Robotica met Arduino – Les 1

Bouwen van de robot

Woorden om te onthouden: chassis, shield, DC motor

Boodschappenlijst:

- Arduino
- Arduino IDE
- chassis set
- Arduino motor shield
- schroevendraaier
- Verschillende snoertjes
- LED's

Introductie

Deze les gaan we het chassis[1] van de robot opbouwen. We monteren 2 DC motoren[2], een Arduino en een motor shield[3] op het chassis. Hierna testen we de motoren en kunnen we ons eerste programma voor de robot schrijven. Dit gebeurt in de Arduino IDE.

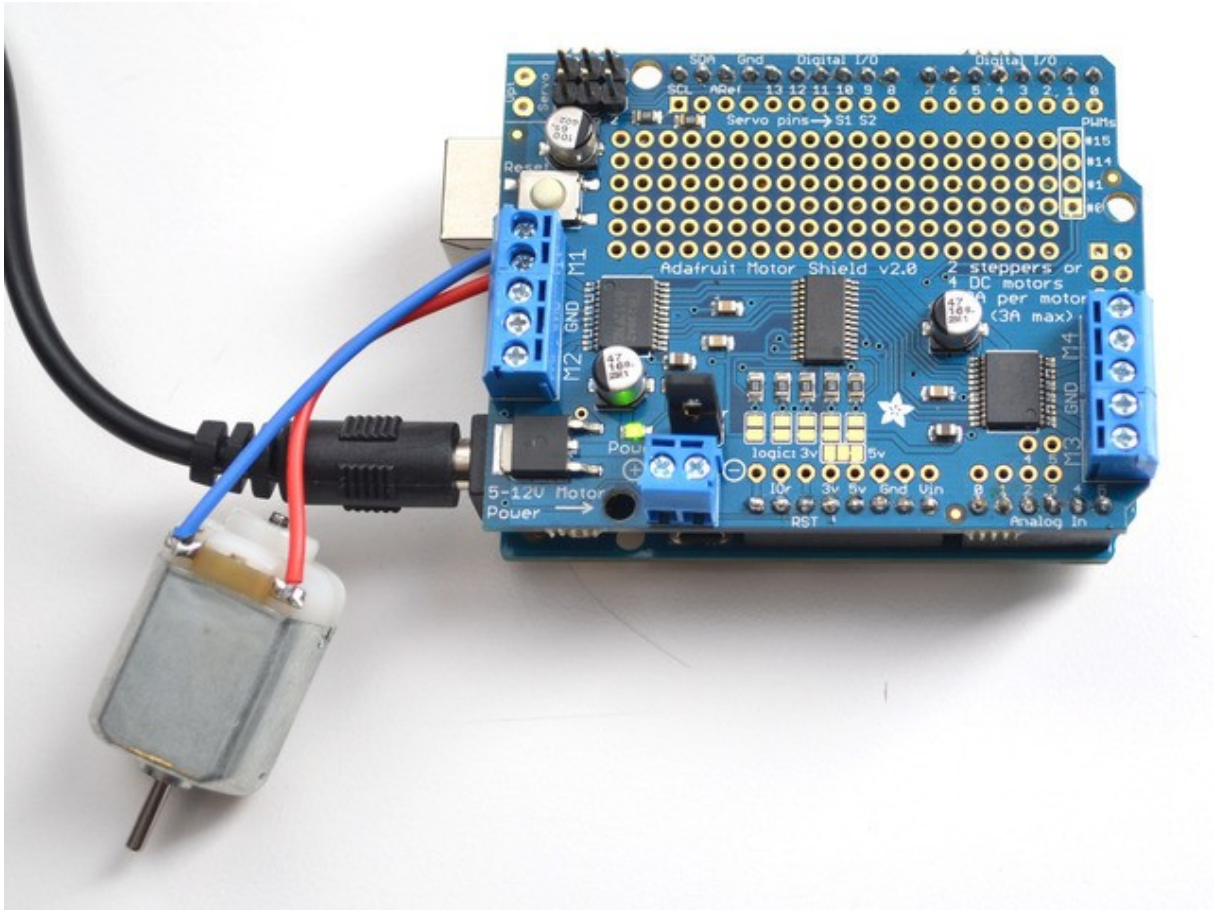
Chassis opbouwen.

Zorg dat je een compleet chassis pakket hebt en een schroevendraaier. Zet het chassis in elkaar met behulp van het bijgevoegde instructieblad. LET OP!!! de DC motoren zijn een beetje lastig om te monteren, schroef ze niet meteen helemaal vast, maar zorg eerst dat alle boutjes/moertjes op hun plek zitten. Kijk goed op het blad hoe alles zit.

1 Een chassis is een compleet rijdbaar onderstel van een voertuig, zonder enige opbouw (wikipedia)
2 Een DC motor of gelijkstroommotor is een motor waarin elektrische energie in de vorm van een gelijkstroom, omgezet wordt in mechanische energie
3 Een shield is een stuk elektronica dat precies op de Arduino past

Arduino en shield monteren

Zet het motor shield op de Arduino (**let goed op of de pootjes goed in de gaatjes passen!!!!)** en zorg er voor dat het goed vast zit. Daarna sluit je de DC motoren aan op het shield.



De ander sluit je aan op de aansluitpunten er naast. Je hebt nu twee motoren aangesloten met vier draadjes. Deze hele constructie kun je dan op je chassis plaatsen en vastzetten met een boutje/moertje.

DC motoren testen

Zorg dat je de juiste bibliotheken in laadt:

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"
```

Creëer het motor_shield object:

```
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
```

Creëer het DC motor object

Vraag de DC motor op bij de motor_shield

```
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);
```

Hier kiezen we de naam en de plaats van de motor. De (1) achter getMotor slaat op de poort waar je de motor op hebt aangesloten, als je op M”aansluit kies je hier voor (2). Als je een tweede motor aan wilt sluiten moet je de functie nog een keer opvragen met een andere naam (bijv myOtherMotor). Je kunt ipv myMotor en myOtherMotor bijv ook links en rechts kiezen als namen. Het wordt dan:

Adafruit_DCMotor *links = AFMS.getMotor(1);

Adufruit_DCMotor *rechts = AFMS.getMotor(2);

of zoiets, je kunt zelf de namen van de motoren kiezen, als je gedurende het hele script maar dezelfde namen gebruikt.

Maak connectie met de Controller

In je void setup() roep je het begin() commando op

```
AFMS.begin();
```

Stel de snelheid in

```
myMotor->setSpeed(100);
```

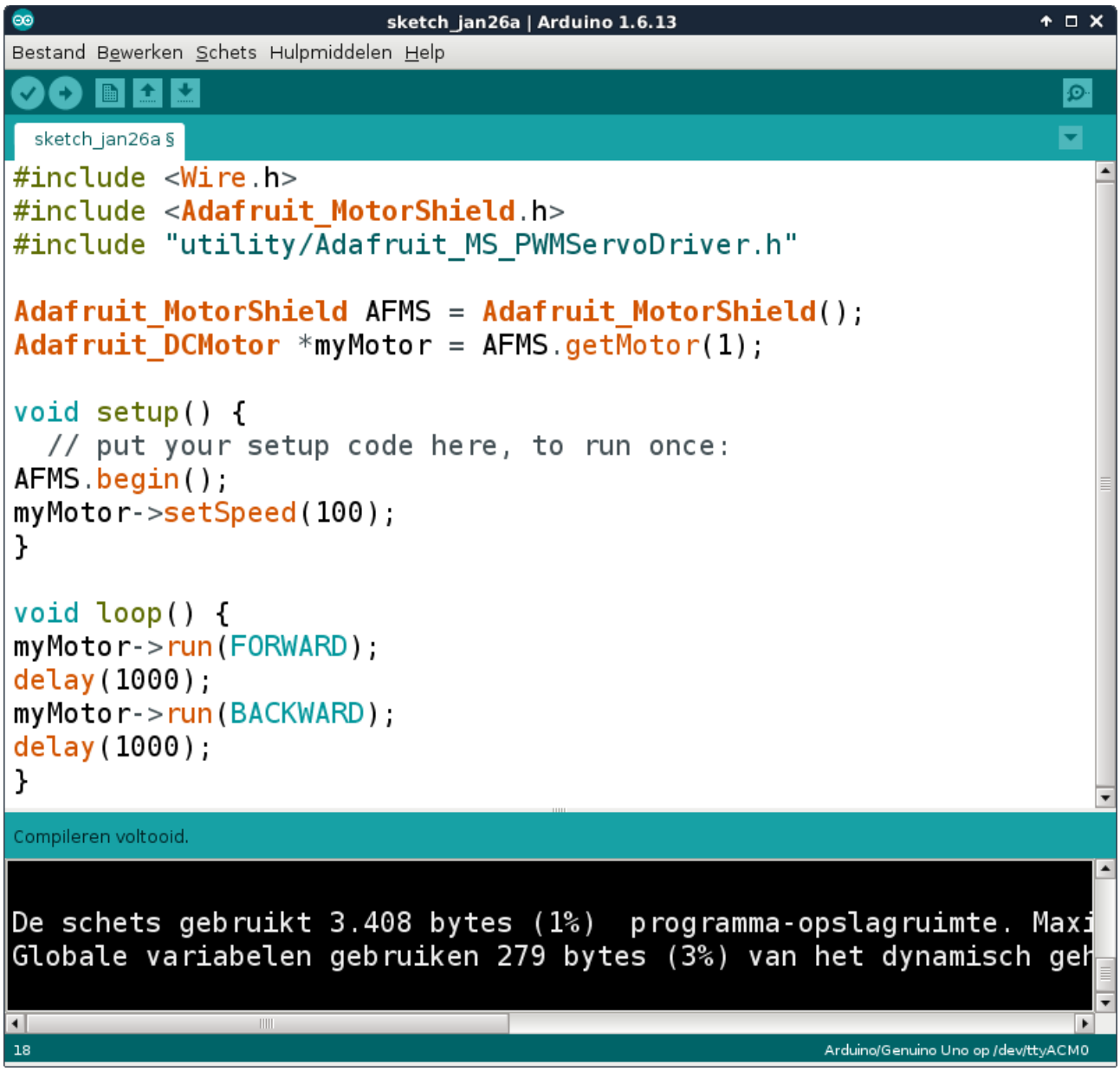
Denk er om dat je de naam gebruikt die je eerder hebt gekozen als je iets anders gebruikt dan myMotor. Je kunt voor de snelheid een waarde tussen de 0 en 255 kiezen.

Stuur je motor aan

Je kunt voor je motor de functies (FORWARD), (BACKWARD) en (RELEASE) kiezen. Plaats deze functie in het void loop() gedeelte als je wilt dat het herhaalt (denk om de naam!!). Als je motor bij de functie (FORWARD) nou niet de kant op gaat die je wilt, moet je de twee draadjes van de motor andersom aansluiten.

```
myMotor->run(FORWARD);
```

Je kunt nu een programma gaan verzinnen zodat je robot een parcours kan rijden door verschillende commando's achter elkaar te zetten en de delay() funtie te gebruiken om aan te geven hoe lang de motor aan moet blijven.


sketch_jan26a | Arduino 1.6.13
Bestand Bewerken Schets Hulpmiddelen Help
sketch_jan26a §
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);
void setup() {
// put your setup code here, to run once:
AFMS.begin();
myMotor->setSpeed(100);
}
void loop() {
myMotor->run(FORWARD);
delay(1000);
myMotor->run(BACKWARD);
delay(1000);
}
Compileren voltooid.
De schets gebruikt 3.408 bytes (1%) programma-opslagruimte. Maxi
Globale variabelen gebruiken 279 bytes (3%) van het dynamisch geh
18
Arduino/Genuino Uno op /dev/ttyACM0