EINDWERK : DE SLUIS

MAURITZ VERBEKE

Voorwoord

Het project De Sluis is gebaseerd op een echte sluis, maar op schaal nagebouwd. Deze miniatuursluis bevat Twee deuren, aangedreven door Twee motoren, en is uitgerust met een waterpomp om het waterniveau op peil te houden. Omdat het een proefopstelling betreft, wordt de sluis bestuurd met behulp van een Raspberry Pi.
Op de Raspberry Pi draait Django in combinatie met Daphne en Channels. Deze software maakt het mogelijk om een gebruikersinterface te creëren waarmee de sluis op afstand bediend kan worden. Het controlepaneel lijkt op een traditioneel sluispaneel, met knoppen en een scherm dat realtime data weergeeft over de status van de sluis.
Een mogelijke uitbreiding van het project is de toevoeging van camera's. Hiermee kan de sluis op afstand worden gemonitord en bediend, zonder dat directe visuele observatie nodig is. Daarnaast biedt dit een basis voor een AI-functie, waarmee de sluis zelfstandig kan opereren.
Met AI kan de sluis bijvoorbeeld detecteren wanneer een boot aan de linkerkant nadert en vervolgens de deuren en het waterniveau voorbereiden om de boot naar de rechterkant te verplaatsen. Deze technologie kan niet alleen de functionaliteit verbeteren, maar ook een stap zetten richting geavanceerde automatisering.

Inhoudsopgave

Moeilijke Woordenlijst

Moeilijk Woord	Eenvoudige Uitleg
Schutsluis	Een soort waterdeur die boten helpt om van het ene waterniveau naar het andere te gaan, zoals een lift voor boten.
Raspberry Pi	Een klein, goedkoop computertje dat je kunt programmeren om dingen automatisch te laten doen, zoals een robot aansturen.
Django	Een framework waarmee je websites en apps kunt bouwen.
Daphne	Een hulpmiddel dat ervoor zorgt dat je website snel en live kan reageren, zoals bij een chat of live data.
Channels	Een uitbreiding van Django die zorgt voor live communicatie, zoals het tonen van live waterstanden.
Frontend	Het deel van een website dat je ziet en gebruikt, zoals knoppen en schermen.
Backend	Het onzichtbare deel van een website dat alles regelt en berekent achter de schermen.
React	Een hulpmiddel om mooie en snelle websites te maken.
Sensor	Een apparaatje dat iets meet, zoals hoe hoog het water staat.
Ultrasone sensor (HC-SR04)	Een sensor die met geluidsgolven meet hoe ver iets weg is, zoals een vleermuis dat doet.
Stappenmotor (28BYJ-48)	Een motortje dat heel precies kan draaien, stapje voor stapje.
Fuzzy expertsysteem	Een slim systeem dat beslissingen neemt, ook als de informatie een beetje vaag of onduidelijk is.

Moeilijk Woord	Eenvoudige Uitleg
Genetisch algoritme	Een manier waarop een computer leert door te proberen, fouten te maken en zichzelf te verbeteren, net als evolutie.
Particle Swarm Optimization (PSO)	Een methode waarbij de computer leert door te doen alsof het een zwerm vogels is die samen de beste oplossing zoekt.
Artificial Bee Colony (ABC)	Een methode waarbij de computer leert zoals bijen die zoeken naar de beste bloemen.
3D-printen	Een techniek waarmee je echte voorwerpen kunt maken door laagjes plastic op elkaar te leggen.
Lasersnijden	Een techniek waarbij een laserstraal dingen heel precies uit een plaat snijdt.
PLA	Een soort plastic dat vaak gebruikt wordt in 3D-printers.
PETG / ABS / Nylon	Sterkere soorten plastic die beter tegen water en hitte kunnen dan PLA.
GPIO-pinnen	Kleine aansluitpunten op de Raspberry Pi waarmee je dingen kunt aansturen, zoals lampjes of motoren.
WebSocket	Een manier waarop een website live informatie kan ontvangen, zoals een live score of waterstand.
API	Een soort brug tussen twee programma's die met elkaar praten.
ASGI-server	Een speciaal soort server die live communicatie mogelijk maakt.
Dupont-kabels	Kleine kabeltjes waarmee je onderdelen makkelijk kunt verbinden, zoals sensoren met een computertje.
Hot-swappable	Iets dat je kunt vervangen zonder dat je het hele systeem moet uitzetten of opnieuw aansluiten.

1 Inleiding

1.1. Inleiding

Het project ontstond vanuit de nood om technische processen inzichtelijker, efficiënter en beter controleerbaar te maken. Traditionele sluissystemen worden vaak nog handmatig of met beperkte automatisering bediend. In een tijd waarin digitale sturing en monitoring steeds belangrijker worden, is er behoefte aan moderne oplossingen die deze processen automatiseren en visualiseren.

Het doel van dit project is om een schaalmodel van een sluis te ontwikkelen dat de werking van een echte sluis nabootst en tegelijk inspeelt op hedendaagse technologieën. De kern van het project bestaat uit het uitlezen van sensorwaarden, het visualiseren van die data via een webinterface, en het opslaan ervan in een database. Op die manier kan de werking van de sluis op afstand gemonitord en bestuurd worden.

De belangrijkste doelstellingen zijn:

- Het bouwen van een werkende miniatuursluis met motoren, een waterpomp en sensoren.
- Het ontwikkelen van een backend die de data verwerkt en opslaat.
- Het creëren van een frontend die de data in realtime visualiseert en bediening mogelijk maakt.
- Het onderzoeken hoe vergaande automatisering, zoals met AI of cameratoezicht, in de toekomst geïntegreerd kan worden.

De centrale onderzoeksvraag luidt: Hoe kan een miniatuursluis via sensoren, software en webtechnologie op een realistische manier bestuurd en gemonitord worden? Daarnaast wordt onderzocht hoe dit project een basis kan vormen voor verdere automatisering en slimme systemen.

1.2. Inhoud

In dit project wordt gekozen om een sluis op schaal te bouwen die de werking van een echte sluis op kleine schaal nabootst. Het doel is om een systeem te ontwikkelen dat gegevens via sensoren uitleest, deze data visualiseert op een webpagina en opslaat in een database. Dit alles gebeurt in het kader van het eindwerk, waarbij gewerkt wordt met moderne technologieën zoals een Raspberry Pi, React en Django.

In de beginfase werd onderzocht op welke manieren een sluis gemotoriseerd en geautomatiseerd kan worden. Er werd gekeken naar bestaande systemen, waarna een eigen ontwerp werd uitgewerkt. Er zijn meerdere schetsen gemaakt om te kijken welke aanpak het meest haalbaar en betrouwbaar is. Hierbij werd ook rekening gehouden met de vereiste onderdelen voor een werkende sluis zoals motoren, pompen en sensoren.

Vervolgens werd de aandacht gericht op de softwarekant van het project. Er werd een backend gebouwd met Django, die de communicatie met de hardware verzorgt en alle meetgegevens opslaat in een database. Daarnaast

werd een frontend ontwikkeld met React, waarin de gegevens realtime worden weergegeven en de sluis op afstand kan worden bediend.

Tijdens het tweede semester ligt de focus op het verder afwerken en optimaliseren van het systeem, zowel op technisch vlak als op het gebied van documentatie. Er wordt verwacht dat alle onderdelen correct samenwerken en dat het systeem stabiel functioneert. Tot slot wordt er een volledig verslag opgesteld waarin het hele proces wordt uitgelegd, van ontwerp tot uitvoering. Aan het einde van het schooljaar wordt het project voorgesteld aan een jury, die het werk zal beoordelen op technische uitvoering, afwerking en presentatie.

2 Ideëenfase

2.1. Opdrachtomschrijving

De opdracht bestaat uit het ontwerpen en realiseren van een miniatuursluis die de werking van een echte sluis op schaal nabootst. Dit project kadert binnen het eindwerk en moet voldoen aan specifieke technische vereisten: het systeem moet gegevens van sensoren kunnen uitlezen, deze realtime visualiseren op een webpagina en de informatie opslaan in een database.

De sluis bevat twee motorisch aangedreven deuren en een waterpomp voor het regelen van het waterniveau. Het geheel wordt aangestuurd via een Raspberry Pi. De backend van het project is opgebouwd met Django, Daphne en Channels, terwijl de frontend gerealiseerd is in React. De webinterface dient gebruiksvriendelijk te zijn en laat toe om de sluis op afstand te bedienen.

Het project combineert mechanische onderdelen, elektronische componenten en softwareontwikkeling. Het biedt een praktijkgerichte oplossing waarbij verschillende technieken samenkomen. De realisatie van het project moet leiden tot een stabiele en functionele proefopstelling, ondersteund door een duidelijke documentatie en presentatie.

2.2. Doelstellingen

Het doel van dit project is om een geautomatiseerde miniatuursluis te bouwen die de werking van een echte sluis op schaal simuleert en die voldoet aan de gestelde eisen binnen het eindwerk: het uitlezen van sensoren, het realtime visualiseren van gegevens en het opslaan ervan in een database. Om dit te bereiken, zijn er verschillende concrete doelstellingen geformuleerd:

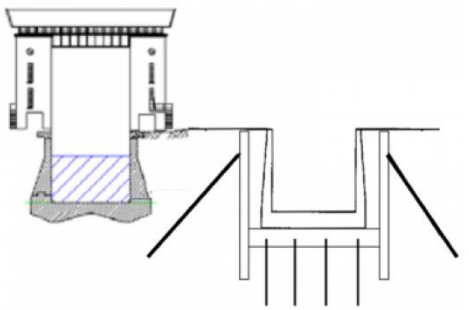
- **Mechanisch ontwerp en realisatie**
  Het bouwen van een miniatuursluis met twee beweegbare deuren, aangedreven door motoren, en een functionele waterpomp die het waterniveau binnen de sluis kan regelen.
- **Elektronische sturing en sensoren**
  Het integreren van sensoren die onder andere het waterniveau meten. Deze sensoren worden verbonden met een Raspberry Pi, die dienstdoet als centrale besturingseenheid.
- **Softwareontwikkeling – backend**
  Het ontwikkelen van een backend in Django, gecombineerd met Daphne en Channels, die verantwoordelijk is voor de communicatie met de hardware, het verwerken van sensordata en het opslaan van gegevens in een database.
- **Softwareontwikkeling – frontend**
  Het bouwen van een gebruiksvriendelijke en visueel aantrekkelijke webinterface in React, waarin de sensordata in realtime wordt weergegeven en van waaruit de sluis op afstand bediend kan worden.
- **Data-opslag en visualisatie**
  Het correct opslaan van alle meetgegevens in een database en het overzichtelijk visualiseren van deze gegevens op de webpagina.
- **Documentatie en presentatie**
  Het opstellen van een volledig en duidelijk technisch verslag, waarin alle stappen van het project worden beschreven. Aan het einde van het traject wordt het project voorgesteld aan een jury.

2.3. Theoretische studie

Voor het ontwerpen van een op afstand bedienbare sluis zijn verschillende technische aspecten onderzocht. In dit onderdeel worden de belangrijkste theorieën, keuzes en beperkingen besproken die relevant zijn voor de uitwerking van dit project.

2.3.1. Systeemkeuze

Er bestaan meerdere soorten sluissystemen, elk met hun eigen werkingsprincipes. Na overleg met een begeleidende leerkracht (Dhr. Everaert) is ge kozen voor een standaard schutsluis, waarbij twee deuren gebruikt worden om het waterniveau te regelen. Dit systeem maakt gebruik van zwaartekracht: door de deuren lichtjes te openen, kan het waterniveau gecontroleerd stijgen of dalen, afhankelijk van de situatie. Dit sluit goed aan bij de schaal en mogelijkheden van een proefopstelling.



Figuur 1: Technische tekening Schutsluis

2.3.2. Raspberry Pi in een vochtige omgeving

De Raspberry Pi vormt het brein van de opstelling, maar is niet bestand tegen vocht. Omdat het project in contact komt met water, moeten maatregelen genomen worden om elektronische schade te vermijden. Eén oplossing is het gebruik van een corrosiewerende spray zoals CorrosionX Trigger Spray. Deze spray brengt een waterafstotende laag aan op de elektronische componenten, waardoor contact met water en vocht vermeden wordt. Dergelijke bescherming is belangrijk om een betrouwbare werking te garanderen.

Figuur 2 : Raspberry Pi 4

2.3.3. Beperkingen van de proefopstelling

In tegenstelling tot een echte sluis beschikt de proefopstelling niet over een natuurlijke of continue watertoevoer. Hierdoor zal het waterniveau na verloop van tijd vanzelf stabiliseren. Om dit effect te compenseren en de dynamiek van een echte sluis beter te benaderen, is een waterpomp toegevoegd. De pomp laat toe om het waterniveau kunstmatig aan te passen, zodat verschillende situaties gesimuleerd kunnen worden tijdens het testen.

2.3.4. HC-SR04 ultrasone sensor

Voor het meten van het waterniveau in de sluis wordt gebruikgemaakt van de HC-SR04 ultrasone sensor. Deze sensor meet de afstand tot het wateroppervlak door middel van ultrasone golven. De sensor zendt een korte geluidspuls uit en wacht op de reflectie. Door de tijd tussen het zenden en ontvangen te meten, wordt de afstand berekend.

De HC-SR04 heeft een meetbereik van 2 cm tot 400 cm en een nauwkeurigheid van ±3 mm. Deze sensor is eenvoudig aan te sluiten op een Raspberry Pi en werkt betrouwbaar voor het meten van vloeistofniveaus in kleine opstellingen. De meetgegevens worden vervolgens doorgestuurd naar de backend, verwerkt en in realtime weergegeven op de webinterface.

Figuur 3 : HC-SR04

2.3.5. 28BYJ-48 stappenmotor

Voor de aandrijving van de sluisdeuren is gekozen voor de 28BYJ-48 stappenmotor, in combinatie met een ULN2003-driver. Deze motor is compact, goedkoop en geschikt voor toepassingen waar nauwkeurige positionering belangrijk is.

De 28BYJ-48 is een unipolaire stappenmotor met een overbrengingsverhouding van ongeveer 64:1, wat zorgt voor een hoog koppel en precieze besturing. De motor draait in stappen van 5,625° per stap (voor een volledige draai zijn dus 64 stappen nodig vóór de overbrenging). Door de overbrenging zijn er effectief 4096 stappen nodig voor één omwenteling van de uitgaande as.

Deze motor is voldoende krachtig om de lichte sluisdeuren in deze proefopstelling te openen en te sluiten. Bovendien is hij eenvoudig aan te sturen met digitale signalen, wat hem ideaal maakt voor integratie in een geautomatiseerd systeem zoals dit.

Figuur 4 : 28BYJ-48 stappenmotor

2.3.6. Fuzzy expertsystemen in sluistoepassingen

Voor het beheer van echte scheepssluizen worden soms fuzzy expertsystemen (FES) gebruikt. Deze systemen nemen beslissingen op basis van vage of onnauwkeurige gegevens, zoals de afstand van een schip tot de sluis of de status van de deuren. Door gebruik te maken van fuzzy logica kunnen systemen flexibeler reageren op veranderende omstandigheden.

Om de prestaties van zulke systemen te optimaliseren, worden verschillende technieken ingezet:

Genetische Algoritmen (GA): maken gebruik van evolutieprincipes zoals selectie en mutatie om oplossingen te verbeteren.

Particle Swarm Optimization (PSO): gebaseerd op het gedrag van zwermen, waarbij elk deeltje zijn positie aanpast op basis van eigen en groepsinformatie.

Artificial Bee Colony Optimization (ABC): geïnspireerd op het zoekgedrag van bijen, waarbij oplossingen geëvalueerd worden aan de hand van hun “nectarwaarde”.

Deze technieken helpen om een goede balans te vinden tussen:

Minimale wachttijd (MWT) – Snelheid van doorgang verhogen.

Minimaal aantal lege sluizen (MNL) – Water- en energieverbruik verminderen.

Onderzoek toont aan dat elk algoritme zijn eigen sterkte heeft, afhankelijk van het scenario en de prioriteit. Zo presteerde ABC goed bij focus op snelle doorgang, terwijl PSO meer geschikt was bij een evenwichtige aanpak.

3 Concept Ontwikkeling

3.1. Situatie

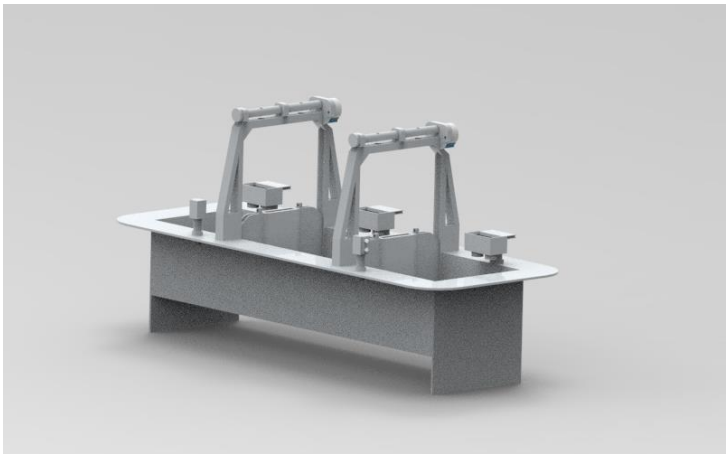
De sluis is een reeds bestaand mechanisch systeem en hoeft in dit project niet volledig herontworpen te worden. Het doel is om een schaalmodel te bouwen dat de werking van een echte sluis op kleine schaal nabootst. Hierbij wordt uitgegaan van het traditionele concept van een schutsluis met twee deuren.

De aanpassing bestaat voornamelijk uit het verkleinen van het ontwerp tot een werkbare proefopstelling. Daarnaast worden componenten toegevoegd die nodig zijn voor de automatisering: sensoren om gegevens te verzamelen, motoren om de deuren te bedienen en een pomp om het waterniveau aan te passen. Verder wordt het systeem gekoppeld aan een elektronische sturing via een Raspberry Pi, waarbij data verwerkt en gevisualiseerd wordt via een webinterface.

Deze aanpak maakt het mogelijk om het traditionele mechanisme van een sluis te combineren met moderne technologieën, zoals afstandsbediening, monitoring en dataverwerking.

3.2. Mechanisch Ontwerp

Voor het ontwerpen van de sluis is gebruikgemaakt van Solid Edge. Dit programma is gekozen omdat er al enige voorkennis mee aanwezig was. Bovendien werkt het gemakkelijk en biedt het veel handige functies, zoals de mogelijkheid om ontwerpen te exporteren voor 3D-printers. Dit maakt het prototypen eenvoudiger en efficiënter.

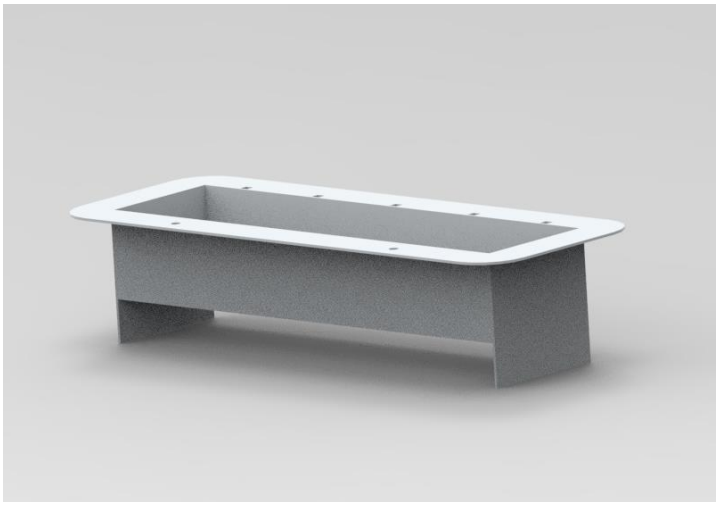


Figuur 5 : Design

3.2.1. Onderdelen van de sluis

3.2.1.1. De sluisbak

De sluisbak is de hoofdconstructie waarin het water wordt vastgehouden en waar de ‘schepen’ doorheen varen. Deze constructie is gemaakt van plexiglas. Deze materiaalkeuze is gemaakt omdat plexiglas eenvoudig op maat te snijden is met een lasersnijder. Daarnaast is het transparant, wat het mogelijk maakt om de werking van de sluis en de interne mechanismen goed zichtbaar te maken voor de mensen.

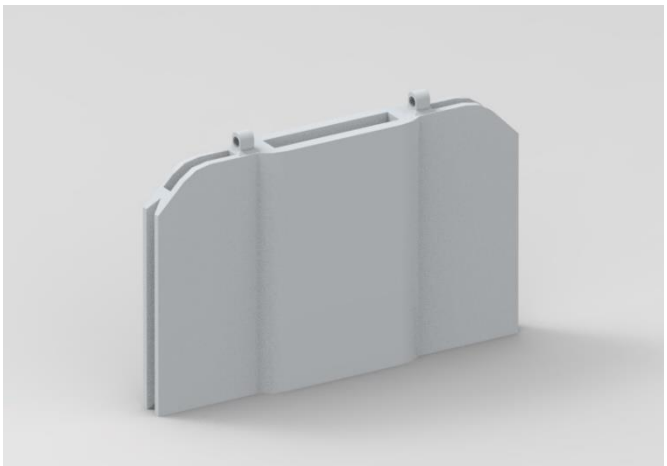


Figuur 6 : De sluisbak

3.2.1.2. De Sluisdeuren

De sluisdeuren vormen de beweegbare onderdelen die verantwoordelijk zijn voor het regelen van het waterniveau en het mogelijk maken van doorvaart. Deze deuren zijn vervaardigd met behulp van een 3D-printer, waarbij het materiaal PLA (Polylactic Acid) is gebruikt.

Hoewel PLA niet het meest geschikte materiaal is voor toepassingen in natte of buitenomgevingen, voldoet het in dit prototype ruim voldoende. PLA is namelijk eenvoudig te printen, goedkoop en heeft een nette afwerking, wat



Figuur 7 : De sluisdeur

het ideaal maakt voor demonstratiemodellen en prototypes die niet continu worden blootgesteld aan water of mechanische belasting.

3.2.1.3. De geleiderails

De geleiderails zorgen ervoor dat de sluisdeuren gecontroleerd kunnen bewegen tijdens het openen en sluiten. Ze bepalen als het ware de looprichting van de deuren. Deze onderdelen zijn 3D-geprint om maatwerk mogelijk te maken en sluiten naadloos aan op de constructie van de sluis.

De rails zijn in twee delen geprint, waardoor het mogelijk is om ze eenvoudig uit elkaar te halen wanneer de deur of een van de raildelen vervangen moet worden. Deze twee delen worden verbonden met een eenvoudig rechthoekig verbindingsstuk dat in beide raildelen past.

Figuur 8 : De geleiderails

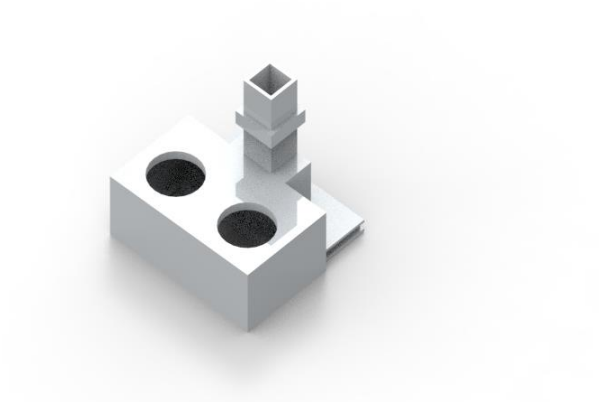
3.2.1.4. De opwindas

De opwindas is een belangrijk mechanisch onderdeel waarmee het touw voor het openen en sluiten van de sluisdeuren wordt op- en afgerold. Dit onderdeel is eveneens 3D-geprint en maakt het mogelijk om de deuren handmatig of automatisch te bedienen via een spoelmechanisme.

Figuur 9 : De opwindas

3.2.1.5. Sensorhouder met deksel

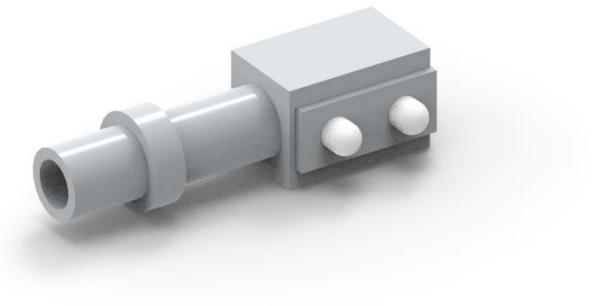
De sensorhouders zorgen voor een stabiele bevestiging van de sensoren die de positie of status van de sluis detecteren. De deksels beschermen de sensoren tegen spatwater of mechanische schade. Beide onderdelen zijn ontworpen en geprint in PLA en passen functioneel binnen het sluismodel.



Figuur 10 : Sensorhouder met deksel

3.2.1.6. De sluisverkeerslichten

De sluisverkeerslichten geven aan of de sluis open of gesloten is voor de ‘scheepvaart’. Dit onderdeel bestaat uit een eenvoudige behuizing met LED’s.



Figuur 11 : Sluisverkeerslicht

3.2.2. Productiemethoden

Voor de realisatie van het miniatuurmodel van de sluis zijn twee verschillende productietechnieken toegepast: 3D-printen en lasersnijden van plexiglas. De keuze voor deze methodes is gebaseerd op het type onderdeel, de gewenste nauwkeurigheid, het materiaalgedrag en de beschikbaarheid van apparatuur.

3.2.2.1. 3D-printen

De kleinere en meer complexe onderdelen, zoals de sluisdeuren, sensorhouders, de stoplichten, de geleiderails en de opwindas, zijn vervaardigd met een 3D-printer. Hiervoor is gebruikgemaakt van het materiaal PLA, vanwege de eenvoudige verwerkbaarheid en voldoende sterkte voor een werkend prototype. Deze methode maakt het mogelijk om onderdelen met interne structuren of bijzondere vormen te produceren zonder extra assemblage.

Figuur 12 : 3D printer

3.2.2.2. Lasersnijden

De grotere vlakke onderdelen, zoals de sluisbak en eventuele ondersteunende wanden of platen, zijn uitgesneden uit plexiglas met behulp van een lasersnijder. Plexiglas is gekozen vanwege de transparantie (om de werking zichtbaar te maken) en het feit dat het eenvoudig en nauwkeurig te bewerken is met een lasersnijder. Deze methode levert strakke randen en een professionele uitstraling op.

Figuur 13 : Lazer cutter

3.2.3. Materiaalstudie 3D-geprinte onderdelen

Voor het prototype van de sluis zijn de 3D-geprinte onderdelen vervaardigd uit PLA (Polylactic Acid). PLA is een veelgebruikt 3D-printmateriaal vanwege de lage printtemperatuur, goede detailweergave en gebruiksgemak. Voor prototypetoepassingen en modellen die niet onder zware omstandigheden functioneren, is PLA een praktische keuze.

Echter, PLA heeft belangrijke beperkingen bij langdurig gebruik in vochtige of warme omgevingen:

- Het is niet waterbestendig en kan opzwellen of vervormen bij langdurige blootstelling aan vocht.
- Het is gevoelig voor Uv-straling, wat leidt tot veroudering en brosheid.
- Het begint al te vervormen bij relatief lage temperaturen (rond 60 °C).

3.2.3.1. Aanbevolen alternatieven

Voor een robuustere en waterbestendige uitvoering van de 3D-geprinte onderdelen zou gekozen kunnen worden voor één van de volgende materialen:

3.2.3.1.1. PETG (Polyethylene Terephthalate Glycol)

- Voordelen: Goede water- en chemische bestendigheid, sterker dan PLA, Uv-bestendig, eenvoudig te printen.

- Toepassing: Zeer geschikt voor functionele onderdelen die in contact komen met water, zoals sluisdeuren of geleiderails.

3.2.3.1.2. ABS (Acrylonitrile Butadiene Styrene)

- Voordelen: Hoge slagvastheid, bestand tegen hogere temperaturen, redelijk bestand tegen water.
- Nadelen: Moeilijker te printen (krimpt snel), vereist geventileerde ruimte door geurvorming.
- Toepassing: Geschikt voor structurele onderdelen die onder belasting staan.

3.2.3.1.3. Nylon (PA – Polyamide)

- Voordelen: Zeer sterk, slijtvast, flexibel, redelijk waterbestendig.
- Nadelen: Moeilijker te printen, gevoelig voor vochtopname tijdens opslag.
- Toepassing: Ideaal voor onderdelen met mechanische belasting of bewegende onderdelen.

3.2.3.2. Conclusie

Voor dit prototype volstaat PLA vanwege het tijdelijke en visuele karakter van het model. Indien de sluis onder realistischere omstandigheden moet functioneren (bijvoorbeeld langdurig in natte omgevingen of met externe belasting), is PETG een zeer geschikte vervanger dankzij de goede waterbestendigheid en printbaarheid.

3.3. Elektronisch ontwerp

Het elektrisch ontwerp van de miniatuursluis is opgebouwd rond een Raspberry Pi 4, die fungeert als de centrale besturingseenheid. Via de GPIO-pinnen worden de sensoren, LED's en motoren aangestuurd. De Raspberry Pi is verbonden met een gebruiksvriendelijke webinterface, gebouwd in React, waarmee de gebruiker de sluis handmatig bedient.

3.3.1. Componentenoverzicht

- ultrasone sensoren (HC-SR04): Deze worden gebruikt om het waterniveau op drie verschillende punten in de sluis te meten. Afhankelijk van het gemeten niveau bepaalt de Raspberry Pi of de deuren geopend of gesloten mogen worden.
- 2 stappenmotoren (28BYJ-48 met ULN2003-driver): Deze sturen de sluisdeuren aan. Elke motor is gekoppeld aan een opwindas die de deur mechanisch opent of sluit.
- LED's (2 rood, 2 groen): Deze LED's fungeren als verkeerslichten voor de sluis. Aan elke kant van de sluis staat een rood/groen combinatie die aangeeft of doorvaart toegestaan is.
- Raspberry Pi 4 Model B: Verzorgt de aansturing van het systeem via Python-scripts en leest continu de waarden van de sensoren en stuurt daarop de actuatoren aan.

3.3.2. Werking van het systeem

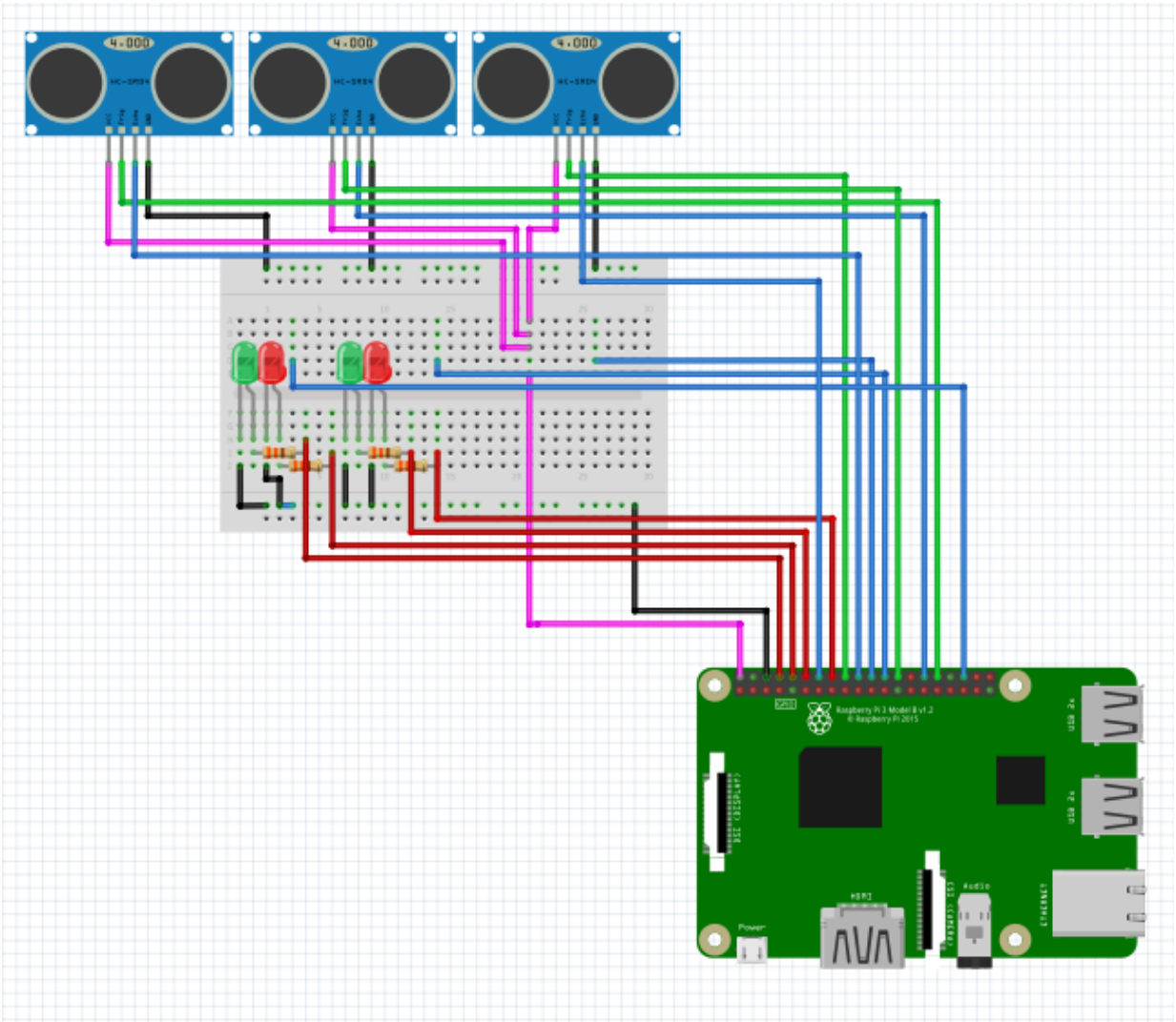
De sluis wordt handmatig bediend via een React-gebruikersinterface die draait op de Raspberry Pi of via het lokale netwerk toegankelijk is. In deze interface heeft de gebruiker knoppen ter beschikking om:

- De sluisdeuren te openen of te sluiten.
- Het waterniveau af te lezen op basis van de waarden van de ultrasone sensoren.

Wanneer de gebruiker een actie uitvoert in de interface (bijvoorbeeld een knop indrukt), stuurt React een HTTP-verzoek (API-call) naar een Python-backend op de Raspberry Pi, die vervolgens het juiste component aanstuurt.

3.3.3. Elektrisch schema

De onderstaande afbeelding toont het huidige elektrisch schema van het systeem:

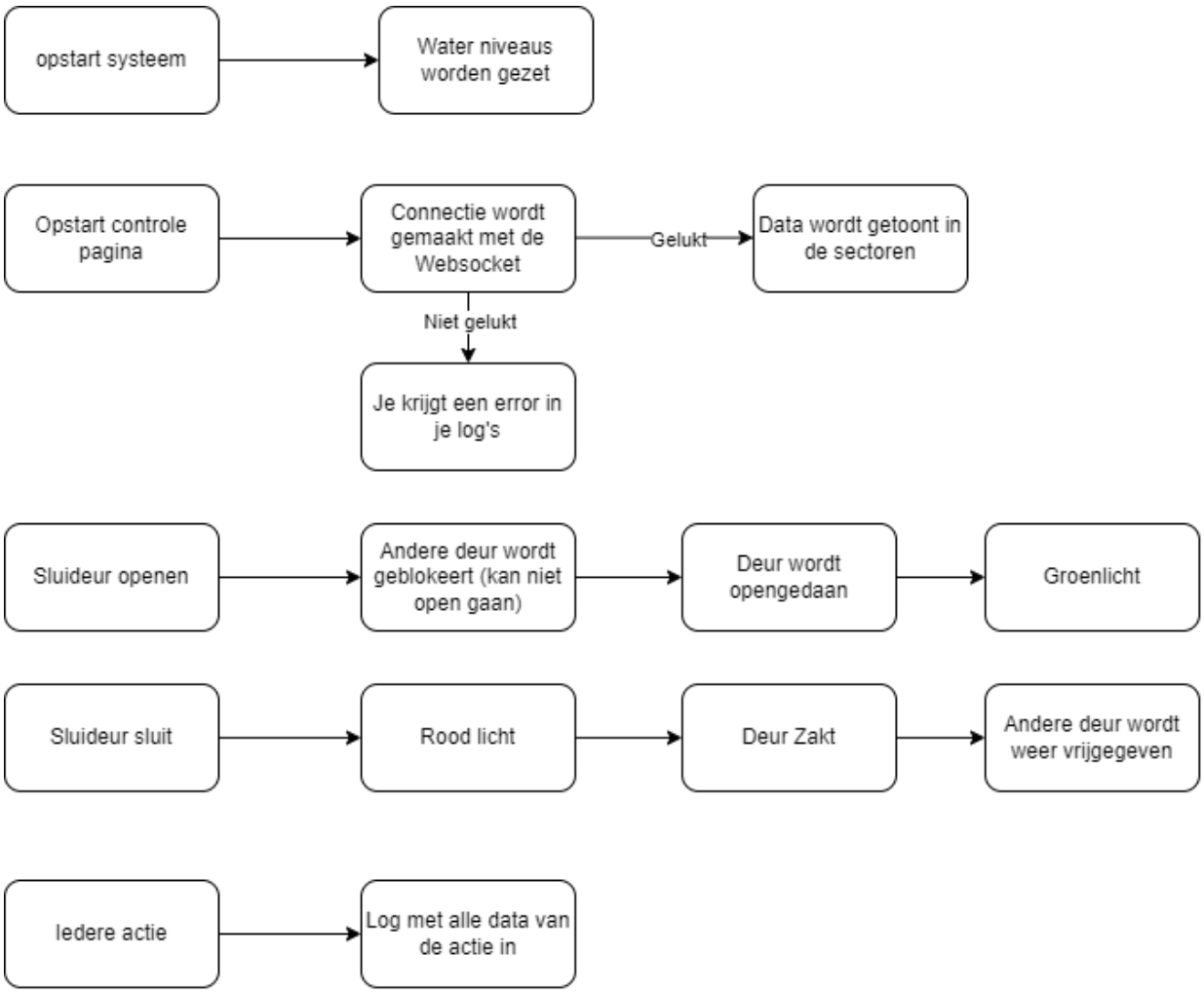


Figuur 14 : Elektrisch schema

In dit schema ontbreken nog de aansluitingen voor de stappenmotoren. Deze zullen worden toegevoegd via vier GPIO-pinnen (stuursignalen) en één 5V-pin DC (voeding)

3.4. Softwarearchitectuur

Onderstaande diagram geeft een overzicht van de logische stappen die het systeem doorloopt bij het uitvoeren van een actie, zoals het openen van een sluisdeur. Het vormt een vereenvoudigde weergave van de softwarearchitectuur in actie, waarbij de communicatie tussen de gebruikersinterface (React), de backend op de Raspberry Pi, en de hardwarecomponenten zichtbaar wordt.



Figuur 15 : procesoverzicht

4 Implementatie

4.1. Frontend (React)

De frontend is ontwikkeld met behulp van React, een populaire JavaScript-bibliotheek voor het bouwen van dynamische en component gebaseerde gebruikersinterfaces. De interface van de applicatie bestaat uit twee hoofdonderdelen

4.1.1. Hoofdpagina's

- **Homepagina**:
  Dit is de startpagina van de applicatie. Hier krijgt de gebruiker een kort overzicht van de functionaliteit van het systeem. Daarnaast is er een knop voorzien om door te gaan naar het controlepaneel. De bedoeling van deze pagina is om gebruikers op een eenvoudige en duidelijke manier kennis te laten maken met de werking van het systeem.
- **Controlepaneel**:
  In dit onderdeel kan de gebruiker effectief handelingen uitvoeren. Hier kunnen de sluisdeuren geopend of gesloten worden, is het waterniveau zichtbaar via metingen van de sensoren, en wordt er een realtime log weergegeven van alle acties en statussen van het systeem. Dit is het operationele centrum van de toepassing.

4.1.2. Routing

Voor de navigatie tussen de verschillende pagina's wordt gebruikgemaakt van React Router. Deze techniek zorgt ervoor dat gebruikers naadloos kunnen wisselen tussen pagina's zonder dat de volledige website opnieuw moet laden.

Voorbeeld van routingstructuur:

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";

<Router>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/controlepaneel" element={<ControlPanel />} />
  </Routes>
</Roccuter>;
```

4.1.3. Componentstructuur binnen het Controlepaneel

Het controlepaneel bestaat uit meerdere kleinere React-componenten, elk verantwoordelijk voor een specifieke functie:

4.1.3.1. ControlPanel.jsx

Dit is de centrale component. Het beheert de staat van de sluis (waterstanden, deurposities, verkeerslichten, logs, etc.) en de interactie met de "backend" (via Web Socket en HTTP calls).
Belangrijke onderdelen in ControlPanel.jsx:

State Management (useState):

- waterLevels: Houdt de waterstanden van de drie sensoren bij. Begint op null om een laadtoestand aan te geven.
- logs: Een array om berichten over de sluisactiviteit op te slaan en weer te geven.
- time: De huidige tijd, wordt elke seconde bijgewerkt.
- deur1Open, deur2Open: Booleans die bijhouden of deur 1 respectievelijk deur 2 open zijn.
- deur1InBeweging, deur2InBeweging: Booleans die bijhouden of een deur momenteel in beweging is (wordt gebruikt om knoppen uit te schakelen tijdens de animatie).
- stoplicht1Status, stoplicht2Status: De status ('rood' of 'groen') van de verkeerslichten bij elke deur.
- isDarkMode: Boolean om de donkere/lichte modus van de UI te beheren.
- isSettingsOpen: Boolean om te regelen of het instellingenmenu open is.
- isLogsVisible: Boolean om te regelen of de logs zichtbaar zijn.
- useWebSocket: Een boolean om te schakelen tussen live WebSocket data en dummy data.

```
const [waterLevels, setWaterLevels] = useState({
  Sensor1: null,
  Sensor2: null,
  Sensor3: null,
});

const [logs, setLogs] = useState([]);
const [time, setTime] = useState("");
const [deur1Open, setDeur1Open] = useState(false);
const [deur2Open, setDeur2Open] = useState(false);
const [deur1InBeweging, setDeur1InBeweging] = useState(false);
const [deur2InBeweging, setDeur2InBeweging] = useState(false);
const [stoplicht1Status, setStoplicht1Status] = useState('rood');
const [stoplicht2Status, setStoplicht2Status] = useState('rood');
```

Effects (useEffect):

- Een effect om de huidige tijd elke seconde bij te werken.
- Een effect dat, afhankelijk van de USE_WEBSOCKET variabele (of de useWebSocket state als je die via instellingen wilt kunnen schakelen), een WebSocket-verbinding opzet of een interval start voor dummy data. Dit effect verwerkt inkomende berichten (waterstanden) en updates de staat.
- Effecten voor het beheren van de UI-thema (donker/licht) en het sluiten van het instellingenmenu bij een klik buiten het menu.

```
    useEffect(() => {
      const interval = setInterval(() => {
        const now = new Date();
        const formattedTime = now.toLocaleTimeString("nl-NL", { hour12: false
});
        setTime(formattedTime);
      }, 1000);
      return () => clearInterval(interval);
    }, []);

    useEffect(() => {
    if (USE_WEBSOCKET) {
      const wsUrl = `ws://${window.location.hostname}:8000/ws/sensors/`;
      ws.current = new WebSocket(wsUrl);

      ws.current.onopen = () => {
        addLog('WebSocket verbinding geopend');
      };
```

```
      ws.current.onmessage = (event) => {
        const data = JSON.parse(event.data);
        setWaterLevels({
          Sensor1: data.Sensor1,
          Sensor2: data.Sensor2,
          Sensor3: data.Sensor3,
        });
        addLog(`WebSocket data ontvangen: Sensor1=${data.Sensor1},
Sensor2=${data.Sensor2}, Sensor3=${data.Sensor3}`);
      };
```

```
    // Dummy data effect
    useEffect(() => {
      if (!USE_WEBSOCKET) {
        const dummyInterval = setInterval(() => {
          const base = Math.floor(Math.random() * 6) + 3;
          const variation = Math.floor(Math.random() * 3) - 1;

          const dummyData = {
            Sensor1: base,
            Sensor2: base + variation,
            Sensor3: base - variation,
          };
          setWaterLevels(dummyData);
          addLog(
            `Dummy data: Sensor1=${dummyData.Sensor1},
Sensor2=${dummyData.Sensor2}, Sensor3=${dummyData.Sensor3}`
          );
        }, 1000);
        return () => clearInterval(dummyInterval);
      }
    }, []);
```

handleDeurActie functie:

- Deze asynchrone functie wordt aangeroepen wanneer op een "Openen" of "Sluiten" knop wordt geklikt.
- Het stuurt een POST-verzoek naar de backend API (/api/deur/{deurNummer}/).
- Tijdens de actie wordt de bijbehorende deurInBeweging staat op true gezet om de knoppen uit te schakelen.
- Na een geslaagd API-antwoord, wordt een setTimeout gebruikt om de deurOpen staat en de stoplichtStatus bij te werken na een gesimuleerde "animatietijd" van 3 seconden. De deurInBeweging staat wordt dan weer op false gezet.
- Foutafhandeling logt eventuele problemen

```
    const handleDeurActie = async (deurNummer, actie) => {
      const isOpenen = actie === 'openen';
      const setDeurOpen = deurNummer === 1 ? setDeur1Open : setDeur2Open;
      const setDeurBeweging = deurNummer === 1 ? setDeur1InBeweging :
setDeur2InBeweging;
      const setStoplichtStatus = deurNummer === 1 ? setStoplicht1Status :
setStoplicht2Status;

      setDeurBeweging(true);
      addLog(`Sluisdeur ${deurNummer} ${isOpenen ? 'opent' : 'sluit'}...`);

      try {
        const res = await
fetch(`http://${window.location.hostname}:8000/api/deur/${deurNummer}/`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ actie }),
```

```
        });

        const result = await res.json();

        if (res.ok) {
          setTimeout(() => {
            setDeurOpen(isOpenen);
            setDeurBeweging(false);
            setStoplichtStatus(result.stoplicht_status);
            addLog(`Sluisdeur ${deurNummer} ${isOpenen ? 'geopend' :
'gesloten'}`);
          }, 3000); // animatietijd
        } else {
          setDeurBeweging(false);
          addLog(`Fout bij openen/sluiten van deur ${deurNummer}:
${result.error}`);
        }
      } catch (err) {
        setDeurBeweging(false);
        addLog(`Netwerkfout bij deur ${deurNummer}: ${err.message}`);
      }
    };
```

JSX Structuur (UI):

- De component rendert de hele controlepaneel-pagina.
- Het bevat een header met logo en een "Terug naar Home" knop.
- De "screen" div bevat de weergave van de sluis met watersectoren en deuren.
- Er worden drie WaterSector componenten gebruikt om de waterstanden van Sensor1, Sensor2 en Sensor3 weer te geven.
- Tussen de WaterSector componenten worden de "sluizen" getoond.
- Elke "sluis" bevat een Stoplichten component en een div die de deur visueel voorstelt.
- De opacity van de sluisdeur div wordt dynamisch ingesteld op basis van de deurOpen staat, met een CSS transition voor een soepele animatie.
- De knoppen "Openen" en "Sluiten" roepen de handleDeurActie functie aan met het juiste deurnummer en de actie ('openen' of 'sluiten'). De disabled prop zorgt ervoor dat de knoppen niet klikbaar zijn wanneer een deur in beweging is, al open is (voor "Openen"), of nog gesloten is (voor "Sluiten"). Ook is er een onderlinge blokkade: als deur 1 open is, kan deur 2 niet geopend worden en vice versa.
- Onderaan is er een sectie voor het weergeven van de logs, met een knop om deze te tonen of te verbergen.

```
    return (
      <div className={isDarkMode ? 'dark-mode' : 'light-mode'}>
        {/* ... header ... */}

        <div className="screen">
          <h1>{time}</h1>
          <div className="sluis-container">
            <WaterSector level={waterLevels.Sensor1} sectorClass="sector1" />
            <div className="sluis">
              <Stoplichten status={stoplicht1Status} />
              <div
                className="sluisdeur"
                style={{
                  opacity: deur1Open ? 0.3 : 1,
                  transition: 'opacity 3s ease'
                }}
              ></div>
              <div className="knoppen">
                <button
                  className="openen"
                  onClick={() => handleDeurActie(1, 'openen')}
                  disabled={deur1Open || deur1InBeweging || deur2Open}
                >
                  Openen
                </button>
                <button
                  className="sluiten"
                  onClick={() => handleDeurActie(1, 'sluiten')}
                  disabled={!deur1Open || deur1InBeweging}
                >
                  Sluiten
```

```
                </button>
              </div>
            </div>
            <WaterSector level={waterLevels.Sensor2} sectorClass="sector2" />
            <div className="sluis">
              <Stoplichten status={stoplicht2Status} />
              <div
                className="sluisdeur"
                style={{
                  opacity: deur2Open ? 0.3 : 1,
                  transition: 'opacity 3s ease'
                }}
              ></div>
              <div className="knoppen">
                <button
                  className="openen"
                  onClick={() => handleDeurActie(2, 'openen')}
                  disabled={deur2Open || deur2InBeweging || deur1Open}
                >
                  Openen
                </button>
                <button
                  className="sluiten"
                  onClick={() => handleDeurActie(2, 'sluiten')}
                  disabled={!deur2Open || deur2InBeweging}
                >
                  Sluiten
                </button>
              </div>
            </div>
            <WaterSector level={waterLevels.Sensor3} sectorClass="sector3" />
          </div>
        </div>

        {/* ... logs container ... */}
      </div>
    );
  };
```

4.1.3.2. WaterSector.jsx

Deze component is verantwoordelijk voor het visualiseren van een watersector en het weergeven van de waterstand.
Belangrijke onderdelen in WaterSector.jsx:

- Props: Het ontvangt level (de waterstand in cm) en sectorClass (voor specifieke styling van de sector).
- State (useState) & Effect (useEffect) voor laadtoestand: Als level nog null is (wat aangeeft dat er nog geen data is ontvangen), toont de component "Loading..." met bewegende puntjes. De useEffect en useState haken beheren deze puntjes animatie.
- Berekening Waterniveau Visualisatie: De level in cm wordt omgerekend naar een percentage (percentage = (level / 10) * 100) om de hoogte van de "water-level" div te bepalen. Dit suggereert dat 10 cm de maximale hoogte is die de visualisatie kan tonen.
- JSX Structuur:
    - Toont de waterstand (${level} cm) of de laadtekst (Loading${dots}).
    - De water-level div wordt visueel gevuld met water door de height stijl dynamisch in te stellen op basis van het berekende percentage.

```
    import React, { useState, useEffect } from 'react';

    const WaterSector = ({ level, sectorClass }) => {
        const percentage = level ? (level / 10) * 100 : 0;
        const [dots, setDots] = useState('');

        useEffect(() => {
          if (!level) {
            const interval = setInterval(() => {
              setDots(prev => prev.length >= 3 ? '' : prev + '.');
            }, 500);
            return () => clearInterval(interval);
          }
        }, [level]);

        return (
          <div className={`sector ${sectorClass}`} style={{ position:
"relative" }}>
            <div className="water-text">{level ? `${level} cm` :
`Loading${dots}`}</div>
            <div className="water-level" style={{ height: `${percentage}%`
}}></div>
          </div>
        );
      };

      export default WaterSector;
```

4.1.3.3. Stoplichten.jsx

Deze component toont de verkeerslichten bij een sluisdeur.
Belangrijke onderdelen in Stoplichten.jsx:

- Prop: Het ontvangt een status prop, die 'rood' of 'groen' kan zijn.
- JSX Structuur:
    - Het bevat twee divs, één voor het rode licht en één voor het groene licht.
    - De klasse van elke div bevat voorwaardelijk 'actief' (${status === 'rood' ? 'actief' : ''}). Dit zorgt ervoor dat via CSS de juiste kleur oplicht op basis van de status prop.

```
import React from 'react';

const Stoplichten = ({ status }) => {
  return (
    <div className="stoplicht">
      <div className={`licht rood ${status === 'rood' ? 'actief' :
''}`}></div>
      <div className={`licht groen ${status === 'groen' ? 'actief' :
''}`}></div>
    </div>
  );
};

export default Stoplichten;
```

4.1.4. Communicatie met de backend

De frontend communiceert met de backend (op de Raspberry Pi) via twee methodes:

- Web Socket: Voor realtime updates van sensordata (zoals waterniveaus).
- HTTP-aanvragen (API): Voor het uitvoeren van acties zoals het openen of sluiten van de deuren.

De uitleg van de web socket en de API volgt nog in het volgende hoofdstukje over de backend, waar ook grondig wordt toegelicht waarom deze onderdelen worden gebruikt.

4.2. Backend (Django, Channels, Daphne)

De backend van dit project vormt het hart van de communicatie tussen de hardware (zoals sensoren en motoren) en de webinterface. Hiervoor werd gekozen voor het Django-framework, uitgebreid met Django Channels en Daphne om realtime communicatie mogelijk te maken.

4.2.1. Django als basisframework

Django is een krachtig Python-framework dat gebruikt wordt voor het bouwen van veilige, schaalbare en onderhoudbare webapplicaties. In dit project wordt Django gebruikt voor:

- Het beheren van de database (sensorwaarden).
- Het aanbieden van een REST API waarmee de frontend gegevens kan opvragen of acties kan uitvoeren.
- Het aanbieden van Web sockets waarmee de frontend constant de data van de sensoren krijgt.
- Het verwerken van aansturingscommando's (zoals het openen van deuren of activeren van de pomp).
  ik

De Django-applicatie draait op de Raspberry Pi en vormt de brug tussen de fysieke componenten en de gebruikersinterface.

4.2.2. Realtime communicatie met Django Channels

Voor realtime communicatie werd Django Channels geïntegreerd. Channels maakt het mogelijk om WebSockets te gebruiken, wat essentieel is voor toepassingen waarbij continue gegevensstroom nodig is, zoals bij het uitlezen van sensoren.

In dit project worden drie sensoren via Web Sockets uitgelezen:

- Waterniveau (HC-SR04 ultrasone sensor)

Deze sensoren worden op vaste intervallen (bijvoorbeeld elke seconde) uitgelezen door een Python-script dat draait op de Raspberry Pi. De meetwaarden worden vervolgens via een Web Socket-verbinding naar de frontend gestuurd, zodat de gebruiker in realtime de status van de sluis kan volgen.

4.2.3. Daphne als ASGI-server

Omdat Web Sockets niet ondersteund worden door traditionele WSGI-servers, wordt Daphne gebruikt als ASGI-server. Daphne zorgt ervoor dat zowel HTTP-verkeer als Web Socket-verkeer correct wordt afgehandeld. Hierdoor kunnen gebruikers gelijktijdig de webinterface gebruiken en live updates ontvangen zonder vertraging.

4.2.4. Opbouw van de backend

De backend bestaat uit verschillende onderdelen:

- Models: definiëren de structuur van de database (bv. Sensor Data).
- Views/API: REST-endpoints voor handmatige bediening of statusopvraging.
- Consumers: WebSocket-handlers die inkomende berichten verwerken en updates terugsturen.
- Hardware-controllers: Python-scripts die de GPIO-pinnen van de Raspberry Pi aansturen en sensorgegevens verzamelen.

Een typische flow:

1. De sensoren worden continu uitgelezen.
2. De waarden worden via een Web Socket naar de frontend gestuurd.
3. De gebruiker ziet de waarden live op het dashboard.
4. Bij een actie (bv. knop indrukken om deur te openen) stuurt de frontend een API-bericht naar de backend.
5. De backend voert de actie uit via de GPIO-pinnen.

4.2.5. Backend code uitleg

4.2.5.1. consumers.py - WebSocket en Sensor Functionaliteit

Dit bestand handelt de realtime communicatie met de ultrasone sensoren af. Het zorgt ervoor dat de waterstanden continu worden gemeten en naar de frontend worden gestuurd.
Belangrijke onderdelen

4.2.5.1.1. Sensor Afstandsmeting

```
def read_distance(trigger_pin, echo_pin):
    # Configureert de GPIO pins voor de ultrasone sensor
    GPIO.setup(trigger_pin, GPIO.OUT)
    GPIO.setup(echo_pin, GPIO.IN)

    # Stuurt een trigger pulse
    GPIO.output(trigger_pin, True)
    time.sleep(0.00001)
    GPIO.output(trigger_pin, False)

    # Meet de echo tijd
    while GPIO.input(echo_pin) == 0:
        pulse_start = time.time()
    while GPIO.input(echo_pin) == 1:
        pulse_end = time.time()

    # Berekent de afstand
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17165  # snelheid van geluid = 34330 cm/s / 2
    return round(distance, 1)
```

4.2.5.1.2. SensorConsumer Class

```
class SensorConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        # Maakt WebSocket verbinding
        self.room_name = "sensor_data"
        self.room_group_name = f"sensors_{self.room_name}"
        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )
        await self.accept()
        await self.send_sensor_data()

    async def send_sensor_data(self):
        # Configureert 3 sensoren
        sensor_1_trigger = 3
        sensor_1_echo = 2
```

```
        sensor_2_trigger = 27
        sensor_2_echo = 17
        sensor_3_trigger = 9
        sensor_3_echo = 10

        while True:
            # Leest sensoren in aparte threads
            loop = asyncio.get_event_loop()
            sensor_1_distance = await loop.run_in_executor(None, read_distance,
sensor_1_trigger, sensor_1_echo)
            sensor_2_distance = await loop.run_in_executor(None, read_distance,
sensor_2_trigger, sensor_2_echo)
            sensor_3_distance = await loop.run_in_executor(None, read_distance,
sensor_3_trigger, sensor_3_echo)

            # Stuurt data naar frontend
            sensors = {
                "Sensor1": sensor_2_distance,
                "Sensor2": sensor_2_distance,
                "Sensor3": sensor_2_distance
            }
            await self.send(text_data=json.dumps(sensors))
```

4.2.5.2. motor_control.py - Motor en Stoplicht Besturing

Dit bestand bevat alle logica voor het besturen van de motoren en stoplichten. Het zorgt voor de fysieke beweging van de deuren en de bijbehorende stoplichten.
Belangrijke onderdelen

4.2.5.2.1. Pin Configuraties:

```
# Motor pins voor beide deuren
MOTOR1_PINS = {
    'IN1': 26, 'IN2': 13,
    'IN3': 19, 'IN4': 6
}
MOTOR2_PINS = {
    'IN1': 16, 'IN2': 12,
    'IN3': 21, 'IN4': 20
}

# Stoplicht pins voor beide deuren
STOPLICHT1_PINS = {
    'rood': 23,  # GPIO23 voor rood licht deur 1
    'groen': 24  # GPIO24 voor groen licht deur 1
}
STOPLICHT2_PINS = {
    'rood': 8,   # GPIO8 voor rood licht deur 2
    'groen': 25  # GPIO25 voor groen licht deur 2
}
```

4.2.5.2.2. Motor stappen:

```
# Half-step sequence voor vloeiende beweging
STEP_SEQUENCE = [
    [1, 0, 0, 0], [1, 1, 0, 0],
    [0, 1, 0, 0], [0, 1, 1, 0],
    [0, 0, 1, 0], [0, 0, 1, 1],
    [0, 0, 0, 1], [1, 0, 0, 1]
]

# Full-step sequence voor maximale kracht
FULL_STEP_SEQUENCE = [
    [1, 1, 0, 0], [0, 1, 1, 0],
    [0, 0, 1, 1], [1, 0, 0, 1]
]
```

4.2.5.2.3. Status Tracking:

```
# Houdt de status van deuren en stoplichten bij
deur_status = {
    1: "gesloten",  # Begin status
    2: "gesloten"   # Begin status
}
stoplicht_status = {
    1: "rood",  # Begin status
    2: "rood"   # Begin status
}
```

4.2.5.2.4. Motor Besturing:

```
def draai_motor(deur_id, actie):
    try:
        # Selecteert juiste motor pins
        motor_pins = MOTOR1_PINS if deur_id == 1 else MOTOR2_PINS

        # Configureert pins
        setup_motor_pins(motor_pins)

        # Bepaalt richting en stappen
        direction = 1 if actie == "openen" else -1
        steps = 5048   # Volledige rotatie

        if actie == "sluiten":
            # Bij sluiten eerst rood licht
            update_stoplicht_status(deur_id, "rood")
            time.sleep(1)

        # Voert beweging uit
        step_motor(motor_pins, steps, direction)
```

```
        # Update status
        deur_status[deur_id] = actie

        # Bij openen groen licht
        if actie == "openen":
            update_stoplicht_status(deur_id, "groen")

        return True

    except Exception as e:
        logger.error(f"Fout bij het aansturen van deur {deur_id}: {str(e)}")
        raise
```

4.2.5.3. pomp_controller.py - Waterpomp Besturing

Dit bestand handelt de automatische waterpomp besturing af. Het monitort het waterniveau en schakelt de pomp aan/uit wanneer nodig.
Belangrijke onderdelen:

4.2.5.3.1. Configuratie:

```
# Instellingen voor pomp besturing
SENSOR = 'Sensor1'      # Welke sensor te monitoren
NIVEAU = 9.0            # Drempelwaarde voor waterniveau
RELAY_PIN = 14          # GPIO pin voor pomp relais
```

4.2.5.3.2. Pomp Besturing:

```
async def main():
    pump_on = False
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(RELAY_PIN, GPIO.OUT)
    GPIO.output(RELAY_PIN, GPIO.LOW)

    # Maakt WebSocket verbinding
    async with websockets.connect(ws_url) as websocket:
        async for message in websocket:
            data = json.loads(message)
            waterlevel = data.get(SENSOR)

            # Schakelt pomp aan/uit op basis van niveau
            if waterlevel < NIVEAU and not pump_on:
                GPIO.output(RELAY_PIN, GPIO.HIGH)
                pump_on = True
                print("Pomp AAN")
            elif waterlevel >= NIVEAU and pump_on:
                GPIO.output(RELAY_PIN, GPIO.LOW)
                pump_on = False
                print("Pomp UIT")
```

4.2.5.4. routing.py en urls.py - URL Configuratie

Deze bestanden configureren de URL routes voor de WebSocket en HTTP endpoints.

```
# routing.py - WebSocket routes
websocket_urlpatterns = [
    re_path(r'^ws/sensors/$', consumers.SensorConsumer.as_asgi()),
]

# urls.py - HTTP routes
urlpatterns = [
    path('deur/<int:deur_id>/', stuur_motor_aan, name='stuur_motor_aan'),
]
```

4.3. Hardware-integratie

De hardware-integratie vormt de brug tussen het mechanische systeem van de sluis en de software die het geheel aanstuurt. In dit hoofdstuk wordt besproken hoe de verschillende componenten met elkaar verbonden zijn, hoe ze samenwerken, en welke uitdagingen hierbij kwamen kijken.

4.3.1. Overzicht van de hardwarecomponenten

De belangrijkste onderdelen van de proefopstelling zijn:

- Raspberry Pi 4: centrale besturingseenheid
- 28BYJ-48 stappenmotoren met ULN2003-driver: voor het openen en sluiten van de sluisdeuren
- HC-SR04 ultrasone sensor: voor het meten van het waterniveau
- Waterpomp: voor het simuleren van het water
- Relaismodule: voor het schakelen van de pomp
- Voedingsmodule: voor het voorzien van stabiele spanning aan de motoren Testen en Resultaten

4.3.2. Beveiliging en betrouwbaarheid

Aangezien het project in contact komt met water, werd er bijzondere aandacht besteed aan de bescherming van de elektronische componenten. In een eerdere fase werd een beschermende behuizing ontworpen voor de Raspberry Pi, specifiek bedoeld voor gebruik met CAT5-kabels. Deze behuizing bood een goede fysieke bescherming, maar werd uiteindelijk niet meer gebruikt. Tijdens de ontwikkeling bleek dat het werken met CAT5-kabels te veel praktische problemen met zich meebracht, waardoor dit idee werd verlaten.

Momenteel bevindt de Raspberry Pi zich tijdelijk zonder behuizing in de opstelling. Ondanks het ontbreken van een case is de veiligheid van de elektronica gewaarborgd doordat de waterbak waarin de sluis zich bevindt volledig waterdicht is afgewerkt. Er is geen risico dat water uit de bak ontsnapt of in contact komt met de elektronica. De afdichting van de bak is zorgvuldig uitgevoerd met waterbestendige materialen en afdichtingsmiddelen, waardoor de kans op lekkage uitgesloten is.

4.3.3. Communicatie tussen hardware en software

De Raspberry Pi leest de sensorgegevens uit via Python-scripts die draaien binnen de Django-backend. Deze scripts sturen ook de motoren en de pomp aan. Dankzij Django Channels en Daphne kunnen deze acties in realtime worden uitgevoerd via de webinterface.

4.4. Mogelijke uitbreidingen

4.4.1. Integratie van camera's en AI-gestuurde detectie

Een logische en waardevolle uitbreiding van dit project is het toevoegen van camera's aan de sluisopstelling. Met behulp van één of meerdere camera's kan de sluis visueel gemonitord worden vanop afstand. Dit verhoogt niet alleen de veiligheid, maar maakt het ook mogelijk om het systeem verder te automatiseren.

Een volgende stap is het inzetten van kunstmatige intelligentie (AI) om objectherkenning toe te passen op de camerabeelden. Zo kan een AI-model getraind worden om boten te detecteren die de sluis naderen. Op basis van deze informatie kan het systeem automatisch beslissingen nemen, zoals het openen van de juiste sluisdeuren en het aanpassen van het waterniveau. Dit zou het werk van de sluismeester aanzienlijk verlichten en het proces versnellen.

Het is echter belangrijk om bij de implementatie van AI ook te voorzien in een failsafe-mechanisme. Indien de AI faalt of onverwacht gedrag vertoont, moet de sluis nog steeds handmatig bediend kunnen worden. Dit is essentieel om de continuïteit van het scheepvaartverkeer te garanderen en economische schade door stilstand te vermijden.

4.4.2. Gebruik van robuustere materialen

Hoewel het huidige prototype grotendeels vervaardigd is uit PLA, een materiaal dat geschikt is voor demonstratiemodellen, zijn er beperkingen op vlak van duurzaamheid en waterbestendigheid. Voor een langdurige of realistischere toepassing zou het gebruik van robuustere materialen zoals PETG, ABS of zelfs nylon een grote meerwaarde bieden.

Deze materialen zijn beter bestand tegen vocht, temperatuurschommelingen en mechanische belasting. Door over te schakelen naar sterkere materialen kan de levensduur van de sluis verhoogd worden en wordt de werking betrouwbaarder, vooral in een omgeving waar water en beweging een constante rol spelen.

4.4.3. Behuizing voor de Raspberry Pi

Op dit moment bevindt de Raspberry Pi zich zonder beschermende behuizing in de opstelling. Dit vormt een risico, aangezien de Pi zich in de nabijheid van water bevindt. Een spat of lek kan ernstige schade veroorzaken aan de elektronica en het hele systeem onbruikbaar maken.

Een volgende stap in de afwerking van het project is het ontwerpen en bouwen van een waterdichte behuizing voor de Raspberry Pi. Deze behuizing kan bijvoorbeeld 3D-geprint worden in PETG of ABS en voorzien worden van rubberen afdichtingen rond de kabeldoorvoeren. Dit verhoogt de veiligheid van het systeem aanzienlijk en maakt het geheel robuuster en betrouwbaarder voor demonstraties of langdurig gebruik.

5 Testen en Resultaten

5.1. Wat werkte goed?

5.1.1. Frontend

De frontend werkte naar behoren en voldeed aan de verwachtingen. De gebruikersinterface is overzichtelijk en gebruiksvriendelijk, waardoor een nieuwe gebruiker (zoals een sluismeester) snel vertrouwd raakt met het systeem. Dankzij de visueel aantrekkelijke opmaak is het bedienen van de sluis niet alleen functioneel, maar ook aangenaam voor het oog.

De verwerking van realtime gegevens via Web Sockets verloopt vlot. De waterniveaus worden correct weergegeven en geüpdatet zonder merkbare vertraging. Daarnaast worden API-aanroepen voor het aansturen van de sluisdeuren correct verstuurd en verwerkt. De combinatie van visuele feedback en directe interactie zorgt voor een intuïtieve gebruikerservaring.

5.1.2. Backend

De backend presteerde sterk, vooral op het vlak van de API voor het aansturen van de sluisdeuren. Deze functionaliteit werd tijdens de lessen grondig behandeld en meermaals geoefend, wat resulteerde in een goed gestructureerde en betrouwbare implementatie.

De API voldoet aan de gestelde vereisten en werkt stabiel. Dankzij de duidelijke documentatie en de modulaire opbouw kon de backend eenvoudig uitgebreid worden met extra functionaliteiten.

5.2. Wat moest aangepast worden?

5.2.1. Backend

Het opzetten van de Web Socket-communicatie via Django Channels en Daphne bleek een van de meest uitdagende onderdelen van het project. Hoewel de documentatie van Django Channels beschikbaar is, was de praktische implementatie complexer dan verwacht.

Er werd veel tijd besteed aan het correct configureren van de Web Socket-server. Vooral de samenwerking tussen Django, Channels en Daphne zorgde voor moeilijkheden. Tijdens het proces moesten meerdere keren herinstallaties van Django en gerelateerde pakketten worden uitgevoerd, wat leidde tot frustratie en tijdverlies.

De problemen lagen voornamelijk bij:

- Het correct instellen van het ASGI-configuratiebestand.
- Het koppelen van de juiste routing voor Web Sockets.
- Het oplossen van compatibiliteitsproblemen tussen versies van Django, Channels en Daphne.

Na een langdurig en iteratief proces is het uiteindelijk gelukt om een stabiele Web Socket-verbinding op te zetten. Deze ervaring heeft niet alleen bijgedragen aan een beter begrip van asynchrone communicatie in webapplicaties, maar ook aan het ontwikkelen van probleemoplossende vaardigheden onder druk.

5.2.2. Design

5.2.2.1. Waterdichtheid en elektronische veiligheid

Het bouwen van een werkende proefopstelling van een sluis bleek van bij de start een complexe opdracht. Tijdens het ontwikkelproces werd duidelijk dat er met veel factoren rekening gehouden moest worden, vooral met betrekking tot het gebruik van water in combinatie met elektronische componenten. Een van de grootste uitdagingen was het garanderen van een volledig waterdichte omgeving, om schade aan de elektronica te voorkomen.

5.2.2.2. Precisie bij het bouwen van de sluisdeuren

Daarnaast was het bijzonder moeilijk om de functionele onderdelen van de sluis, zoals de deuren, op schaal te realiseren. De vereiste precisie en nauwe toleranties zijn moeilijk haalbaar op kleine schaal, zeker met beperkte middelen. Er werd gestreefd naar een zo goed mogelijke passing van de deuren, wat redelijk goed gelukt is. Toch laat de sluis nog een kleine hoeveelheid water door onder de deuren. Dit wordt gecompenseerd door de pomp, die het waterniveau actief op peil houdt.

5.2.2.3. Prototyping en handmatige ondersteuning

Om tot een werkbare oplossing te komen, zijn meerdere prototypes van de sluisdeuren ontworpen en getest. Uiteindelijk werd aanvaard dat een minimale lekkage onder de deuren onvermijdelijk was binnen de beschikbare tijd en middelen. Een bijkomend probleem was dat de deuren moeilijk naar beneden zakten door de zeer nauwe passing. Door tijdsdruk en de gevorderde staat van het project werd besloten om dit niet volledig mechanisch op te lossen. In de huidige versie wordt de neerwaartse beweging van de deuren handmatig ondersteund. Hoewel dit niet ideaal is, laat het wel toe om de werking van de sluis correct te demonstreren.

5.2.2.4. Gebruik van netwerkkabels (CAT5)

Aan het begin van het jaar werd, op aanraden van Dhr. De kesel, het idee geopperd om netwerkkabels (CAT5) te gebruiken om alle sensoren en signalisatiecomponenten (zoals stoplichten) aan te sluiten op de Raspberry Pi. Dit had als voordeel dat de componenten hot-swappable zouden zijn: bij een defect kon een sensor eenvoudig vervangen worden zonder de hele bekabeling aan te passen.

Hoewel dit idee in theorie zeer interessant was, bleek het in de praktijk moeilijk uitvoerbaar. De combinatie van CAT5-kabels met het breadboard en de schakelingen zorgde voor onbetrouwbare verbindingen. Sommige kabels gaven geen signaal door, ondanks dat ze correct waren aangesloten en visueel in orde leken. Na langdurig testen werd besloten om dit systeem te verlaten.

5.2.2.4.1. Oplossing met Dupont-kabels

De netwerkkabels werden vervangen door zelfgemaakte Dupont-kabels. Hoewel deze oplossing het geheel er iets minder netjes doet uitzien, werkt het systeem nu betrouwbaar. Bovendien blijft het voordeel van hot-swappable componenten behouden: sensoren kunnen nog steeds eenvoudig losgekoppeld en vervangen worden. Dit maakt het systeem onderhoudsvriendelijk en duurzaam, aangezien enkel het defecte onderdeel vervangen hoeft te worden, en niet de volledige bekabeling. Dit draagt bij aan minder elektronisch afval en een meer ecologische aanpak.

6 Besluit

Het eindresultaat van dit project is bijzonder positief. Er werd gedurende het schooljaar veel tijd en inzet besteed aan de ontwikkeling van de miniatuursluis. Hoewel het aanvankelijk moeilijk te bevatten was hoeveel werk dit project zou vergen, werd gaandeweg duidelijk dat de vele technische uitdagingen en onverwachte problemen een grote invloed hadden op de tijdsinvestering. Elk probleem moest zorgvuldig worden geanalyseerd en opgelost, wat het leerproces alleen maar versterkte.

Het project was niet alleen technisch uitdagend, maar ook bijzonder leerrijk en boeiend om uit te voeren. Er werd veel bijgeleerd over hoe een project gestructureerd aangepakt moet worden. Zo werd het belang duidelijk van eerst een prototype op een breadboard te bouwen en dit grondig te testen, alvorens over te gaan tot de definitieve implementatie. Deze aanpak bleek ook nuttig bij het programmeren: in plaats van alles in één keer te schrijven, werd er gewerkt in afzonderlijke modules, zoals het uitlezen van sensoren en het aansturen van motoren. Dit maakte het eenvoudiger om fouten op te sporen en te debuggen.

Op programmeervlak werd er veel vooruitgang geboekt. Er werd geleerd hoe een frontend opgebouwd wordt met React, hoe routing tussen pagina's werkt, en hoe een visueel aantrekkelijke interface gecreëerd kan worden. Hoewel er al enige voorkennis aanwezig was, zijn er tijdens dit project toch nog nieuwe inzichten opgedaan die de vaardigheden verder hebben verdiept.

Dit project is het eerste binnen de zes schooljaren dat we volledig werkend werd gerealiseerd. Eerdere projecten waren technisch goed uitgewerkt, maar bereikten vaak net niet het einddoel, bijvoorbeeld door moeilijkheden met elektronica of soldeerwerk. Doorheen de jaren werd echter elk jaar bijgeleerd, wat uiteindelijk geleid heeft tot een succesvolle en functionele eindopstelling.

6.1. Duurzaamheid en herbruikbaarheid

Tijdens de ontwikkeling van dit project werd ook aandacht besteed aan duurzaamheid en herbruikbaarheid van componenten. Door te kiezen voor modulaire bekabeling met Dupont-kabels, kunnen sensoren en andere onderdelen eenvoudig vervangen worden zonder dat de volledige bekabeling opnieuw moet worden aangelegd. Dit maakt het systeem onderhoudsvriendelijk en verlengt de levensduur van de opstelling.

Bovendien draagt deze aanpak bij aan het verminderen van elektronisch afval. In plaats van complete kabelbomen of printplaten te vervangen bij een defect, kan nu enkel het beschadigde onderdeel worden vervangen. Dit is niet alleen economisch voordelig, maar ook ecologisch verantwoord.

Ook het gebruik van 3D-geprinte onderdelen draagt bij aan herbruikbaarheid. Indien een onderdeel beschadigd raakt of verbeterd moet worden, kan het eenvoudig opnieuw worden geprint zonder dat het volledige systeem aangepast hoeft te worden. Deze flexibiliteit maakt het systeem geschikt voor verdere ontwikkeling en experimentatie, zonder onnodige verspilling van materialen.

6.2. Korte samenvatting

Dit project was zowel leerrijk als plezierig om te maken. Ondanks de vele tegenslagen en technische obstakels, is het eindresultaat – een werkende miniatuursluis – iets om trots op te zijn.