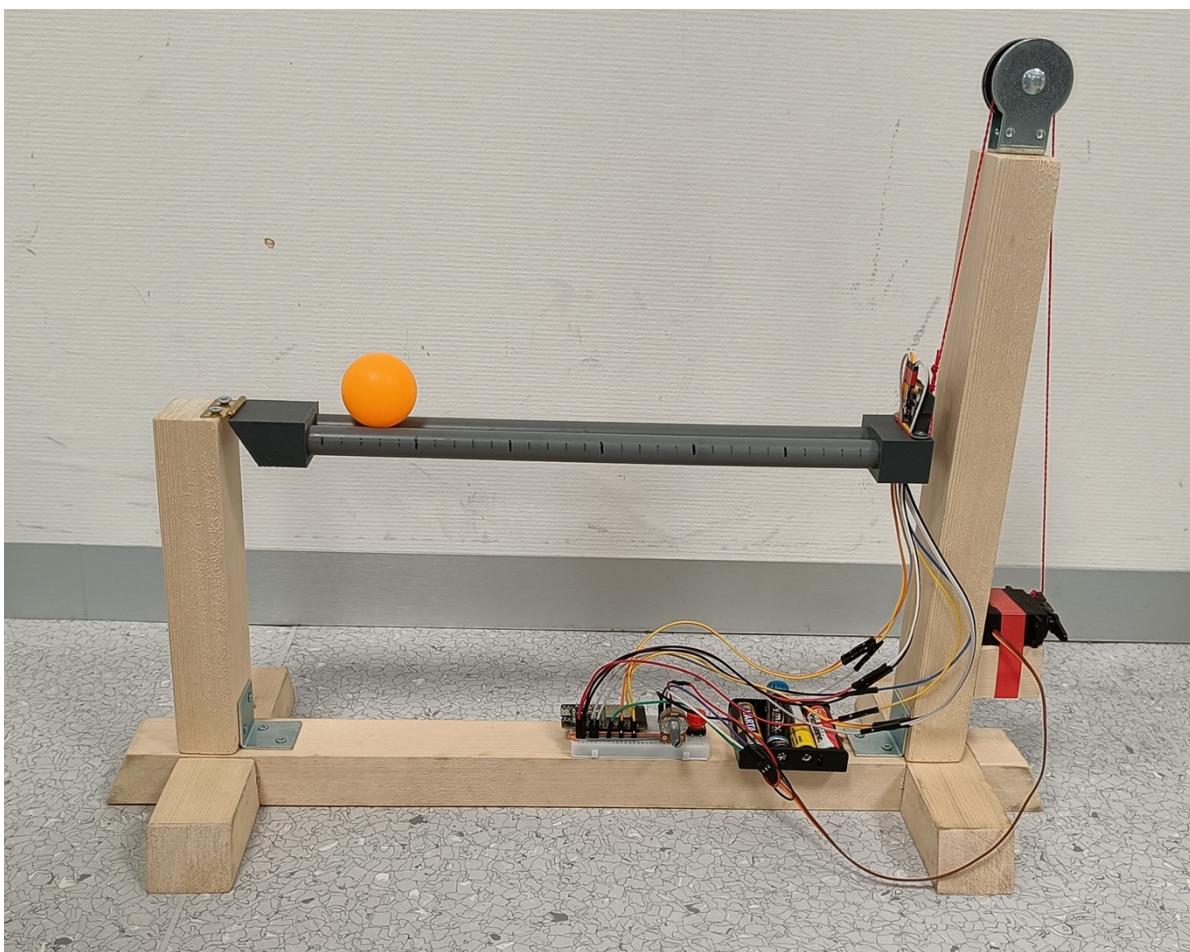


Technisch Product Dossier

Door Daphne Annink

Voor mijn MRB-eindtoets heb ik besloten om een project te realiseren waarbij ik een bal wil laten balanceren op buizen. Dit technisch dossier biedt een gedetailleerde beschrijving van mijn aanpak en de stappen die ik heb genomen om dit te realiseren.



Figuur 1

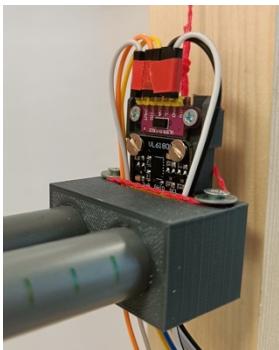
Inhoudsopgave

Ontwerpkeuzes	3
<i>Bouw</i>	3
<i>Sensoren</i>	3
<i>Actuatoren</i>	4
Filters	5
<i>Moving avarage</i>	5
<i>Exponentieel moving avarage</i>	5
<i>Conclusie</i>	5
PID-setting.....	5
Performance.....	6
Schema's	7
<i>Hardware schema's</i>	7
<i>Softwareschema's</i>	8

Ontwerpkeuzes



Figuur 2



Figuur 3

Bouw

Mijn doel voor de bouw was om een solide basis te bouwen die zo min mogelijk verstoring creëert bij het balanceren van de bal. Om dit te bereiken, heb ik gebruikgemaakt van een houten frame met deuvels, houtlijm, hoekankers, keepverbindingen en schroeven voor de constructie. Daarnaast heb ik gekozen voor een balk die aan één kant een scharnier heeft en aan de andere kant de stand /hoek in graden kan bepalen (zie figuur 1).

Mijn voorkeur ging uit naar een balk met een vast punt aan een zijkant in plaats van het midden. Dit ontwerp biedt meer controle en stabiliteit. De balk zelf bestaat uit twee PVC buizen waar een balletje tussen valt. Om de buizen bij elkaar te houden, heb ik twee 3D-geprinte en zelf gemodelleerde stukken gebruikt (zie figuur 2 & 3). Het stuk aan de rechterkant van de balk bevat tevens houders voor de sensoren (zie figuur 3).

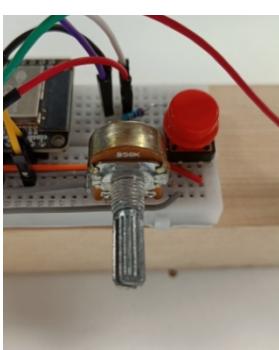
Boven op de lange staander heb ik een katrol geplaatst om ervoor te zorgen dat het draad waar de balk aan hangt zo min mogelijk weerstand ondervindt tijdens het bewegen.



Figuur 4

Sensoren

Voor het meten van de positie van de bal heb ik gebruik gemaakt van de VL6180X en VL53LOX time-of-flight sensoren (zie figuur 4). Na onderzoek bleken deze sensoren de meest nauwkeurige afstandssensoren te zijn die beschikbaar waren in het lab. Bovendien had ik nog geen ervaring met deze sensoren en leek het me interessant om iets nieuws uit te proberen.



Figuur 5

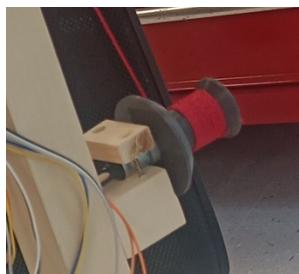
In eerste instantie maakte ik alleen gebruik van de VL6180X-sensor. Volgens de datasheet kon deze sensor metingen verrichten tot 60 cm afstand, maar na enkele tests ontdekte ik dat de nauwkeurigheid beperkt was tot slechts 20 cm en dat de metingen trager werden naarmate de afstand groter werd dan 10 cm. Daarom moest ik op zoek naar een goede vervanging of aanvulling voor deze sensor. Er was echter geen enkele sensor die zowel goed kon meten op korte afstanden (0 tot 10 cm) als op langere afstanden (10 tot 30 cm). Om deze reden besloot ik om twee sensoren te gebruiken en voegde ik de VL53LOX-sensor toe. De VL53LOX-sensor is nauwkeurig in het bereik van 10 tot ongeveer 60 cm, wat ruim voldoende is voor mijn toepassing.

Nu rees de vraag hoe ik deze twee sensoren goed kon laten samenwerken. Om te beginnen heb ik beide sensoren gekalibreerd door te controleren of de gemeten afstanden overeenkwamen met de werkelijke afstanden. Dit bleek niet het geval te zijn. De VL6180X-sensor vertoonde een standaardafwijking, die ik eenvoudig kon corrigeren door bij elke meting de afwijking te compenseren. De VL53LOX-sensor daarentegen vertoonde niet alleen

een standaardafwijking, maar ook een toenemende afwijking naarmate de afstand groter werd. Om dit op te lossen, paste ik na elke meting een berekening toe om deze afwijking te corrigeren.

Nu alle waarden nauwkeurig waren, besloot ik de langeafstands-sensor als hoofdsensor te gebruiken. Deze sensor kon immers de korte afstanden wel meten, zij het niet erg nauwkeurig. Hierdoor kon ik bepalen wanneer de bal zich in het korte gebied bevond, en op dat moment gebruik te maken van de korteafstands-sensor om de metingen uit te voeren en de waarde van de langeafstands-sensor te vervangen. Op deze manier heb ik uiteindelijk de afstand van de bal bepaalt.

Verder kun je het setpoint instellen met de potentiometer (zie figuur 5) en met de rode knop kan u de integraal (I) weer op 0 zetten. De potentiometer helemaal linksom is 28 cm van het begin en helemaal rechtsom is 2 cm van het begin. De potentiometer ergens er tussen zetten zorgt voor een setpoint ergens lineair ertussenin. Het begin ligt aan de kant van de sensoren en het 0 punt is het punt waar de balk start. Op dit punt begint de meetlat die op de balk getekend is ook.



Figuur 6

Actuatoren

In eerste instantie heb ik gebruik gemaakt van een 5V DC-motor in combinatie met een H-brug om de balk te laten bewegen. Deze motor was gemonteerd op de achterkant van het houten frame, met behulp van een 3D-geprinte spoel (zie figuur 5). Na het uitvoeren van enkele tests, kwam ik erachter dat de motor voldoende vermogen had, maar niet geschikt was voor langzamere rotaties. De motor draaide te snel, waardoor het moeilijk was om de beweging van de balk gecontroleerd te regelen. Vanwege deze beperking ben ik op zoek gegaan naar een alternatieve oplossing.



Figuur 7

Uiteindelijk heb ik gekozen voor een 180 graden servo (zie figuur 6) om de beweging van de balk te regelen. Om ervoor te zorgen dat de balk een voldoende grote hoek kon bereiken, heb ik de arm van de servo verlengd. Ik heb het touw naar de balk precies lang genoeg gemaakt dat de balk waterpas stond wanneer de servo in een hoek van 90 graden staat. Het is belangrijk op te merken dat de servo niet direct een even grote verandering in hoek van de balk veroorzaakt wanneer de stuurbeweging twee keer zo groot is. Om dit te compenseren, heb ik de juiste hoek voor de servo berekend met behulp van sinusfuncties.

```
degrees -= (asin((35 * stuuractie)/35)) * (180.0/3.141592653);
```

Hierdoor beweegt de hoek van de balk lineair in overeenstemming met de stuurbeweging.

Filters

Voor het filteren van de binnen komende data heb ik twee filters toegepast. De moving avarage en de exponentieel moving avarage.

Moving avarage

Om deze binnenkomende meetwaarden goed te kunnen gebruiken wil je ze eerst filteren ze kunnen namelijk ruis of onregelmatigheden bevatten, wat kan leiden tot ongewenste schommelingen in het regelsignaal. Om dit te verminderen heb ik een moving average filter toegepast. Dit filter berekent het gemiddelde van een bepaald aantal recente meetwaarden waardoor de invloed van korte termijn fluctuaties verminderd. Hierdoor ontstaat een meer stabiel en betrouwbaar regelsignaal dat minder gevoelig is voor ruis en andere onregelmatigheden. Ik heb meerder test gedaan om te kijken hoeveel oude waardes je mee wilt nemen voor een stabiel resultaat. Bij mij kwamen de beste resultaten bij 12 waardes. Zo bleef het systeem snel genoeg reageren op verandering maar nam het ruis zo min mogelijk mee.

Exponentieel moving avarage

Het EMA-filter (Exponentieel moving avarage) is een variant van het moving average filter waarbij recente metingen meer gewicht krijgen dan oudere metingen. Dit zorgt ervoor dat het resulterende signaal sneller reageert op veranderingen in de metingen dan het traditionele moving average filter. Ik heb deze ook toegepast maar dit zorgde ervoor dat de ruis net te veel het regelsysteem verstoerde.

Conclusie

Uiteindelijk heb ik gekozen voor een combinatie van beide filters, het EMA-filter en het moving average-filter. Beide filters hebben evenveel gewicht gekregen in mijn systeem. Door deze combinatie kon ik profiteren van de snelle respons op veranderingen die het EMA-filter biedt, terwijl tegelijkertijd de ruis sterk werd verminderd door zowel het EMA-filter als het moving average-filter.

PID-setting

Om te beginnen heb ik gekeken naar de tijdsduur van een volledige loop, omdat deze gegevens nodig waren om de tijdstap (dt) te bepalen. Na het instellen van de dt ben ik begonnen met het iteratief implementeren van de proportionele term (P) door middel van trial-and-error. Vervolgens heb ik de differentiële term (D) geïmplementeerd, aangezien deze een significantere invloed had op het systeem dan de integrale term (I).

Na het implementeren van de P- en D-termen functioneerde het systeem al zeer goed. Het enige probleem was dat het af en toe net naast het ingestelde doel (setpoint) bleef hangen en niet voldoende stuuractie gaf om de bal in beweging te brengen. Dit was een constante fout, en daarom heb ik een kleine I-term (integraal) toegevoegd om dit te corrigeren.

Uiteindelijk heb ik de volgende instellingen gebruikt voor mijn PID-regeling:

```
const float kp = 0.003;  
const float ki = 0.0002;
```

```
const float kd = 0.011;  
const float dt = 0.1;
```

Door middel van experimenteren en finetunen heb ik deze instellingen geoptimaliseerd om een nauwkeurige en stabiele regeling van het systeem te bereiken. Om de instellingen voor de integrale term (I) te bepalen, heb ik ook geëxperimenteerd om te bepalen bij welke waarde ik de term wil laten terugkeren naar nul. Ik heb het systeem laten draaien en geobserveerd hoe hoog de integrale term opliep. Over het algemeen kwam deze term nooit hoger dan 1000. Dus, wanneer de waarde een keer hoger werd dan 1200, bijvoorbeeld als de bal niet aanwezig was of tijdelijk werd vastgehouden, werd de integrale term gereset naar nul. Dit zorgde ervoor dat de integrale term beheerst bleef en niet ongecontroleerd zou blijven toenemen in dergelijke situaties.

Performance

Het balanssysteem presteert over het algemeen goed bij het naderen van het setpoint. Het kan de bal binnen ongeveer 4 seconden stilleggen binnen een afstand van ongeveer 1 cm van het setpoint. Het systeem voert ongeveer 18,5 loops per seconde uit, wat betekent dat het snel reageert op veranderingen en nauwkeurige aanpassingen kan maken.

Echter, het systeem ondervindt soms moeilijkheden wanneer de bal een zeer licht zwaartepunt heeft en langzaam beweegt. In dergelijke situaties kan de bal moeite hebben met rollen en kan de prestatie van het systeem worden beïnvloed. Om de prestaties van het balanssysteem verder te verbeteren, zijn er enkele mogelijke

Verbeteringen die nog overwogen kunnen worden:

1. Sensorfusie: Overweeg het gebruik van meerder lange afstand sensoren, de lange afstand sensor zorgde namelijk voor de meeste ruis en hier kan nog op gewonnen worden.
2. Mechanische aanpassingen: Als de bal moeite heeft met rollen bij langzame snelheden, kan er overwogen worden om het ontwerp van het systeem aan te passen of bijvoorbeeld een andere bal te gebruiken.
3. Geavanceerde regelstrategieën: Als het balanssysteem complexer wordt, kan er gekeken worden naar geavanceerdere regelstrategieën, zoals modelgebaseerde regeling of adaptieve regeling.

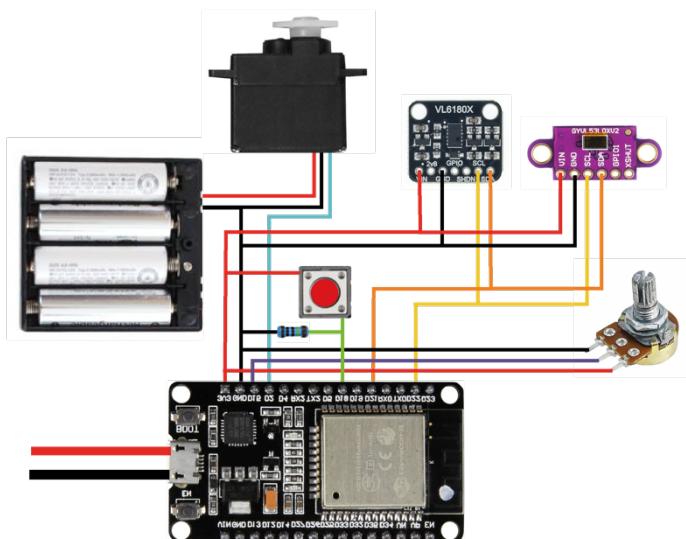
Samengevat vertoont het balanssysteem goede prestaties. Er zijn mogelijkheden om de systeemprestaties verder te verbeteren, zowel op het gebied van regeltechniek als ontwerp, Maar voor de MRB-opdracht ben ik tevreden met het resultaat.

Schema's

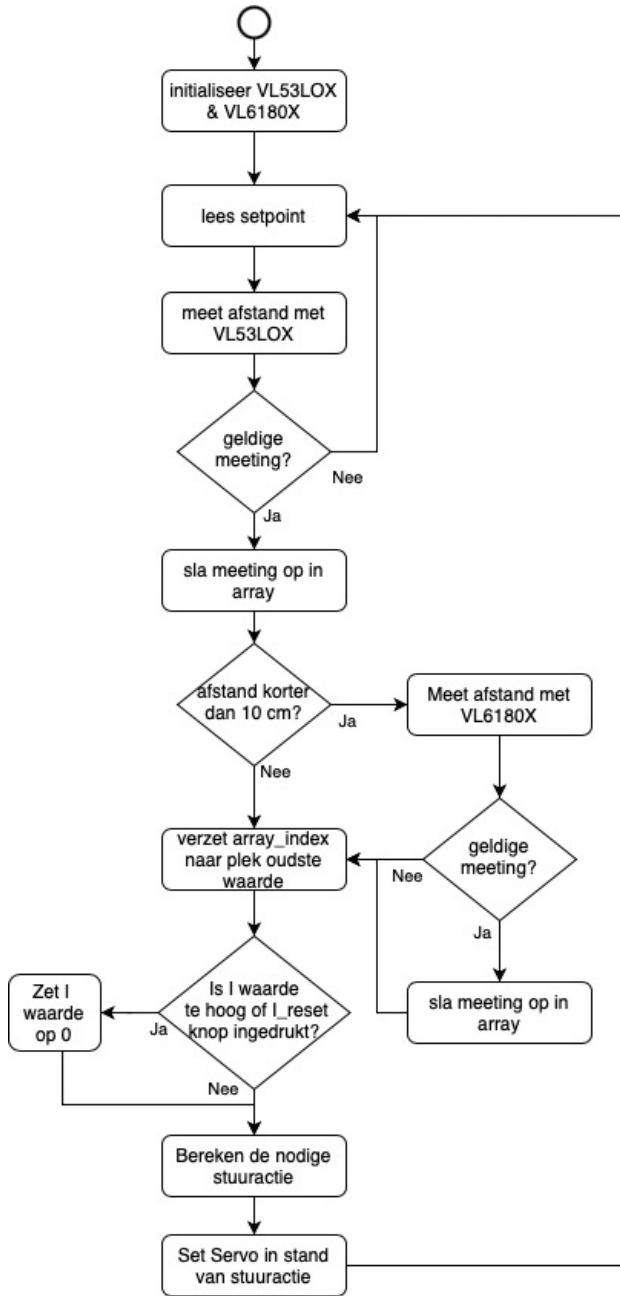
Hardware schema's

Mijn project bevat verschillende hardwarecomponenten. Hieronder (zie figuur 8) kunt u zien hoe deze met elkaar zijn verbonden. Ik轻 hier nog een keer elk gebruikte component toe:

- ESP-32: als brein voor deze installatie heb ik de ESP-32 gebruikt. Dit omdat de ESP genoeg rekenkracht heeft lekker klein is en ik er nog niet eerder mee heb gewerkt, zo leer ik ook weer een nieuw component kennen.
- VL6180X en VL53LOX sensoren: De sensoren zijn verbonden met de ESP en communiceren via een I2C protocol.
- Voeding: usb aansluiting aan de ESP en een aparte 6V voeding voor de servo. De grond van de servo voeding is gekoppeld met die van de ESP.
- Servo: om de hoek van de balk in te kunnen stellen
- Potmeter: Om het setpoint van het regelsysteem in te stellen.
- Rode knop: Om de I uit de PID-regelaar te resetten



Figuur 8



Figuur 9

setpoint. De PID-regelaar wordt toegepast op basis van de fout, de foutintegratie en de foutafgeleide. De resulterende stuursignaalwaarde wordt berekend en doorgegeven aan de `set_servo` functie om de servo aan te sturen.

Softwareschema's

Bibliotheken:

- Arduino
- SPI
- Wire
- Adafruit_VL6180X
- Adafruit_VL53LOX
- Servo

De Adafruit library maakt gebruik van de SPI en Wire library vandaar dat deze included zijn.

Functies:

- `set_servo`: functie wordt gedefinieerd om de servo aan te sturen op basis van de stuursignaalwaarde.
- `moving_average`: Functie voor het berekenen van de moving average
- `Exponentieel_moving_average`: Functie voor het berekenen van de exponentieel moving average

In de loop wordt het hoofdgedeelte van de code uitgevoerd. Hier wordt eerst het gewenste setpoint bepaald op basis van de potentiometerwaarde. Vervolgens worden afstandsgegevens van de sensoren verzameld.

De verkregen afstandswaarden worden verwerkt met behulp van het voortschrijdend gemiddelde en het exponentiële voortschrijdend gemiddelde. Deze waarden worden gebruikt om de fout te bepalen ten opzichte van het