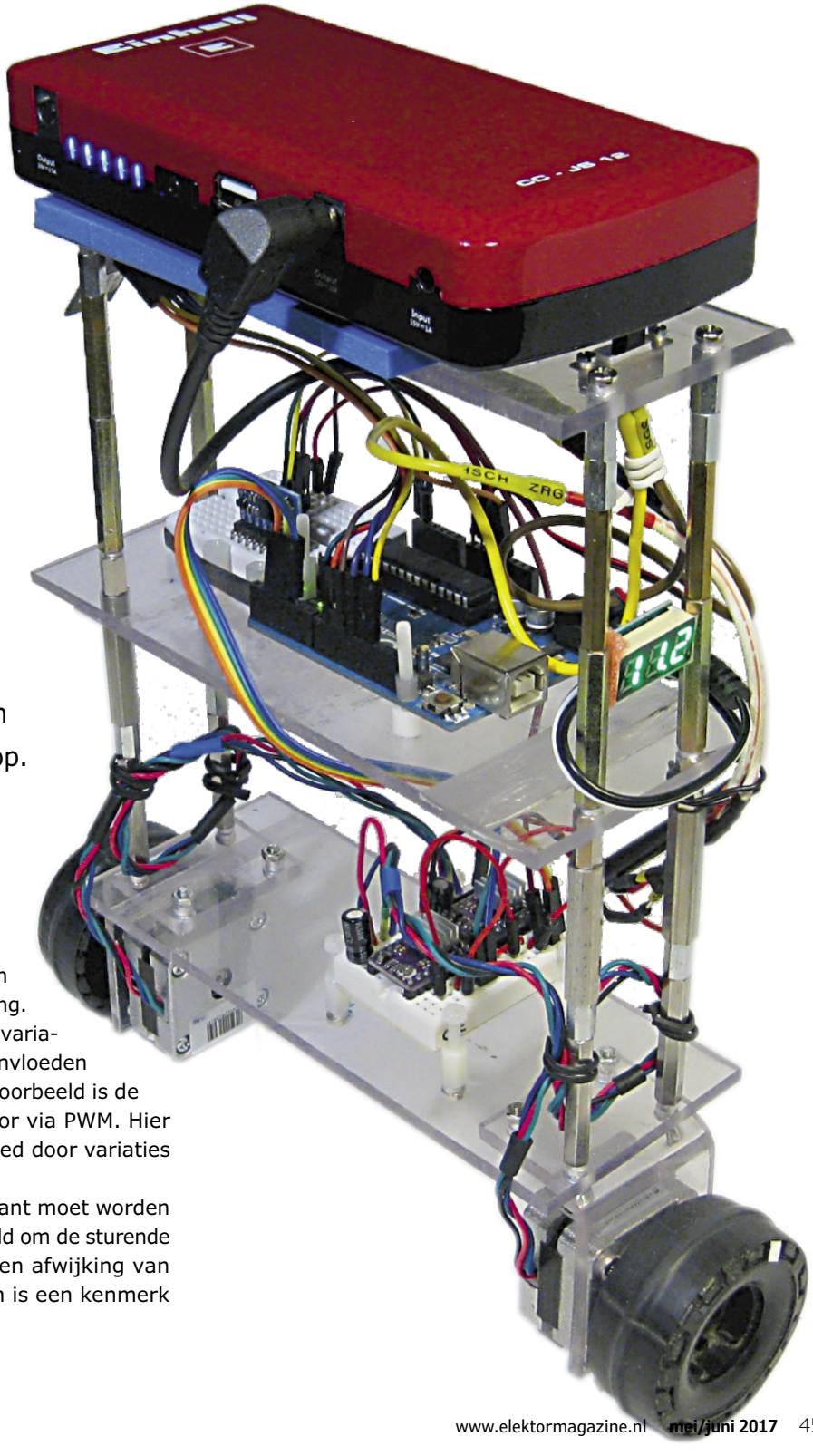
HOMELAB PROJECT

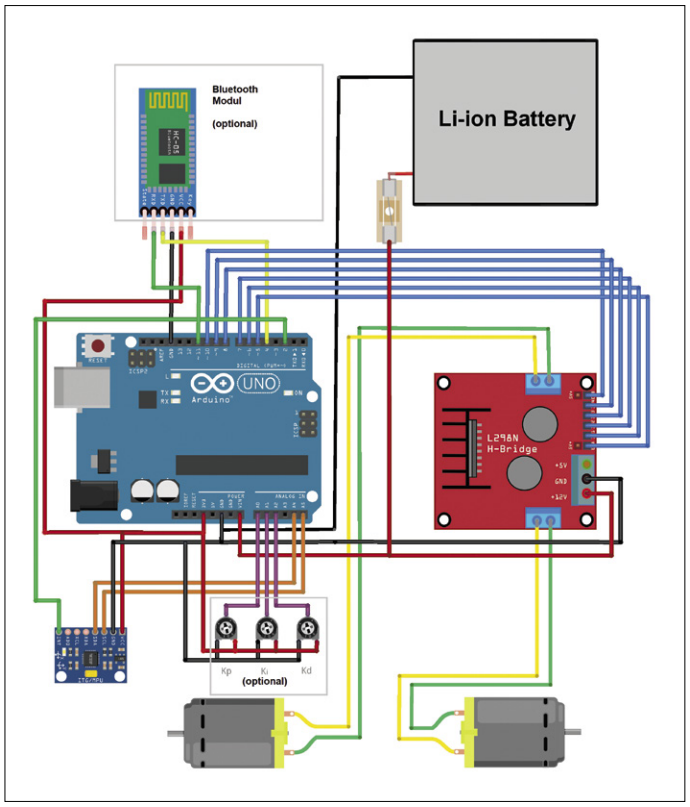
BalBot:
een zelfbalancerende robot
moderne versnellingssensoren vergemakkelijken de bouw

Dr. Günter Spanner (Duitsland)

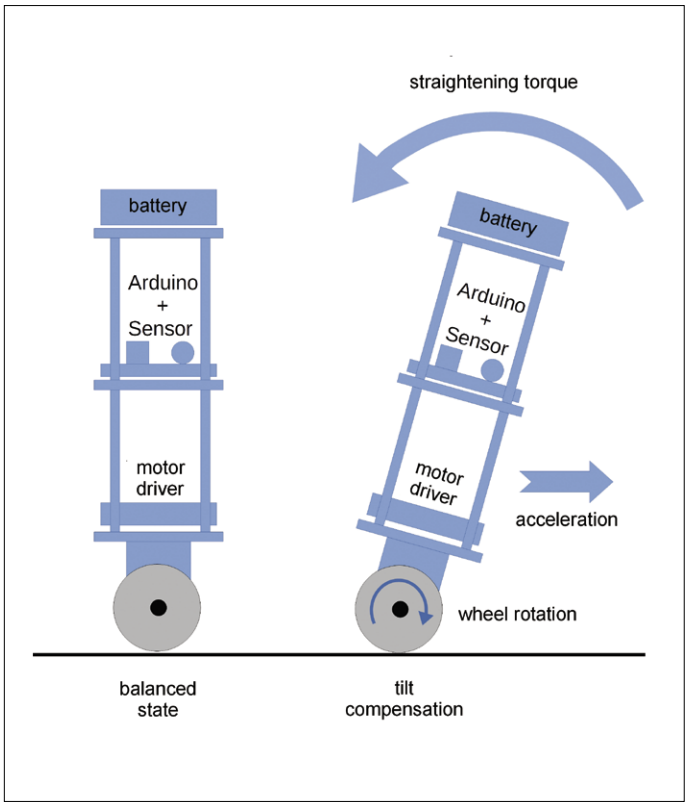


Regelsystemen komen niet alleen in de techniek voor, maar ook in de natuur. De constructie van zelfbalancerende robots kan als een eerste fase bij de ontwikkeling van androïden dienen, aangezien voortbeweging op twee (parallelle) wielen een vergelijkbare regeling vereist als rechtop lopen. In dit project wordt gebruik gemaakt van een Arduino Uno en een IC met een versnellingssensor en een gyroscoop.

In het dagelijks leven wordt, in tegenstelling tot technologie en wetenschap, vaak geen onderscheid gemaakt tussen regeling en sturing. Een eenvoudige sturing bewaakt de uitgangsvariabele niet, waardoor deze door externe stoorinvloeden kan veranderen. Een eenvoudig en typisch voorbeeld is de snelheidsregeling van een gelijkstroommotor via PWM. Hier wordt de snelheid van de motor ook beïnvloed door variaties in de belasting.
Nu naar de regeling: als een toerental constant moet worden gehouden, is terugkoppeling nodig, bijvoorbeeld om de sturende PWM- of DC-spanning aan te passen aan een afwijking van het toerental. Zo'n teruggekoppeld systeem is een kenmerk van een regelkring.



Figuur 1. Bedrading van BalBot. Bluetooth-module en potentiometers zijn optioneel.



Figuur 2. Principe van zelfbalancering.

ONDERDELENLIJST

Arduino Uno
Motordriver (-module) L298
MPU6050
2 DC-reductiemotoren, bijv. PGM37DC12
Accu's (7 NiMH of 2 LiPo)
Zekering 2 A
Aan/uit-schakelaar indien nodig

Regelen betekent dus dat de uitgangsvariabele (bijvoorbeeld snelheid) wordt gemeten en dat bij een afwijking van de gewenste waarde de stuurgrootheid (hier de PWM) navenant wordt gewijzigd. De centrale begrippen zijn:

- werkelijke waarde: x
- gewenste waarde: w
- regelafwijking: e = w – x
- stuurgrootheid: y
- storingsgrootheid: z

Deze grootheden volstaan om een regelsysteem te beschrijven. Er zijn ook regelsystemen met verschillende regelkarakteristieken. De zogenaamde PID-regelaar is zeer gangbaar en is in veel gevallen bruikbaar. Deze heeft een proportionele (P) integrerende (I) en differentiërende (D) karakteristiek. Door het aanpassen van de parameters kunnen veel regeltechnische problemen met hoge precisie worden opgelost.

Balanceren op één as

Een interessant voorbeeld voor de toepassing van dergelijke regelkringen is een zelfbalancerende robot. In principe is dit een omgekeerde slinger op wielen. Vanwege zijn instabiliteit is de omgekeerde slinger een klassiek voorbeeld van de toepassing van een actieve regelkring.
De bekende Segway-roller is een éénassig, tweewielig, elektrische transportmiddel. De te transporteren persoon staat tussen de wielen nagenoeg op de as. Eenassige voertuigen zijn in principe instabiel. Een regelsysteem moet dus voor het nodige evenwicht zorgen – dan wordt rijden leuk. De populariteit van die rollers stimuleerde de ontwikkeling van zelfbalancerende robots.
Het is niet echt moeilijk om een roller volgens het 'Segway-principe' te bouwen. Er bestaan zelfs complete kits. Het hoeft echter niet per se zo'n roller te zijn om met de onderliggende regeltechniek aan de slag te gaan: voor dit doel volstaat een kleine zelfbalancerende robot, zoals hieronder beschreven.
Om zo'n zelfbalancerende robot in evenwicht te houden, moeten zijn wielen de neiging tot kantelen voortdurend tegengaan. Hiervoor zijn een passende regelkring, een sensor en actuatoren nodig. De MPU6050 is bij uitstek geschikt als sensorelement, omdat dit IC een versnellingssensor en een gyroscoop bevat. Dit maakt een zeer nauwkeurige meting van de versnelling en rotatie langs de drie ruimtelijke assen mogelijk. Aangezien dergelijke chips in veel apparaten zoals camera's of smartphones worden gebruikt, zijn ze dankzij massaproductie zeer goedkoop. Een Arduino Uno is alles wat nodig is om de meetgegevens te verwerken. De motoren die aan de wielen zijn gekoppeld zijn de actuatoren.
Als de MPU6050 is aangesloten op een Arduino (linksonder in

figuur 1) kan de goede werking worden gecontroleerd met een testprogramma. Het download-archief bij dit artikel [1] bevat voor dit doel het programma 'MPU6050_test'. Na het laden van het testprogramma kan de seriële monitor van de Arduino IDE worden gestart. Zes lijnen tonen de verandering van de sensorwaarden bij draaiing of beweging. Als de juiste bewegingsgegevens worden weergegeven, kan de motordriver worden aangesloten op de Arduino. De L298N (het IC of een module die ermee is uitgerust) is geschikt als driver voor kleine gelijkstroommotoren. Voor de voeding zorgen zeven 1,2-V NiMH-accu's (= 8,4 V) of twee 3,7-V LiPo-accu's (= 7,4 V). Figuur 1 laat zien hoe de afzonderlijke componenten en modules (zie onderdelenlijst) met elkaar worden verbonden.

Algoritmen zorgen voor evenwicht

Zoals al opgemerkt hebben we een geschikte regelkring nodig. De positie van de robot komt overeen met de werkelijke waarde. Deze moet zo weinig mogelijk afwijken van de verticale positie (doel- of gewenste waarde). Een PID-regelaar is voor dit doel ideaal. Onze robot (die we BalBot hebben gedoopt) bereikt een verbazingwekkend stabiel dynamisch evenwicht met optimaal aangepaste Kp-, Ki- en Kd-parameters. De doelwaarde (de verticaal) wordt eerst in de software vastgelegd. **Figuur 2** illustreert hoe dit in de praktijk werkt.

De MPU6050-module registreert permanent de momentele stand van de robot. De meetwaarden worden verwerkt door het PID-algoritme in de Arduino. De regel-sketch voert de berekeningen uit voor de aansturing van de motoren. Als de robot in één richting dreigt te kantelen, wordt het betreffende kantelmoment gecompenseerd door een overeenkomstige compenserende beweging van de wielen. De robot blijft dan overeind, zelfs als we hem een duwtje zouden geven. De bijbehorende sketch BalBot.ino is ook opgenomen in het gratis download-archief.

De code is bestaat uit zes delen:

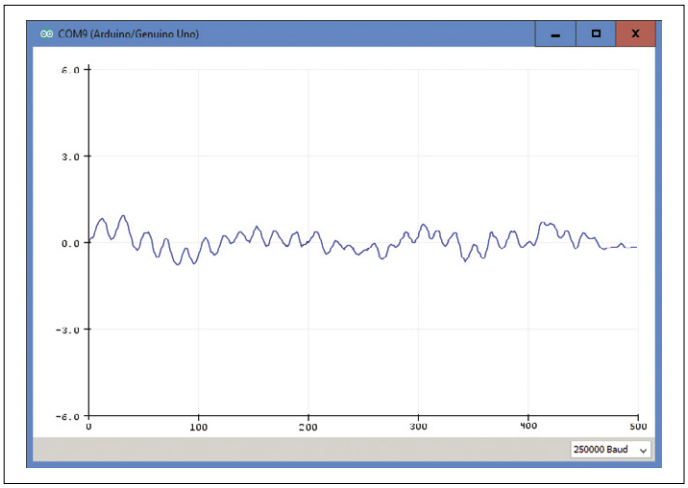
1. Eerst worden de bibliotheken geladen. Indien niet opgenomen in de standaard Arduino IDE, kunnen ze worden gedownload [2][3]. Vervolgens worden de motoraansluitingen vastgelegd. Via EN (enable) en IN (input) kunnen de motoren zowel vooruit als achteruit draaien. Het motorvermogen wordt geregeld via PWM.
2. De voor de werking van de MPU6050 vereiste waarden worden daarna vastgelegd. Dit zijn in wezen de standaardinstellingen. Details zijn te vinden in de datasheet van het IC [4].
3. Nu worden de PID-parameters vastgelegd. Deze zijn van eminent belang. De volgende drie waarden definiëren het regelgedrag: Kp (bereik 0...1000; standaardwaarde 400), Kd (bereik 0...100; standaardwaarde 30) en Ki (bereik 0...500; standaardwaarde 200).
4. Deze waarden moeten worden aangepast aan de reële omstandigheden van de betreffende apparatuur. Verschillen in mechanische constructie (door verschillende motoren, wielmaten enzovoort) kunnen hier worden gecompenseerd. Aanwijzingen voor het aanpassen van deze parameters komen verderop nog aan de orde.
5. Tijdens de setup worden nu de vereiste interfaces geactiveerd. De seriële interface wordt alleen gebruikt om debug-informatie uit te voeren. Deze code kan worden weggelaten in de 'definitieve versie'. Aansluitend wordt de MPU6050-module gestart en wordt de werking gecontroleerd – indien nodig wordt er een foutmelding gegeven. De opgegeven offsetwaarden kunnen meestal zonder wijziging worden overgenomen, aangezien de gyroscoop zichzelf automatisch kalibreert.
6. In de hoofdlus wordt alleen het MPU-signaal uitgelezen en wordt het stuursignaal voor de motoren berekend via de PID-regelaar `pid.Compute()` en doorgestuurd naar de motordriver met de volgende instructie:

```
motorController.move(PIDout, MIN_ABS_SPEED);
```

Ook wordt gecontroleerd of de FIFO-buffer van de MPU6050 overloopt. Dit zou bij normaal gebruik niet mogen gebeuren. Indien nodig kan de grootte van de buffer worden aangepast. De regel:

```
Serial.println(ypr[1] * 180/M_PI); // actual tilt
```

geeft de kantelhoek aan. In de seriële plotter van de Arduino IDE kan deze continu worden gevolgd. Als er een regelmatig, sinusvormig verloop verschijnt, is dit een duidelijke indicatie dat de regeling oscilleert. In dit geval moeten de regelparameters worden aangepast. In de meeste gevallen helpt het om de Kp-waarde te verkleinen. Idealiter zou de lijn alleen kleine, onregelmatige uitslagen mogen vertonen, zoals die welke worden veroorzaakt door willekeurige verstoringen (luchtbewegingen of een hobbelige vloer). **Figuur 3** toont deze toevallige compensatiebewegingen rond de rustpositie.



Figuur 3. Verloop van de kantelhoek in de seriële plotter van de Arduino IDE.

Optimaliseren en afregeling van de parameters

De optimalisatie van de PID-waarden kan ook in een simulatie plaatsvinden, bijvoorbeeld met MATLAB. In de professionele sector is deze procedure standaard voor complexe regelingen. Maar met BalBot kan ook de klassieke 'trial and error' benadering worden gebruikt. Hier worden de afzonderlijke parameters geleidelijk aangepast tot de best mogelijke regeling is bereikt:

1. Eerst worden de drie parameters Kp, Ki en Kd op nul gezet.
2. Dan wordt Kp in kleine stapjes verhoogd. Als Kp te laag

is, zal de robot kantelen omdat de correcties niet snel of niet groot genoeg zijn. Als Kp te hoog is, zal de robot wild heen-en-weer zwaaien zonder het evenwicht te bereiken. Kp moet zo worden afgesteld dat de robot slechts licht heen en weer beweegt.

3. Rechtop balanceren is hiermee een feit. Vaak is er nog een zekere neiging tot oscilleren: de robot schommelt of trilt snel rond de ideale evenwichtspositie.
4. Nu wordt de Kd-waarde aangepast zodat de oscillaties worden gereduceerd. In het ideale geval blijft de robot nu vrijwel roerloos rechtop staan. Als Kp en Kd goed zijn afgesteld, blijft de robot rechtop staan, zelfs als hij een zetje krijgt.
5. Ten slotte wordt de Ki-parameter ingesteld. Zelfs bij een optimale Kp is er nog steeds een zekere neiging tot oscilleren bij grote verstoringen; de robot kan zelfs zijn evenwicht verliezen. Ki is optimaal ingesteld wanneer de robot na een verstoring snel weer de evenwichtspositie bereikt.

Als alle drie de parameters optimaal zijn ingesteld, behoudt de robot een bijna verticale positie zodra het vangbereik van de regeling is bereikt. Hij staat dan rechtop alsof hij door een spookachtige hand wordt vastgehouden en compenseert met gemak zelfs grotere verstoringen.
Als er geen evenwicht kan worden bereikt, is het mogelijk dat de motoraansluitingen zijn verwisseld. In dit geval versnelt de robot niet in de richting van het kantelmoment, maar in de tegenovergestelde richting. **Figuur 4** toont de afgewerkte robot in een rechtopstaande, zelfbalancerende positie.
Omdat het instellen van de parameters met behulp van software nogal ingewikkeld is, is er ook een eenvoudigere variant. Een Arduino heeft verschillende analoge ingangen. Hierop kunnen drie potentiometers worden aangesloten, waarmee de waarden voor P, I en D direct kunnen worden ingesteld. De software hoeft alleen de potmeterwaarden in te lezen (zie **Listing Potmeters**).
Zo kunnen de parameters tijdens bedrijf worden geoptimaliseerd via de potmeters (zie *BalBot_Pots_BT.ino* in de download). De aansluiting van de optionele potentiometers is in figuur 1 al meegenomen.

Listing Potmeters.

```
void readPIDTuningValues()
{   int potKp = analogRead(A0);
    int potKi = analogRead(A1);
    int potKd = analogRead(A2);
    kp = map(potKp, 0, 1023, MIN_P, MAX_P) / 100.0;
    ki = map(potKi, 0, 1023, MIN_I, MAX_I) / 100.0;
    kd = map(potKd, 0, 1023, MIN_D, MAX_D) / 100.0;
}
```

Listing Bluetooth.

```
void readPIDTuningValues()
{   int potKp = analogRead(A0);
    int potKi = analogRead(A1);
    int potKd = analogRead(A2);
    kp = map(potKp, 0, 1023, MIN_P, MAX_P) / 100.0;
    ki = map(potKi, 0, 1023, MIN_I, MAX_I) / 100.0;
    kd = map(potKd, 0, 1023, MIN_D, MAX_D) / 100.0;
}

content=blue.readString();

#if LOG_BT
     Serial.print("content: ");Serial.print(content);
     Serial.print(" - setpoint: "); Serial.print(setpoint);
     Serial.print(" - rotation: "); Serial.println(rotate);
#endif

if(content[0]=='F')
       // forward
else if(content[0]=='B')
       // backward
else if(content[0]=='L')
       // left
else if(content[0]=='R')
       // right
```

Afstandsbediening via Smartphone

Het bouwen van een zelfbalancerende robot is een interessante uitdaging. Als alles werkt, dan is dat in eerste instantie fijne resultaat op de lange termijn niet meer zo spannend. Een afstandsbediening kan dat veranderen. Hiermee kan BalBot draadloos alle kanten opgestuurd kunnen worden. Aangezien iedereen (en zeker elke elektronicus) wel een smartphone met een Bluetooth-interface heeft en veel daarvan onder Android draaien, gebruiken we zo'n smartphone gewoon als afstandsbediening van de robot.
We hoeven slechts een Bluetooth-module op de Arduino aan te sluiten. Een HC05/06-receiver is zeer geschikt (die is in figuur 1 al ingetekend).
Om de module te gebruiken moet de standaard Bluetooth-bibliotheek van de Arduino IDE worden geïntegreerd en moet de sketch navenant worden aangevuld (**Listing Bluetooth**).
Deze uitbreiding is ook al opgenomen in de sketch *BalBot_Pots_BT.ino*. De smartphone heeft dan alleen de 'Arduino Bluetooth'-app van Circuit Magic nodig. Die kan gratis worden gedownload van de Playstore [5]. Na installatie en zodra er verbinding is met de HC05/06-module, stuurt de app de benodigde stuurcommando's naar de

Arduino. **Figuur 5** toont de gebruikersinterface van de app, die wordt bediend met de pijltjestoetsen aan de linkerkant van het scherm.

Conclusie en vooruitblik

De hier gepresenteerde robot kan vrij eenvoudig in elkaar worden gezet. Naast de aandrijfmotoren zijn alleen een Arduino, een motordriver en een versnellings-/gyroscoopmodule nodig. Het instellen van de regelparameters wordt door enkele extra potmeters sterk vereenvoudigd. BalBot kan zelfs via Bluetooth afstandsbediend door uw woning bewegen.

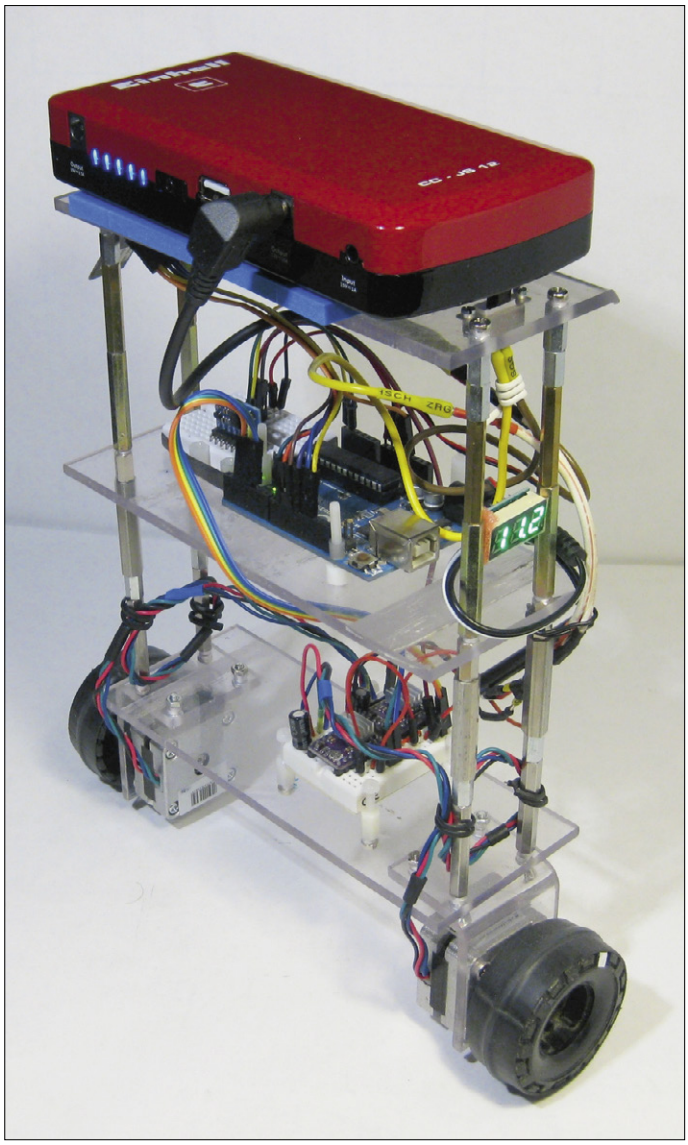


Foto 4. BalBot in actie.

Figuur 5. Android-app voor het besturen van BalBot via Bluetooth.

Een verdere verbetering van het evenwicht en gedrag, met name bij afstandsbediening, zou kunnen worden bereikt met draai-encoders. Hiermee kan de draaisnelheid van de wielen worden geregistreerd. Deze informatie kan worden gebruikt om de stabiliteit bij het voor- en achteruit rijden verder te verbeteren. In de praktijk verliest de op afstand bediende BalBot af en toe zijn evenwicht als alleen de waarden van de MPU6050 beschikbaar zijn. Helaas kan deze sensor alleen versnelling en rotatie direct meten, maar niet de snelheid.
Een andere mogelijkheid is het gebruik van stappenmotoren. Het tellen van de stappen levert informatie op die vergelijkbaar is met de informatie van draai-encoders. Naast het vervangen van de motoren moet de L298 natuurlijk ook worden vervangen door geschikte stappenmotordrivers (zoals de DRV8825-driver). In het boek “Robotik und Künstliche Intelligenz” van de auteur treft u nog veel meer informatie en roboticaprojecten aan (zie kader).

200074-04

Weblinks

[1] Projectpagina bij dit artikel: www.elektormagazine.nl/200074-04
[2] PID_v1.h: https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h
[3] MPU6050_6Axis_MotionApps20.h: https://bit.ly/37T7T1c
[4] Datasheet MPU6050: www.invensense.com/products/motion-tracking/6-axis/mpu-6050/
[5] “Arduino Bluetooth”-app: https://play.google.com/store/apps/details?id=com.circuitmagic.arduinobluetooth&hl=de