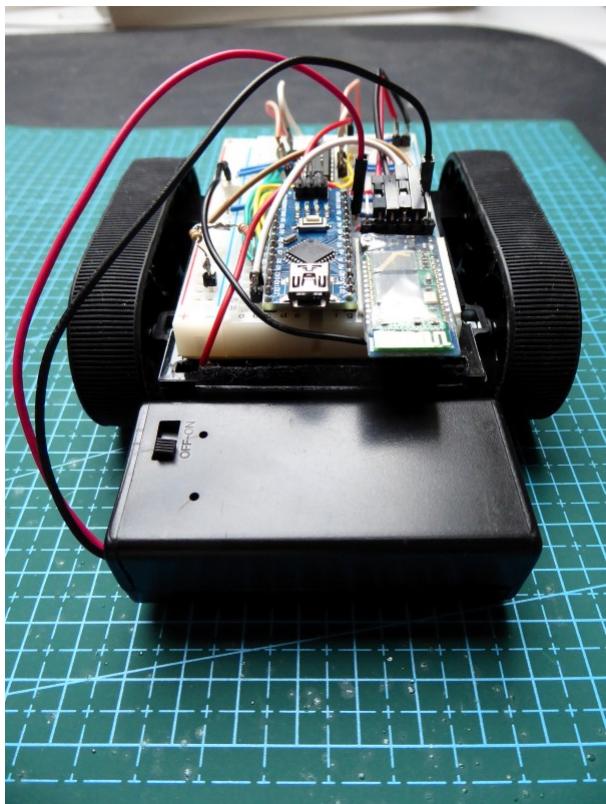
Rover geassembleerd



Het breadboard werd met de zelfklevende onderkant op het zumo chassis gekleefd.

De 9V batterijhouder werd met velcro aan de voorkant van het voertuig bevestigd. De positieve pool werd verbonden met de Vin pin van de Arduino, de negatieve pool met de negatieve rail van het breadboard.

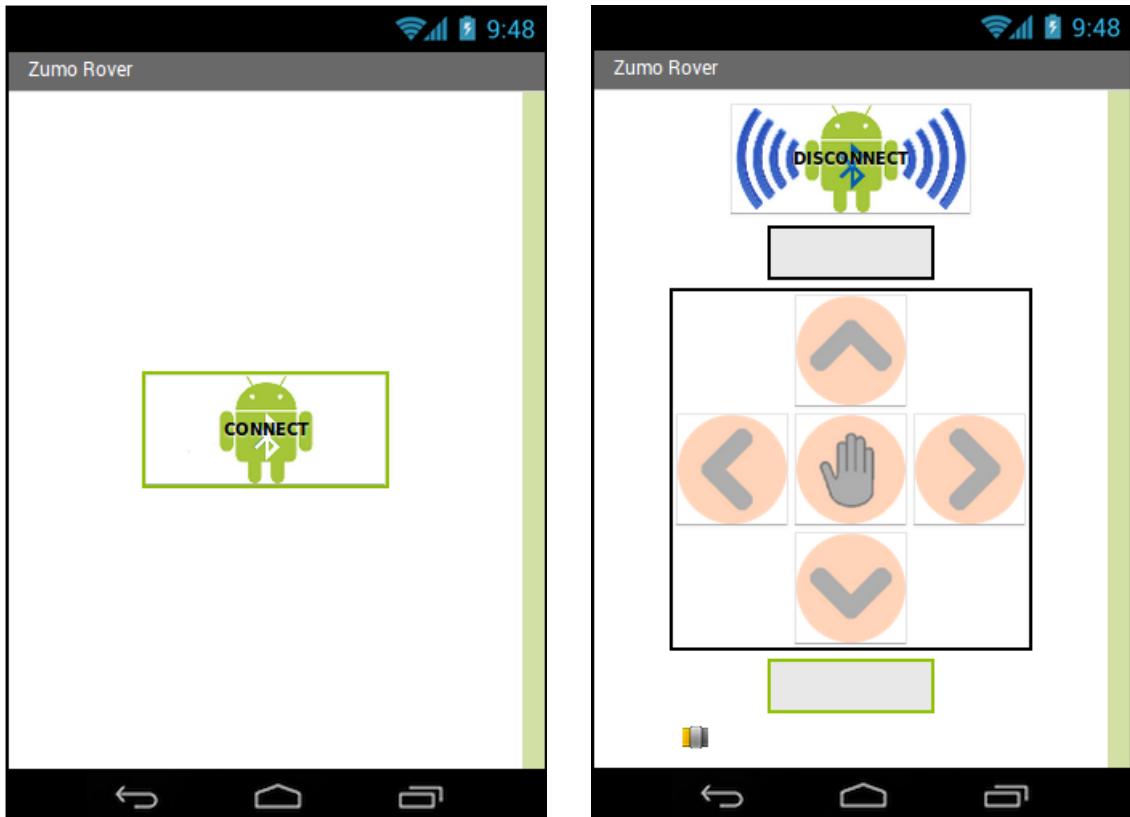
Tenslotte zijn nog de motoren verbonden met de respectievelijke motor terminal pinnen van de L293DNE dubbele H-brug chip.



Software

MIT App Inventor code

Designer



Interface element	Functie	Tonen bij BT connectie
LstBluetooth	Lijst van de beschikbare Bluetooth apparaten. Tap hierop om te verbinden met een BT apparaat.	Nee
BtnDisconnect	Knop om de Bluetooth verbinding te verbreken.	Ja
HASpacer1	Wat ruimte tussen de BT knop en de besturingsknoppen.	Ja
TblBesturing	Een tabel om de knoppen van de besturing te positioneren.	Ja
BtnVooruit	Knop vooruit. Bij tap wordt een 'V' verstuurd via Bluetooth.	Ja
BtnHalt	Knop stoppen. Bij tap wordt een 'H' verstuurd via Bluetooth.	Ja
BtnAchteruit	Knop achteruit. Bij tap wordt een 'A' verstuurd via Bluetooth.	Ja
BtnLinks	Knop links. Bij tap wordt een 'L' verstuurd via Bluetooth.	Ja
BtnRechts	Knop rechts. Bij tap wordt een 'R' verstuurd via Bluetooth.	Ja
HASpacer2	Wat ruimte tussen de besturingsknoppen en de snelheid slider.	Ja

Interface element	Functie	Tonen bij BT connectie
SldrSnelheid	Slider om de snelheid te kiezen. Bij verandering wordt 'Sxxx' doorgestuurd; 'S' en een getal (0-255) zonder voorloopnullen.	Ja
BluetoothClient1	Bluetooth Client verzorgt de BT communicatie.	Onzichtbaar

Blocks Bluetooth

```
when LstBluetooth .BeforePicking
do set LstBluetooth . Elements to BluetoothClient1 . AddressesAndNames
```

Deze code zorgt ervoor dat de lijst LstBluetooth opgevuld wordt met de beschikbare BT apparaten met hun naam en adres. De lijst wordt vervolgens getoond op een nieuw scherm waaruit je het juiste BT apparaat kan kiezen. Dat is uiteraard onze HC-05 BT module die verbonden is met de Arduino.

```
when LstBluetooth .AfterPicking
do set LstBluetooth . Selection to call BluetoothClient1 . Connect
   address LstBluetooth . Selection
   if BluetoothClient1 . IsConnected
   then set LstBluetooth . Visible to false
        set BtnDisconnect . Visible to true
        set HASpacer1 . Visible to true
        set tblBesturing . Visible to true
        set BtnVooruit . Visible to true
        set BtnHalt . Visible to true
        set BtnAchteruit . Visible to true
        set BtnLinks . Visible to true
        set BtnRechts . Visible to true
        set HASpacer2 . Visible to true
        set sldrSnelheid . Visible to true
```

Na het kiezen van de HC-05 BT module uit de lijst LstBluetooth wordt deze code uitgevoerd. Er wordt geprobeerd een Bluetooth verbinding op te zetten met de module. Als dit lukt wordt de zichtbaarheid van de diverse interface elementen van de app aangepast om het besturingsscherm te tonen (zie tabel vorige pagina).

```

when BtnDisconnect .Click
do call BluetoothClient1 .Disconnect
set LstBluetooth .Visible to true
set BtnDisconnect .Visible to false
set HASpacer1 .Visible to false
set tblBesturing .Visible to false
set BtnVooruit .Visible to false
set BtnHalt .Visible to false
set BtnAchteruit .Visible to false
set BtnLinks .Visible to false
set BtnRechts .Visible to false
set HASpacer2 .Visible to false
set sldrSnelheid .Visible to false

```

Als er op de knop BtnDisconnect wordt gedrukt, dan wordt de Bluetooth verbinding met de HC-05 BT module verbroken en wordt de zichtbaarheid van de diverse elementen aangepast om nog enkel de lijst LstBluetooth te tonen (zie tabel vorige pagina). Er kan dan terug een nieuwe BT verbinding opgezet worden.

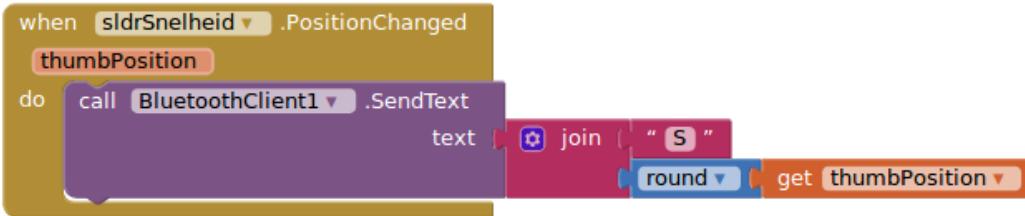
Blocks Besturing

```

when BtnVooruit .Click
do call BluetoothClient1 .SendText
text " V "
when BtnHalt .Click
do call BluetoothClient1 .SendText
text " H "
when BtnAchteruit .Click
do call BluetoothClient1 .SendText
text " A "
when BtnLinks .Click
do call BluetoothClient1 .SendText
text " L "
when BtnRechts .Click
do call BluetoothClient1 .SendText
text " R "

```

Deze code zorgt ervoor dat bij het drukken op de besturingsknoppen het juiste commando via Bluetooth wordt doorgestuurd naar de HC-05 BT module. Het commando bestaat uit één enkele letter.



Voor het doorsturen van de snelheid wordt deze code gebruikt. Bij het slepen van de slider SldrSnelheid wordt een tekst van maximaal 4 karakters doorgestuurd. Het eerste karakter is steeds een ‘S’ van snelheid, gevuld door de waarde van de slider positie (0-255). Er worden geen voorloopnullen meegestuurd.

Arduino sketch

Bibliotheken

```

// bibliotheken
#include <SoftwareSerial.h>

const int txdPin = 9;
const int rxdPin = 11;
SoftwareSerial zumoSerial = SoftwareSerial(rxdPin, txdPin);

```

Voor de communicatie met de HC-05 Bluetooth module maak ik gebruik van softwarematige seriële communicatie. Dit heeft als voordeel dat ik de seriële console kan blijven gebruiken voor debugging en dat ik de HC-05 niet moet loskoppelen voor het uploaden van de sketch in de Nano. Er wordt een seriële communicatie zumoSerial opgezet op pinnen 9 (Tx) en 11 (Rx).

Constanten

```

// constanten
const int motorL_sturing1 = 4;      // pins linker motor
const int motorL_sturing2 = 2;
const int motorL_enable = 3;         // PWM !
const int motorR_sturing1 = 6;      // pins rechter motor
const int motorR_sturing2 = 7;
const int motorR_enable = 5;         // PWM !
const int min_snelheid = 96;        // beginsnelheid PWM 0-255
const char VOORUIT    = 'V';        // commando's besturing
const char ACHTERUIT = 'A';
const char LINKS     = 'L';
const char RECHTS    = 'R';
const char STOPPEN   = 'H';
const char SNELHEID  = 'S';

```

Alle pinnen voor de controle en sturing van de beide motoren worden hier gedefinieerd. Er worden tevens meer beschrijvende constanten aangemaakt voor de letter commando’s die we doorkrijgen van de besturing app.

Variabelen

```
// variabelen
int motor_snelheid = min_snelheid;
int draaitijd = 0; // draaitijd bij links/rechts draaien (ms)
// is omgekeerd evenredig met de motorsnelheid
```

De motorsnelheid en draaitijd worden hier als globale variabele gedefinieerd, zodat deze toegankelijk zijn doorheen de gehele sketch. De snelheid wordt geïnitialiseerd op de minimum snelheid.

Setup

```
void setup() {
    // pinnen H-brug als output
    pinMode(motorL_sturing1, OUTPUT);
    pinMode(motorL_sturing2, OUTPUT);
    pinMode(motorL_enable, OUTPUT);
    pinMode(motorR_sturing1, OUTPUT);
    pinMode(motorR_sturing2, OUTPUT);
    pinMode(motorR_enable, OUTPUT);

    // zumo seriële communicatie starten
    zumoSerial.begin(9600);

    // seriële debug console
    Serial.begin(9600);

    // zumo stoppen
    zumo(STOPPEN);
}
```

In de setup worden de controle- en sturingspinnen van de beide motoren als output gedefinieerd. De zumo seriële communicatie wordt gestart, alsook de seriële debug console. Tenslotte wordt het robotvoertuig gestopt.

Loop

```
void loop() {
    // lees inkomende BT signalen
    while (zumoSerial.available() > 0) {
        char kar = zumoSerial.read();
        switch(kar) {
            case VOORUIT:
                Serial.println("VOORUIT");
                zumo(VOORUIT);
                break;
            case ACHTERUIT:
                Serial.println("ACHTERUIT");
                zumo(ACHTERUIT);
                break;
            case STOPPEN:
                Serial.println("STOPPEN");
                zumo(STOPPEN);
                break;
            case LINKS:
                Serial.println("LINKS");
                zumo(LINKS);
                break;
            case RECHTS:
                Serial.println("RECHTS");
                zumo(RECHTS);
                break;
            case SNELHEID:
                motor_snelheid = zumoSerial.parseInt(); // lees de snelheid in als integer
                Serial.print("SNELHEID ");
                Serial.println(motor_snelheid);
                zumo(SNELHEID);
                break;
        }
    }
}
```

In de loop worden de besturingscommando's van de HC-05 Bluetooth module serieel ingelezen. Deze worden in een switch/case statement gedispatched, doorgestuurd naar de debug console en meegegeven als parameter aan de zumo functie die de volledige besturing van het robotvoertuig voor zijn rekening neemt.

Het uitlezen van de snelheid ('Sxxx') gebeurt met het commando `zumoSerial.parseInt()` die de cijfers na de 'S' automatisch omzet naar de motor_snelheid, ongeacht of er 1, 2 of 3 cijfers doorkomen.

Functies

```
// besturing zumo robotvoertuigchassis
void zumo(char commando) {

    switch(commando) {

        case VOORUIT:
            // beide motoren -> sturing 1 laag - sturing 2 hoog
            digitalWrite(motorL_sturing1, LOW);
            digitalWrite(motorL_sturing2, HIGH);
            digitalWrite(motorR_sturing1, LOW);
            digitalWrite(motorR_sturing2, HIGH);
            // beide motoren enablen met motorsnelheid (PWM)
            analogWrite(motorL_enable, motor_snelheid);
            analogWrite(motorR_enable, motor_snelheid);
            // debug info
            Serial.print("Zumo VOORUIT - snelheid ");
            Serial.println(motor_snelheid);
            break;

        case LINKS:
            // linker motor -> sturing 1 hoog - sturing 2 laag
            // rechter motor -> sturing 1 laag - sturing 2 hoog
            digitalWrite(motorL_sturing1, HIGH);
            digitalWrite(motorL_sturing2, LOW);
            digitalWrite(motorR_sturing1, LOW);
            digitalWrite(motorR_sturing2, HIGH);
            // beide motoren enablen met motorsnelheid (PWM)
            analogWrite(motorL_enable, motor_snelheid);
            analogWrite(motorR_enable, motor_snelheid);
            // debug info
            Serial.print("Zumo LINKS - snelheid ");
            Serial.print(motor_snelheid);
            draaitijd = map(motor_snelheid, 0, 255, 1000, 0); // omgekeerd evenredig met
motorsnelheid
            Serial.print(" - tijd ");
            Serial.println(draaitijd);
            // na draaitijd stoppen
            delay(draaitijd);
            zumo(STOPPEN);
            break;

        case STOPPEN:
            // beide motoren -> sturing 1 laag - sturing 2 laag
            digitalWrite(motorL_sturing1, LOW);
            digitalWrite(motorL_sturing2, LOW);
            digitalWrite(motorR_sturing1, LOW);
            digitalWrite(motorR_sturing2, LOW);
            // beide motoren -> enable hoog (remmen !)
            analogWrite(motorL_enable, HIGH);
            analogWrite(motorR_enable, HIGH);
            // debug info
            Serial.println("Zumo STOPPEN");
            break;

        . . .
    }
}
```

```

. . .

case RECHTS:
    // linker motor -> sturing 1 laag - sturing 2 hoog
    // rechter motor -> sturing 1 hoog - sturing 2 laag
    digitalWrite(motorL_sturing1, LOW);
    digitalWrite(motorL_sturing2, HIGH);
    digitalWrite(motorR_sturing1, HIGH);
    digitalWrite(motorR_sturing2, LOW);
    // beide motoren enablen met motorsnelheid (PWM)
    analogWrite(motorL_enable, motor_snelheid);
    analogWrite(motorR_enable, motor_snelheid);
    // debug info
    Serial.print("Zumo RECHTS - snelheid ");
    Serial.print(motor_snelheid);
    draaitijd = map(motor_snelheid, 0, 255, 1000, 0); // omgekeerd evenredig met
motorsnelheid
    Serial.print(" - tijd ");
    Serial.println(draaitijd);
    // na draaitijd stoppen
    delay(draaitijd);
    zumo(STOPPEN);      break;

case ACHTERUIT:
    // beide motoren -> sturing 1 hoog - sturing 2 laag
    digitalWrite(motorL_sturing1, HIGH);
    digitalWrite(motorL_sturing2, LOW);
    digitalWrite(motorR_sturing1, HIGH);
    digitalWrite(motorR_sturing2, LOW);
    // beide motoren enablen met motorsnelheid (PWM)
    analogWrite(motorL_enable, motor_snelheid);
    analogWrite(motorR_enable, motor_snelheid);
    // debug info
    Serial.print("Zumo ACHTERUIT - snelheid ");
    Serial.println(motor_snelheid);
    break;

case SNELHEID:
    // beide motoren enablen met motorsnelheid (PWM)
    analogWrite(motorL_enable, motor_snelheid);
    analogWrite(motorR_enable, motor_snelheid);
    // debug info
    Serial.print("Zumo SNELHEID ");
    Serial.println(motor_snelheid);
    break;

default:
    // debug info
    Serial.println("Zumo ONGELDIG COMMANDO");
    break;
}
}

```

De gehele besturing van het robotvoertuig zit in deze ene functie. Afhankelijk van het doorgestuurde karakter in de parameter worden via een switch/case statement de juiste signalen naar de controle- en besturingspinnen van de L293DNE dubbele H-brug doorgestuurd die de motoren in de juiste richting en met de gewenste snelheid laten draaien.

De commando's LINKS en RECHTS worden beperkt in de tijd, omdat het robotvoertuig anders gaat spinnen om zijn as. De draaitijd wordt hiervoor via de functie map() omgekeerd evenredig gekozen aan de huidige motorsnelheid. Dus hoe rapper de snelheid, hoe korter de tijd van het draaien. Na de draaitijd wordt de zumo gestopt.