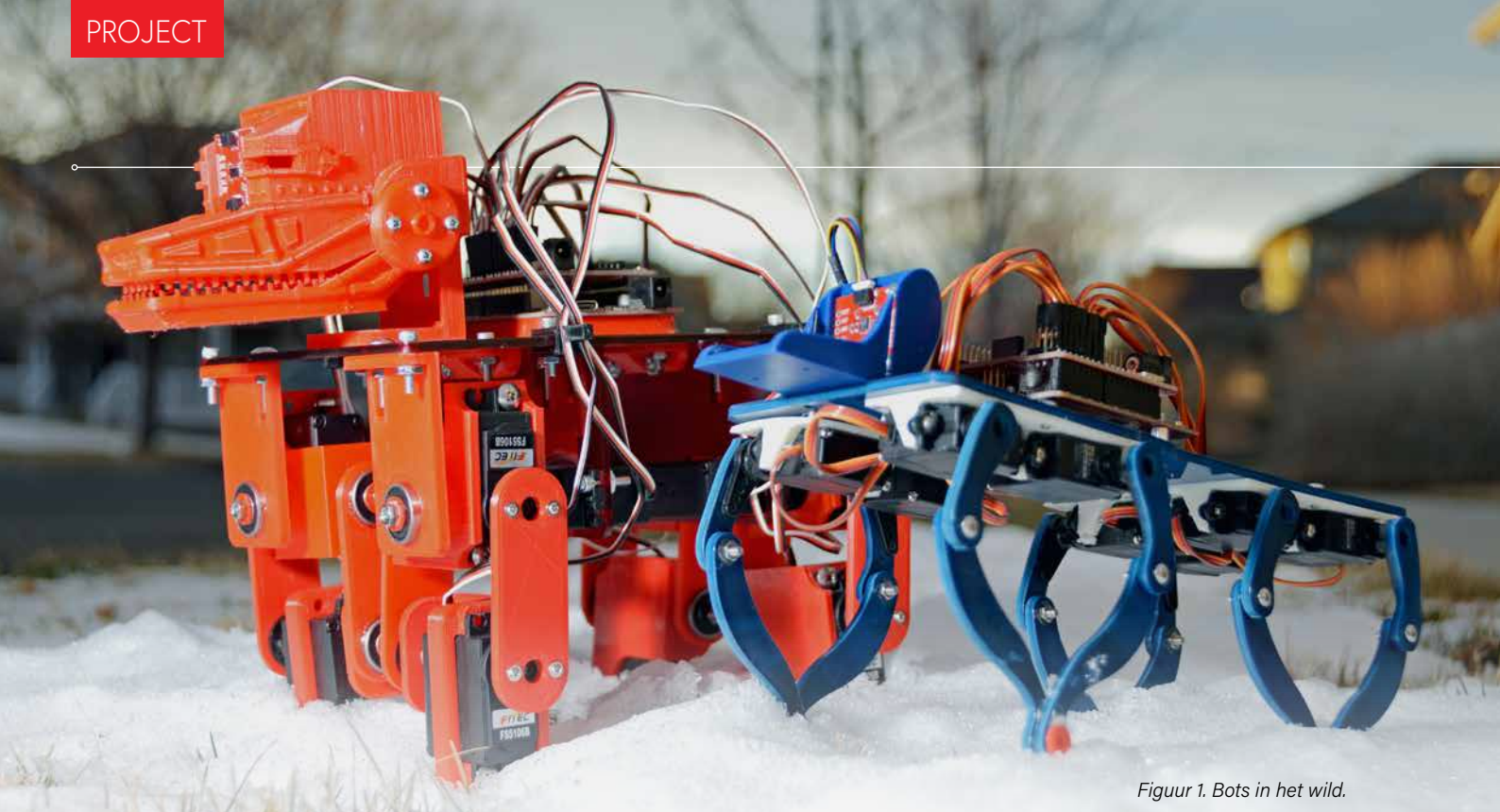


Figuur 1. Bots in het wild.

Viervoetige robots voor zelfbouw

Rob Reynolds (USA)

Natuurlijk wilt u ook een viervoetige robot bouwen! We laten hier zien hoe u twee basisplatforms kunt bouwen die elk meerdere uitsparingen hebben voor het monteren van servo's, sensoren en microcontrollers.

Ik weet niet of ik voor alle makers kan spreken, maar ik denk dat toen we in onze kindertijd aan robots dachten, we tweevoeters voor ogen hadden met interactieve vaardigheden die we nu AI zouden noemen, en misschien zelfs met alle mooie fysieke mogelijkheden van de mens. Maar dat waren dromen, of creaties in films of in onze gedachten. Wezens die tot leven werden gebracht door mensen in metalen kostuums, of verborgen poppenspelers, of genieën zoals Ray Harryhausen en zijn robot-uil Bubo uit *Clash of the Titans*. U kunt zich wellicht voorstellen dat ik nu als volwassene verbaasd en teleurgesteld ben dat mijn huishoudrobots een stofzuiger en een 'robot pâtissier' zijn, en die laatste is eigenlijk gewoon een keukenmachine die helemaal niks robotachtigs doet. Het lijkt er dan ook op dat ik maar één optie heb: ik zal mijn eigen robots moeten maken (**figuur 1**).

Het ontwerp

Een van mijn favoriete dingen bij het creëren van zoiets als een robot – of, eerlijk gezegd, elk project – is het begin. Op dat moment is alles nog mogelijk. In Stephen Sondheim's musical "Sunday in the Park with George" zegt het personage George terwijl hij zich zijn grootvader, kunstenaar George Seurat, herinnert: "Wit. Een lege pagina, of een leeg doek. Zijn favoriet. Dan is nog alles mogelijk!" Misschien weet ik gewoon niet genoeg (en hopelijk ook nooit zal weten) om te weten wat niet mogelijk zou moeten zijn, zodat in mijn gedachten alles mogelijk is.

Hoe moet mijn robot er dus uit zien? Voor de 'wow-factor' moet het misschien een tweevoeter-robot met natuurlijke vormen zijn, zoals die in de film *I, Robot*. Maar als ik iets wil dat zich gemakkelijk in mijn huis kan bewegen en met mij of zijn omgeving kan communiceren, dan is een robot op vier wielen waarschijnlijk de eenvoudigste oplossing. Maar daarmee keren we in principe weer terug naar de robotstofzuiger. Ik zoek de gulden middenweg: een beetje praktisch, maar toch indrukwekkend genoeg om mensen twee keer te laten kijken. Ik wilde er ook zeker van zijn dat ik het in de loop van de tijd zou kunnen aanpassen en verbeteren. Uiteindelijk heb ik besloten om voor een viervoeter te gaan.

Voor inspiratie hoef ik niet verder te kijken dan Boston Dynamics en hun werk met viervoeters. Van Big Dog uit 2005, groot en onhandig uitziend en over een open veld waggelend, tot Spot uit 2018, gestroomlijnd en wendbaar, deuren openend en dansend op Uptown Funk. Dat is een prachtig voorbeeld van vooruitgang

en iteratie in de tijd. Voor mijn viervoeter wilde ik een basisplatform maken waarop ik kon uitbreiden, misschien zelfs met verwisselbare modules. Ik speelde met verschillende ontwerpen en configuraties, en uiteindelijk heb ik me op twee verschillende ontwerpen gestort. Als u ze nu bekijkt, is het heel duidelijk dat de ene geïnspireerd is door Big Dog, en de andere door Spot. Ze hebben verschillende mogelijkheden, maar beide moeten kunnen worden uitgebreid.

Ik ben eerst begonnen met het ontwerpen van de kleinere en slanke Spot-achtige robot. Met een idee van de mechanica die ik wilde gebruiken, tekende ik een serie strakke rechte lijnen, resulterend in oninteressante onderdelen. Maar toen ik iets meer nadacht over de Spot van Boston Dynamics, besefte ik dat in dit geval de functie de vorm kon volgen. Ik kon me rondingen veroorloven om deze bot er beter uit te laten zien. Of dat gelukt is weet ik niet; in elk geval is hij wat ronder en structureel wat minder stabiel. Ik ben er denk ik wel in geslaagd hem er minder steriel uit te laten zien.

Daarna ben ik begonnen met de tweede robot, de meer industrieel uitziende Big Dog-geïnspireerde robot. Na enkele uren ontwerpwerk kwam het bij me op dat ik waarschijnlijk het wiel niet opnieuw hoefde uit te vinden. En inderdaad bleken er enkele ontwerpen te zijn die ik kon gebruiken, of op zijn minst als uitgangspunt kon nemen. Ik vond uiteindelijk een geweldig ontwerp van Technovation op Instructables [1].

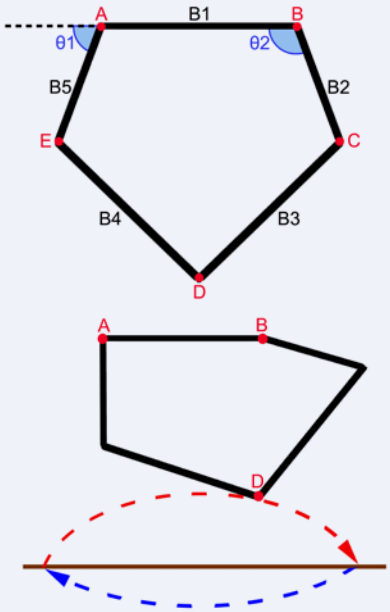
Beide platforms hebben meerdere uitsparingen voor de montage van servo's, het toevoegen van sensoren en het plaatsen van de microcontroller. Ik ben van plan om de SparkFun Redboard Artemis te gebruiken, die de bekende footprint van de Arduino Uno heeft. De Redboard Artemis biedt meer geheugen en snelheid dan de Uno en heeft BLE aan boord, en door deze footprint te gebruiken kan ik eenvoudig uitbreiden naar de SparkFun Artemis ATP als ik meer pinnen nodig heb, want ook die heeft dezelfde footprint. Daarnaast gebruik ik het SparkFun Wireless Motor Driver Shield.

De robotbenen van de viervoeter

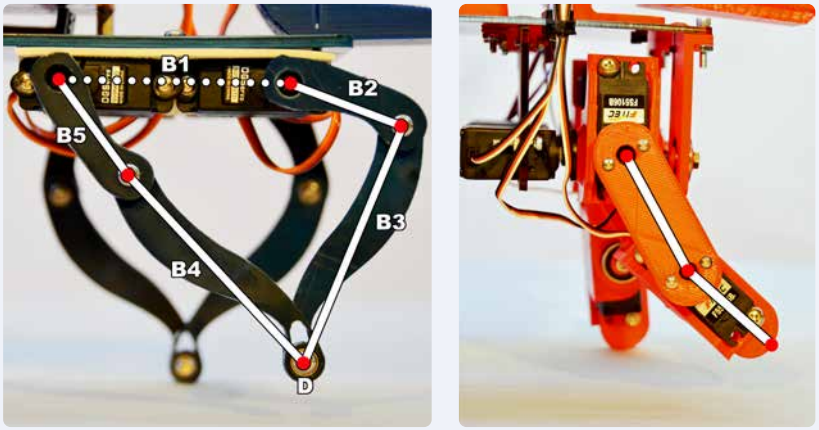
Voor de benen heb ik voor elke robot een andere keus gemaakt. Voor de kleinere robot – laten we die Bluesette noemen – heb ik een vijfarmige hefconstructie gebruikt. Dat is misschien niet de ideale constructie voor een viervoeter, zeker niet als die gebruikt wordt met servomotoren, maar de moeite waard om mee te experimenteren om kennis en inzicht in beweging en Inverse Kinematics op te doen. De vijfarmige hefconstructie is een mechanisme met twee vrijheidsgraden waarbij alle vijf de armen tot een lus zijn gekoppeld (**figuur 2**).

Dit mechanisme regelt de x- en y-coördinaten van gewricht D, het eindpunt (of in ons geval de robotvoet) door gelijktijdig de hoeken θ 1 en θ 2 aan te passen en zo de hoek van de armen B2 en B5 te regelen. Door het eindpunt van elke arm langs een elliptisch pad te bewegen, kan de robot naar voren en naar achteren lopen, zijn hoogte aanpassen en zelfs in een cirkel draaien. Zoals ik al zei is dit voor een viervoeter met servomotoren misschien niet de beste manier van voortbewegen, maar als je een set van acht krachtige borstelloze motoren gebruikt, kan deze opstelling heel effectief zijn. Een mooi voorbeeld daarvan is het Stanford Doggo Project[2]. Met de snelheid, het vermogen en de besturing van de borstelloze motoren kan deze kleine viervoeter een meter hoog springen en zelfs een salto achterover maken (**figuur 3**).

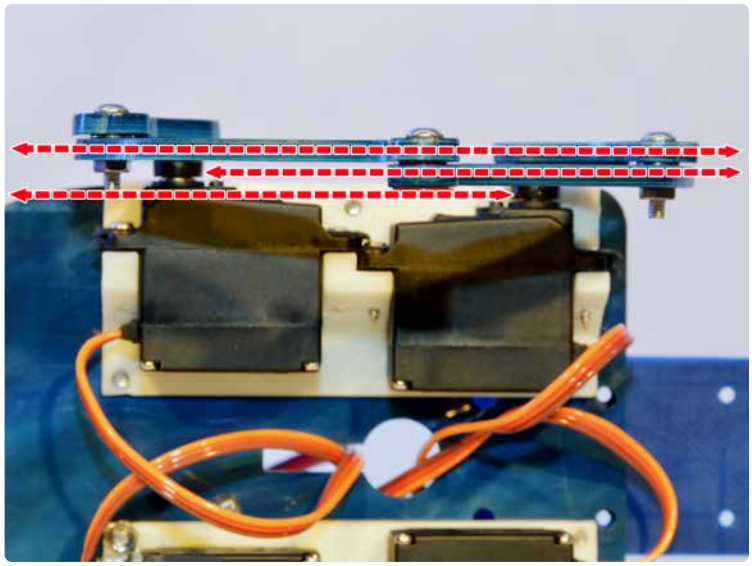
Mijn grotere en grovere robot, die ik Big Red heb genoemd, maakt gebruik van een meer gebruikelijk mechanisme voor viervoeters, bekend als de seriële manipulator. Dit is een mechanisme waarbij elk gewricht zijn eigen motor heeft, van de basis tot de eindactu-



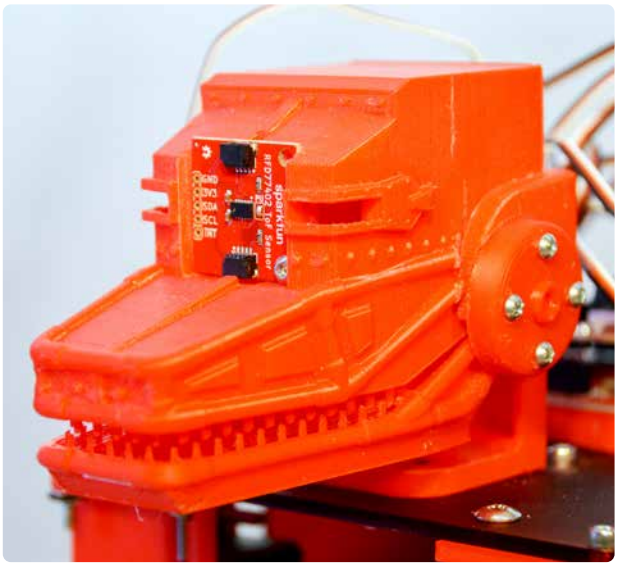
Figuur 2. Vijf-armige mechanismen.



Figuur 3. Beide systemen gebruiken twee servo's om de beenpositie te besturen, maar op heel verschillende manieren.



Figuur 4. Door bij de servobevestiging te compenseren voor de breedte van de armen beweegt alles in één vlak.



Figuur 5. Hondenkop.

ator. Deze gewrichten worden vaak aangeduid als schouder-, elleboog- en polsgewrichten, vooral bij het creëren van een antropomorfe armstructuur, zoals een éénarmige pick&place-machine. (Denk aan een auto-assemblagelijn.) Door de armen kort te houden voor deze eerste versie van Big Red, kan ik de kracht en stabiliteit toevoegen die de poten van Bluesette ontberen.

Wat betreft de constructie en de materialen heb ik gebruikgemaakt van een combinatie van lasergesneden 3mm-acryl en 3D-geprinte PLA-onderdelen. Veel plaatselijke bibliotheken en maker-ruimtes hebben 3D-printers, en sommige hebben ook lasersnijmachines. En als u geen toegang hebt tot een lasersnijder, kunt u 3mm-multiplex gebruiken, dat met handgereedschap kan worden gezaagd en geboord. De 3D-geprinte onderdelen, met name de servobevestigingen voor elke robot, zijn de belangrijkste onderdelen. Maar ook zonder toegang tot een 3D-printer kunt u met behulp van kabelbinders of hete lijm de motoren vastzetten, in ieder geval in de robot met de vijfarmige poten. Wees een beetje creatief – we zijn tenslotte engineers en makers!

Hoewel ze misschien niet helemaal nodig zijn, gebruik ik voor beide robots in alle gewrichten lagers. Het zorgt voor een veel soepeler beweging en vermindert de wrijving in de gewrichten en daardoor de stroomopname van de servomotoren. Een andere belangrijke ontwerpkenmerk van Bluesette is dat de stappenmotoren 3 mm (of de dikte van het gebruikte materiaal voor de benen) worden verschoven. Als de motoren in hetzelfde vlak worden gezet (**figuur 4**), zal het samenvoegen tot een lus resulteren in zijdelingse torsie bij een van de verbindingen, waardoor weer meer stroom nodig is om de servomotoren aan te drijven.

> *Voor mijn viervoeter wilde ik een basisplatform maken waarop ik kon uitbreiden, misschien zelfs met verwisselbare modules.*
>
> Rob Reynolds

Ik wist dat ik een soort steun voor de sensoren moest maken. Ik had natuurlijk snel en simpel een basismontagebeugel kunnen ontwerpen, maar ik wist dat ik hiermee een kans kreeg om creatief te zijn. Voor Bluesette heb ik gewoon een paar primitieve vormen in elkaar gezet, de scherpe hoeken afgerond, wat montagegaten toegevoegd en alles geprint. Aan Big Red heb ik wat meer tijd en aandacht besteed. Ik herinnerde me de film *Isle of Dogs* met een aantal echt goed uitziende robothonden, en heb aan de hand van een aantal beelden uit de film de robothondenkoppen nagemaakt, met wat aanpassingen om de meeste Qwiic-sensorboards te kunnen monteren (**figuur 5**). Ik vreesde dat deze robot daardoor aan de voorkant te zwaar zou worden, maar heb dit gecompenseerd door een groot accupakket aan de achterkant te monteren.

De hardware die ik heb gebruikt voor de montage van beide robots bestond uit M3-schroeven van verschillende lengtes (met uitzondering van de standoffs, die zijn 4-40). En ik heb enkele bussen in de lagers van Big Red en de polsgewrichten van Bluesette 3D-geprint.

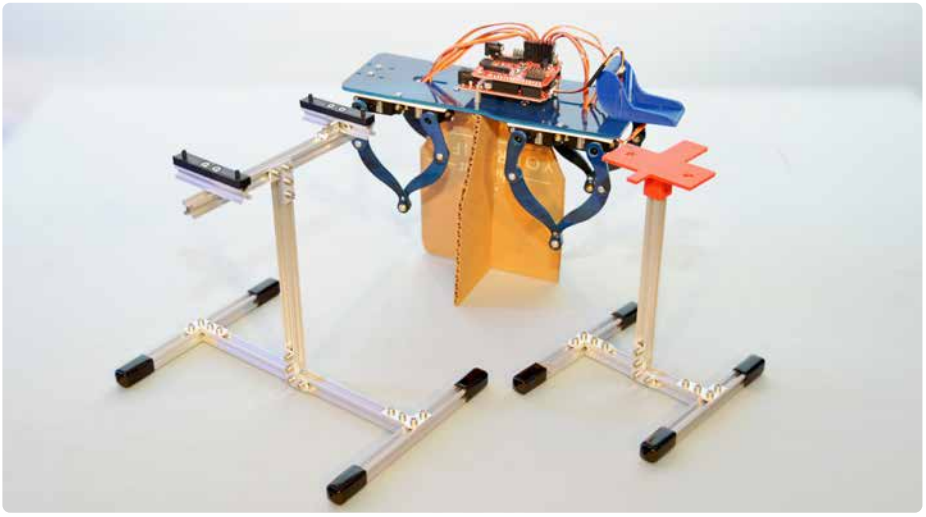
Even over coderen en testen. Als je iets ontwerpt en bouwt dat kán bewegen, gáát het ook bewegen, en meestal op onverwachte en onpassende momenten. Ik heb deze les voor het eerst geleerd tijdens het werken aan een klein autonoom voertuig. Ik had wat code aan mijn sketch toegevoegd en was deze aan het uploaden naar mijn voertuig. Op het moment dat het uploaden klaar was, ging die er als een speer vandoor, vloog van mijn werkbank en knalde tegen de muur. Nathan, de oprichter van SparkFun, had hetzelfde probleem tijdens een van zijn projecten, maar hij werkte aan een autonoom voertuig dat groot genoeg was om in te rijden. Toen die er vandoor ging, rukte hij de laptop van de werkbank. Het verdient daarom aanbeveling ervoor te zorgen dat de onderdelen die uw schepping laten bewegen, geen contact met de ondergrond hebben. Als u aan een auto werkt, voldoet een doos die iets kleiner is dan de wielbasis van het voertuig prima. Voor deze bots had ik met wat aluminium profielen en een snel aangepast 3D-onderdeel een houder geknutseld om de bots van de vloer te houden (**figuur 6**). (Als u op zoek bent naar iets dergelijks in de EU of het Verenigd Koninkrijk, denk ik dat MakerBeam de beste kansen biedt).

De code

Na de assemblage kunnen we beginnen met het schrijven van code (**listing 1**). Onze primaire resource is de servo-bibliotheek, maar naarmate we verder komen en extra mogelijkheden toevoegen, zullen er zeker andere bibliotheken in beeld komen. Om te beginnen hebben we onze sketch nodig

om de servo-objecten aan te maken en aan te sturen. Een snelle en makkelijke manier om dat te doen is door simpelweg een paar kleine aanpassingen te doen aan bijvoorbeeld de Sweep-sketch. Om het, zowel nu als in de toekomst, zo simpel mogelijk te houden kunnen we de servo's nummeren in een `for`-loop. Als u van plan bent door te gaan met deze of andere robots met een soortgelijk ontwerp is dat een goede aanpak, omdat u eenvoudig het aantal servo-objecten dat u aanmaakt en aansluit kunt aanpassen door de integer te wijzigen. Vervolgens stellen we een startpositie in voor elke servo en geven we een korte sweep-test, om er zeker van te zijn dat ze allemaal communiceren en op de juiste manier worden gevoed. Bij het werken aan code waarvan ik weet dat die zich blijft uitbreiden en misschien meerdere iteraties zal hebben, vind ik het het beste om de code op te splitsen in functies in plaats van alles in de `void:loop()` onder te brengen. Hierdoor zijn zaken makkelijker terug te vinden en te veranderen, en het maakt ook kopiëren en plakken in andere iteraties van de sketch eenvoudiger, of in compleet andere sketches die misschien slechts een klein deel van een bestaande code nodig hebben. (Misschien komt dit wel omdat ik oud genoeg ben om me het gebruik van subroutines in BASIC te herinneren).

Hiermee is het fundament gelegd. De volgende stappen, althans voor mij, zullen in oplopende moeilijkheidsgraad plaatsvinden. De hoogte van de bot halveren is eenvoudig genoeg en kan een soort van stealth-modus creëren. Vanaf daar zou de volgende logische stap een paar loopfuncties zijn, zowel voor- als achteruit. Daarna komt waarschijnlijk draaien in beide richtingen. Vervolgens kunnen we naar BLE-besturing of autonome beweging gaan, afhankelijk van wat het uiteindelijke doel is van de robot. Verbinding maken met laptop of tablet via Bluetooth en het verzenden van seriële commando's met één karakter kan een snelle en eenvoudige manier zijn om de bot op afstand te bedienen, of u zou hem kunnen koppelen met een micro:bit board voor een draagbare afstandsbediening. Als u uw robot vanaf een Android-telefoon wilt kunnen bedienen, kunt u de App Inventor (https://appinventor.mit.edu/) van MIT gebruiken om een interface te maken op een controller die u altijd bij de hand hebt. En als u niet zeker bent, kunt u altijd een functie voor elk van hen schrijven, en vervolgens die functies wegcommentariëren die u besluit niet te gebruiken. Nabijheidssensoren zijn een populaire add-on voor elke robot, en lijken het meest zinvol als volgende stap. Misschien wilt u een omgevingssensor toevoegen voor zaken als temperatuur of vochtigheid. De laatste stap voor deze keer is het toevoegen van elementaire machine learning- of AI-mogelijkheden. Als u TensorFlow draait, zal het niet moeilijk zijn om uw robot te trainen om te reageren op eenvoudige spraakcommando's – stop, ga, ga zitten, val aan of wat dan ook!



Figuur 6- Ik gebruikte twee statieven van aluminium profielen, maar u kunt net zo goed gewoon golfkarton gebruiken.

Klaar om er zelf een te bouwen?

Zoals bij de meeste van mijn projecten blijven ook deze 'werk in uitvoering' en zullen ze als deze tekst wordt gedrukt al veel verder zijn dan nu het geval is. Daarom beperk ik me nu hier tot de basisprincipes, en zet alles in een Github repository (https://github.com/ThingsRobMade/QuadrupedRobots), die ik zal blijven updaten als ik nieuwe functies aan deze twee viervoeters toevoeg. Als u overweegt er zelf een te bouwen, wacht dan niet om te zien wat ik aan het doen ben. Begin gewoon met bouwen en programmeren. De grote meerwaarde van de open-source gemeenschap is het feit dat we met z'n allen slimmer zijn dan ieder voor zich, en waar ik een knelpunt tegenkom, zal een van u ongetwijfeld met iets komen dat zo briljant is dat het me zal inspireren tot nieuwe methoden en richtingen. Tot die tijd: blijf dromen, blijf creëren, en *happy hacking*!

200712-04

WEBLINKS

[1] Technovation, "3D Printed Arduino Powered Quadruped Robot," Instructables, 2020. : https://www.instructables.com/3D-Printed-Arduino-Powered-Quadruped-Robot/

[2] Nate711, "StanfordDoggoProject," GitHub, 2019. : https://github.com/Nate711/StanfordDoggoProject

Listing 1. Dit zet de gewrichten van de robot in een neutrale positie en beweegt ze dan lichtjes heen en weer.

```
/*
* Quadrupedal Robot Setup Sketch
* Rob Reynolds
* SparkFun Electronics
*
* This simple sketch sets the joints of a
* quadrupedal robot to a neutral position,
* then swings them slighty back and forth
* simply to test control and movement.
* Currently setup for 8 DoF, but can easily
* be adjusted by changing to integer in
* Servo leg[*] to correspond to the
* number of servos.
*/

#include <Servo.h>
 Servo leg[8];   // create servo objects to control a servo
int pos = 90;    // variable to store the servo position

void setup() {
  delay(1000);
  for (int l = 0; l < 8; l++){
    leg[l].attach(l + 2); // attach servos sequencially starting with pin 2
    delay(100);
  }
  delay(1000);
  for (int l = 0; l < 8; l++){
    leg[l].write(pos); // set all servos to 90°
    delay(100);
  }
}

void loop() {
  allServoTest(); // Initial test or all servo movement
  while(1){
  }
}

void allServoTest(){
  // Initial movement test for all servos. All servos are set to 90°.
  // then swept front and back before returning to 90° once again.
   for (int i = 0; i < 8; i++){
    for (pos = 90; pos <= 135; pos += 1) {
      leg[i].write(pos);                  // tell servo to go to position in variable 'pos'
      delay(10);                          // wait 10ms for the servo to reach the position
    }
    for (pos = 135; pos >= 45; pos -= 1) {
      leg[i].write(pos);                  // tell servo to go to position in variable 'pos'
      delay(10);                          // waits 10ms for the servo to reach the position
    }
    for (pos = 45; pos <= 90; pos += 1) {
      leg[i].write(pos);                  // tell servo to go to position in variable 'pos'
      delay(10);                          // waits 10ms for the servo to reach the position
    }
  }
}
```