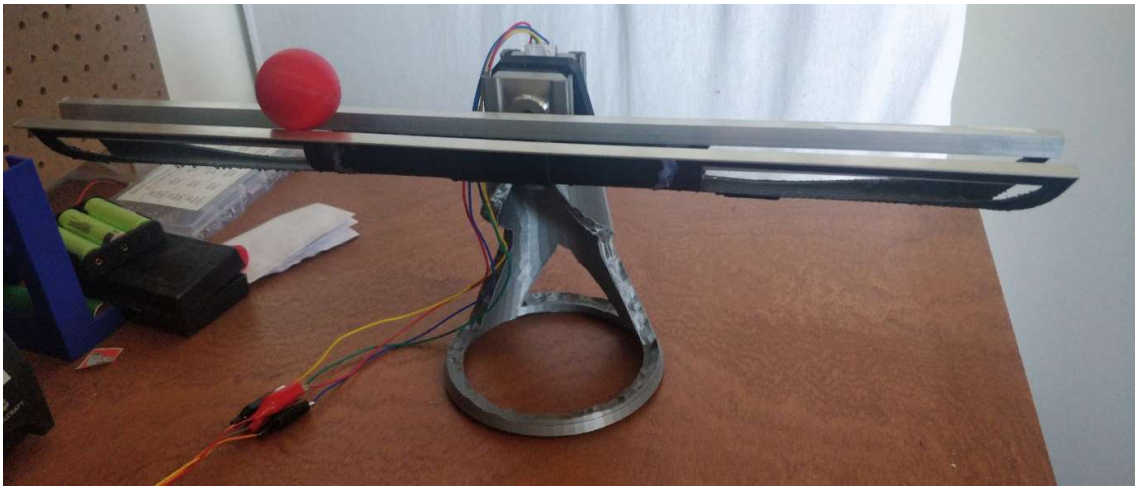
Computer Vision Bar balance

Niels Post (1740627)



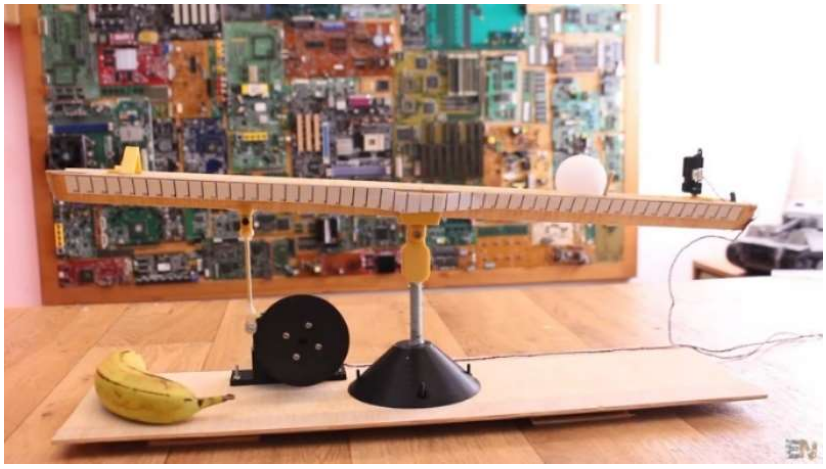
Inhoud

0. Inleiding

In dit TPD (Technisch Product Dossier) licht ik mijn inlevering in voor de eindopdracht voor het vak MRB, wat deel is van de opleidingsrichting Technische Informatica van de opleiding HBO-ICT aan de Hogeschool Utrecht. De docenten van dit vak zijn Marius Versteegen en Bart Bozon.

Het doel van deze eindopdracht is het bouwen van een zelfregelend systeem, gebruikmakend van hardware-onderdelen en digitale filtertechnieken.

Voor deze opdracht heb ik gekozen om een balanceerbalk te maken. Dit is in feite een goot waar een balletje doorheen rolt (zie afbeelding). Het doel is hierbij voor het systeem om te zorgen dat de bal op een specifieke plek stil blijft liggen.

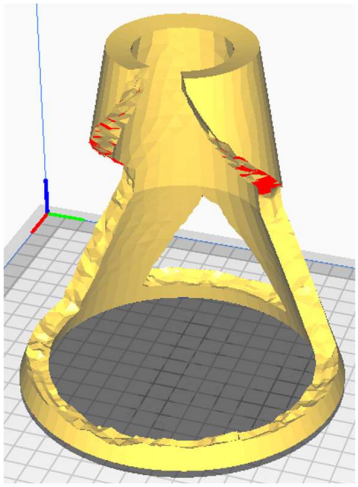


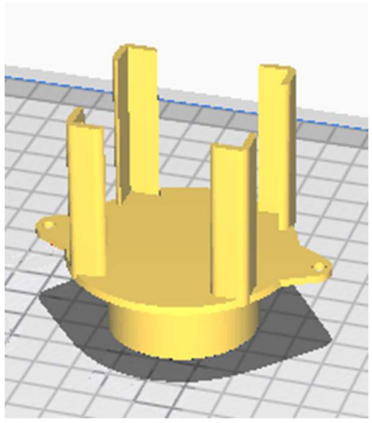
1. Hardware

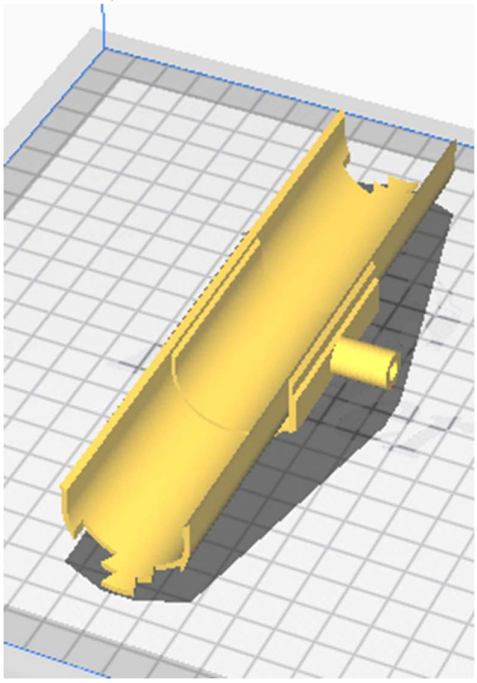
1.1. Balk

Voor het opbouwen van de balk maak ik voornamelijk gebruik van 3D printing. Hiervoor zijn een aantal onderdelen die in elkaar passen om de opstelling te vormen. Van links naar rechts (niet op schaal): *Een basis om op tafel te plaatsen, Een houder voor de motor, het middelste deel van de balanceerbalk.*

De motorhouder past in de basis, en de motor zelf bevestigt dan aan de balanceerbalk. De balanceerbalk bestaat uit 3 delen, die van te voren aan elkaar gelijmd kunnen worden. Om de bal soepeler te laten rollen bevestig ik aluminium hoekprofielen op de balk (zie productfoto's).







1.2. Sensoren

Voor het meten van de positie van de bal maak ik geen gebruik van standaard hardware sensoren, maar van een webcam.

Afwegingen

Vaak worden er voor dit soort balanceerbalken sensoren gebruikt die een afstand meten. Een nadeel hiervan vind ik dat deze aan het einde van de balk bevestigd worden, en dat deze daardoor de bal blokkeren wanneer deze naar de rand rolt.

Het leek mij een interessante uitdaging om een systeem te maken waarbij de bal van de balk af kan rollen. Hierdoor is het essentieel dat het systeem zo min mogelijk overshoot heeft, om te voorkomen dat de bal werkelijk van de balk afrolt.

1.3. Actuatoren

Om de balk te kunnen draaien maak ik gebruik van een NEMA17-steppermotor die ik bevestig op het midden van de balanceerbalk. Deze stuur ik digitaal aan met een TMC2209 stepper driver.

Afwegingen

Voor een systeem als dit zijn er voor de actuatoren twee gebruikelijke keuzes:

- Servo/servomotor
- Stepper motor

Voor de actuator heb ik gekozen voor wat ik in huis had. Voor mijn stage had ik een aantal van deze steppermotoren gebruikt voor mijn project. Hierdoor had ik ook al ervaring met deze motoren, wat hielp bij het aansturen.

1.4. Aansturing

Alle software van het systeem draait op een Jetson Nano. Dit is een ontwikkelboard van NVIDIA wat veel weg heeft van een Raspberry Pi. Dit komt goed uit, aangezien ik de digitale pinnen hiervan nodig heb om de steppermotor aan te sturen.

Een groot voordeel van de Jetson Nano is dat deze in vergelijking met de Raspberry Pi een sterkere processor heeft, wat er voor zorgt dat deze erg goed is in het verwerken van live beelden.

Omdat de pin-libraries en de pin-layout van de Nano en de Pi compatible zijn zou dit project volledig moeten draaien op een Raspberry Pi. Dit heb ik alleen niet kunnen testen, waardoor ik ook niet weet wat de performance dan is.

1.5. Interface

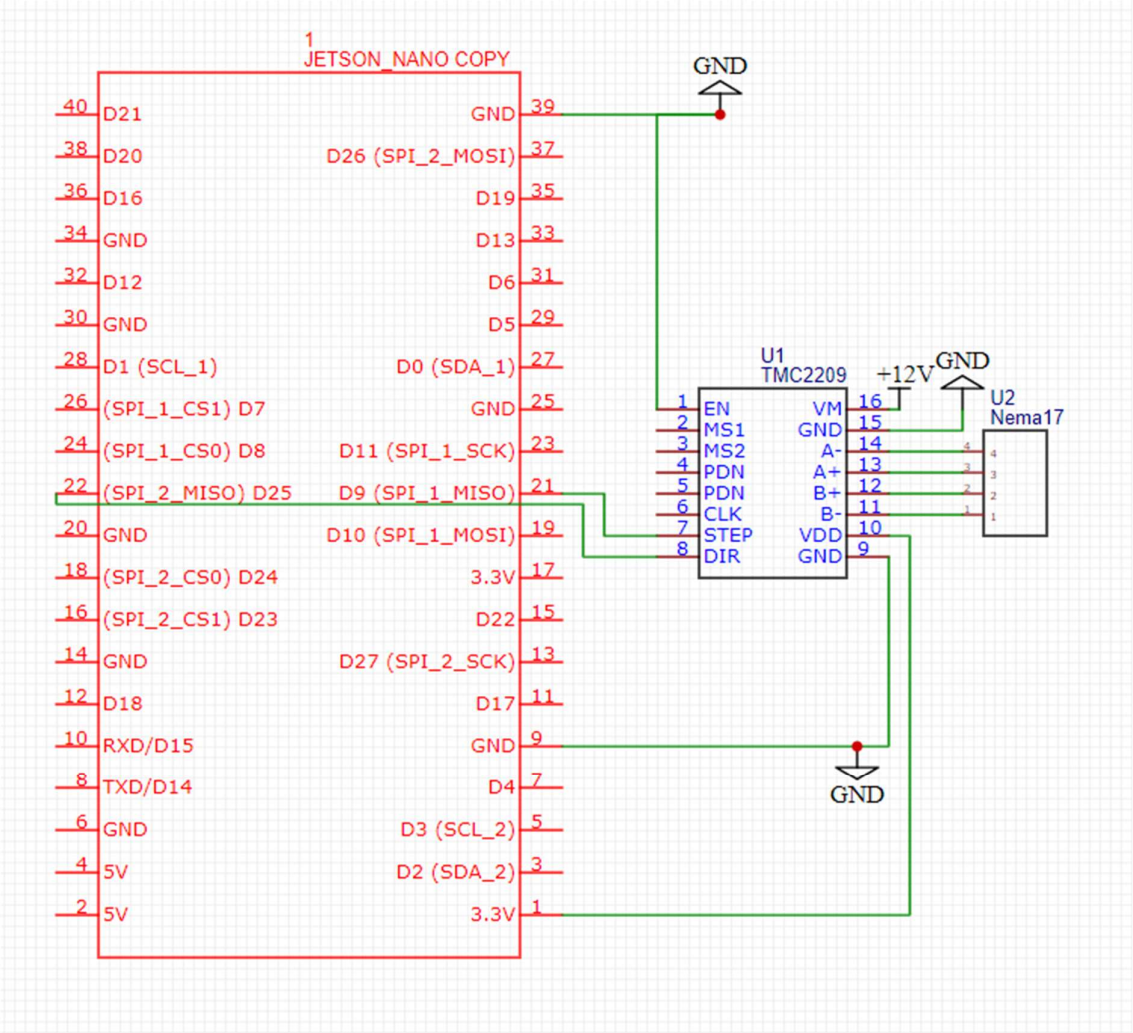
Omdat ik zowel voor de sensor als voor de actuator een digitale techniek heb gekozen is er niet erg veel te zeggen over de interfacetechniek.

Webcam

De webcam die ik gebruik sluit aan via de USB-poort van de Jetson Nano. Omdat ik gebruik maak van OpenCV zullen de meeste USB-webcams ondersteund zijn. Ook kan er dan gebruik gemaakt worden van de specifieke webcampoort die op de Nano zit (net als bij de Raspberry Pi)

Steppermotor

Zoals eerder genoemd sluit ik de steppermotor aan op de Nano via een TMC2209 steppermotor driver. Het aansluitdiagram hiervoor is als volgt:



Bij het aansturen van de motordriver gebruikt de Nano één pin om de snelheid van stappen aan te geven (STEP) en één om de richting aan te geven (DIR). Bij beiden gaat het om een digitaal signaal met een logic level van 3.3V. Omdat er bij het aansturen van de motor door de TMC2209 gebruik gemaakt word van de aangesloten 12V voeding hoeven deze signalen geen hoge stroom te dragen.

2. Software

Alle code voor dit project is te vinden op https://github.com/Niels-Post/CVBarBalance.

2.1. Libraries

De code maakt gebruik van 3 externe libraries:

- **OpenCV-** Computer Vision library. Een aantal functies hiervan gebruik ik in het detectieproces van de bal.
- **Numpy-** Standaard wetenschappelijke library. Deze gebruik ik voor een aantal wiskundige operaties in het berekenen van de bal-positie
- **RPi.GPIO-** Library voor Raspberry Pi om digital pins aan te sturen. Deze gebruik ik om de motoren aan te sturen

Verder heb ik zelf klassen geschreven voor de volgende onderdelen

- **Stepper motoren-** Een simpele interface om de steppermotoren via de TMC2209 driver aan te sturen.
- **PID-** Een klasse om PID berekeningen uit te voeren zonder precieze kennis nodig te hebben van hoe PID werkt
- **MovingAverage-** Een datastructuur om een moving average te berekenen, gebaseerd op een FIFO-queue.

2.2. Filters

Tijdens de normale werking van het project gebruik ik 2 filtertechnieken om de data te verwerken.

Kleurenfilter

Om de bal te vinden zoek ik naar pixels met een kleur die dicht bij de kleur van de bal liggen. Hiervoor gebruik ik een HSV-representatie van het camerabeeld. Uit dit beeld filter ik alle pixels weg die hier niet in de buurt liggen. Om dit te doen bepaal ik per kleurdimensie (h, S en V) welke waarde de bal ongeveer zou moeten hebben. Voor elke kleurdimensie bepaal ik aan de hand daarvan een gebied waar de waarde zich mag bevinden. In de praktijk werkt het in mijn ervaring het best om de gebieden van H en S klein te houden, waar het gebied voor V groot mag zijn.

Met OpenCV zet ik alle pixels waarvan de waarden buiten dit gebied vallen op 0, en alle andere pixels op 1. Hierna neem ik de gemiddelde positie van alle pixels die 1 zijn om de positie van de bal te vinden. Mochten er dan nog andere pixels op het beeld zijn die de juiste kleur hebben, worden deze door de gemiddelde-berekening ook verworpen.

Positie van de bal

Wanneer ik de positie van de bal zonder filtertechniek zou verwerken zou deze erge last hebben van ruis. Deze ruis kan worden veroorzaakt doordat er ruis in het camerabeeld zit, maar ook bijvoorbeeld doordat de tafel waar de opstelling opstaat licht heen en weer beweegt.

Om deze reden pas ik, na het berekenen van de positie van de bal een moving-average toe. Hierbij pas ik apart een moving average toe op de x en op de y coordinaat. Na experimenteren heb ik gekozen voor een moving-average met een lengte van 3. Dit is lang genoeg zodat de ruis adequaat weggewerkt wordt, maar niet zo lang dat er een significante vertraging optreedt in de uitvoer.

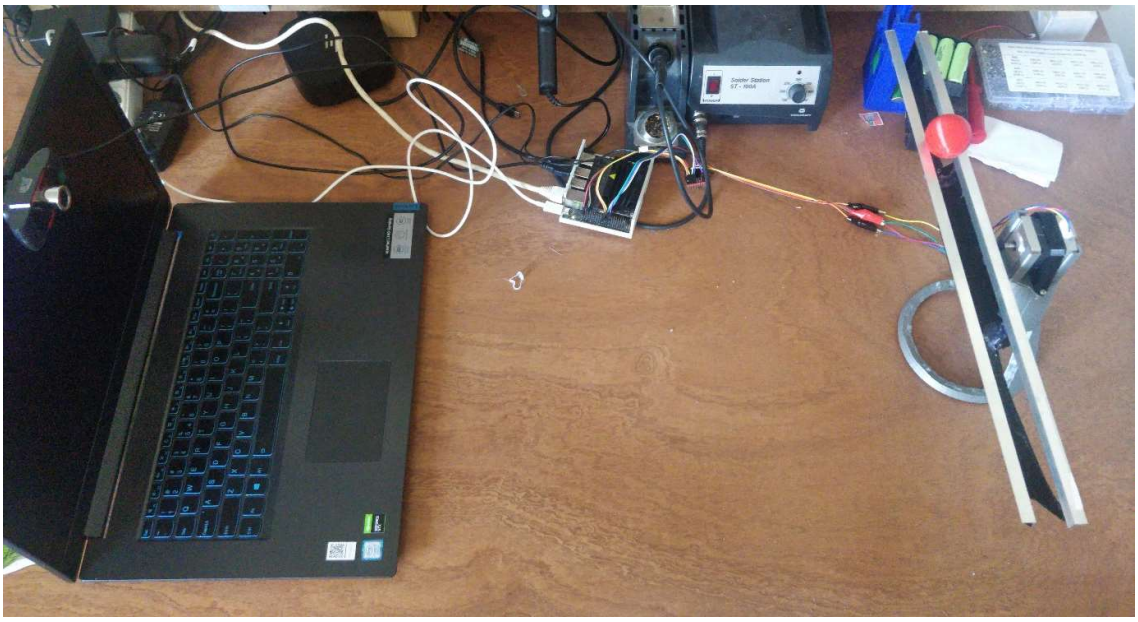
2.3. PID

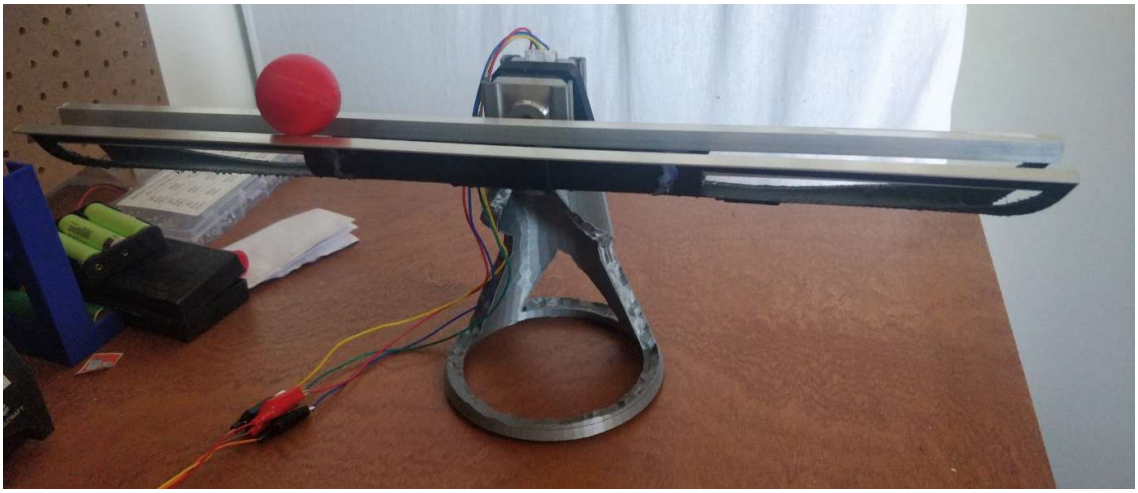
Om de PID-waarden van het systeem ingesteld ben ik vooral gaan uitproberen welke waarden het beste werkten. Hiervoor heb ik de volgende ideeën gebruikt:

- Een I-waarde kan ik het beste als laatste toevoegen, zolang ik de balk goed recht instel tijdens het testen. Ik verwacht niet dat deze op een andere manier kan ontstaan dan een scheefstaande balk
- Ik begin met instellen op de P waarde. Hierbij laat ik de I- en D-waarden nog op 0 staan.
- Bij het testen van P controleer ik wat de meest extreme stuuractie zou zijn; De bal staat volledig links en het setpoint volledig rechts. Als dit een stuuractie geeft die te extreem is om te herstellen staat P te hoog. Ik denk dat de ideale hoek ongeveer 45 gaden zou zijn. Hierbij houdt de bal altijd nog goed contact met de balk
- Ook controleer ik een gemiddelde/lage stuuractie. Als de P-waarde te klein is kan het zijn dat het balletje nooit gaat rollen wanneer deze niet erg ver hoeft te gaan. Zolang de bal op zo'n kleine afstand begint met rollen is de P goed. Zodra dit niet gebeurt is de P-waarde te laag.
- Bij het testen van D controleer ik het vermogen van het systeem om de bal stil te leggen. Hiervoor schakel ik de P-waarde op 0. Het setpoint laat ik op midden staan, en ik geef de bal vanaf de zijkant een tik naar het midden toe. Als de D-waarde goed is moet de bal bijna stil te komen liggen, of in ieder geval flink tegengehouden worden. In de praktijk kan de P-waarde het dan weer overnemen.
- Hierna schakel ik de P en D allebei in op de gevonden waarde. Vanaf daar zou ik met kleine aanpassingen de juiste waarden moeten kunnen vinden. Hiervoor observeer ik hetzelfde gedrag als hiervoor beschreven, en maak aanpassingen waar nodig.

3. Eindresultaat

3.1. Foto's





3.2. Video werking

https://github.com/Niels-Post/CVBarBalance/raw/master/Oplevering/Demo.mkv

3.3. Performance

Zie video.

Regeling aan het einde van het project

Al met al ben ik erg tevreden met de regeling van de positie van de bal. In de video is te zien dat de bal steeds redelijk snel en met weinig overshoot bij de positie komt. Een probleem wat nog te zien is is dat de positie niet altijd precies wordt bereikt. De bal blijft nog wel eens steken vlak naast de positie.

Ik vermoed dat dit deels ligt aan de imperfecties van de bal. Omdat ik ook de bal met een 3d-printer heb gemaakt, zitten er “ribbels” op de bal, waardoor deze minder soepel kan rollen dan bijvoorbeeld een pingpongbal.

Soepel rollen

Eerder in het project rolde het balletje nog minder soepel. Hierbij bleef de bal steeds haken achter stukken van de balanceerbalk. Om dit op te lossen heb ik advies gevraagd aan Bart Bozon. Hij heeft mij aangeraden om een aluminium lint/staaf langs de balk te lijmen om de bal te geleiden. Hiervoor heb ik uiteindelijk kleine aluminium hoekprofielen gebruikt. Hierna werkte het systeem een stuk beter.

Een verschil van dag en nacht

Tijdens het testen merkte ik dat het systeem niet altijd even precies werkt, en soms nogal schokkerig bewoog. Dit bleek te komen omdat de bal ’s nachts een donkerdere kleur had dan overdag, waardoor de bal niet goed werd gedetecteerd. Om dit op te lossen heb ik het gebied van de V-waarde groter gemaakt. Hierna werkte dit ’s avonds ook beter.

4. Schema’s

Omdat de schema’s nogal groot zijn zet ik deze niet allemaal in het verslag. Deze zijn te vinden in de volgende links:

https://github.com/Niels-Post/CVBarBalance/raw/master/Oplevering/Aansluitdiagram.png

https://github.com/Niels-Post/CVBarBalance/raw/master/Oplevering/MRB-Flowchart.png