

Smart Fire Extinguisher - Tussentijds Verslag

Academiejaar 2022 – 2023

TEAM 6: Anna-Laura, Emile, Jérôme, Jesse

Inleiding

Bij de brandbestrijding in grote warenhuizen worden momenteel **sprinklers** gebruikt. Deze zijn vastge maakt aan het plafond van het gebouw en zijn aan de waterleiding aangesloten om bij brand water te doen neerdalen. Ze werken heel efficient, maar hebben wel enkele grote nadelen. De voornaamste zijn de kost van de aanleg en het onderhoud van alle leidingen die de sprinklers van water voorzien. Ook de verhoogde kans op waterschade bij het springen van een van de vele waterleidingen maakt sprinklers ietwat minder aantrekkelijk.

Daarom zijn wij op zoek gegaan naar een efficiëntere manier om branden te blussen. Een **Smart Fire Extinguisher**, die zelf de branden kan detecteren, localiseren en gericht kan blussen. Zo zou 1 apparaat (met dus maar 1 aansluiting op de waterleiding of een eigen waterreservoir) een groot oppervlakte brandveilig kunnen maken. Dit zou het veel goedkoper maken voor de eigenaar die geen eindeloos lange waterleidingen moet aanleggen en onderhouden. De totale kost voor grote warenhuizen zou dus veel lager liggen en de kans op waterschade bij het springen van waterleidingen is veel kleiner.

1 Ontwerp

1.1 Klantenvereisten

De klant verwacht een apparaat dat zelfstandig branden kan detecteren en deze kan blussen. Hiervoor moet het de exacte locatie van de brand kunnen vaststellen en de arm in de juiste richting richten. Het apparaat moet water vanuit een jerrycan in de richting van de brand spuiten en zelf stoppen wanneer de brand geblust is.

Alles moet automatisch werken, maar er moet ook een manuele override zijn waarbij het apparaat volledig manueel kan worden bestuurd en worden uitgeschakeld. Al dit moet gebeuren in communicatie met een PC.

1.2 Ontwerpspecificaties

- Branden detecteren en blussen in een rechthoek van 7m x 6m op 3m afstand
- Maximale uitwijking horizontaal: 90°
- Maximale uitwijking verticaal: 90°
- Minimale spuitdruk: ...
- Hoeveelheid water per blussing: (wordt nog bekend gemaakt, foutenmarge nog inrekenen)
- Hoeveelheid beschikbaar water: 10L
- Elektronica afgeschermd van water
- Massa robot: max. 20 kg

1.3 Ontwerp

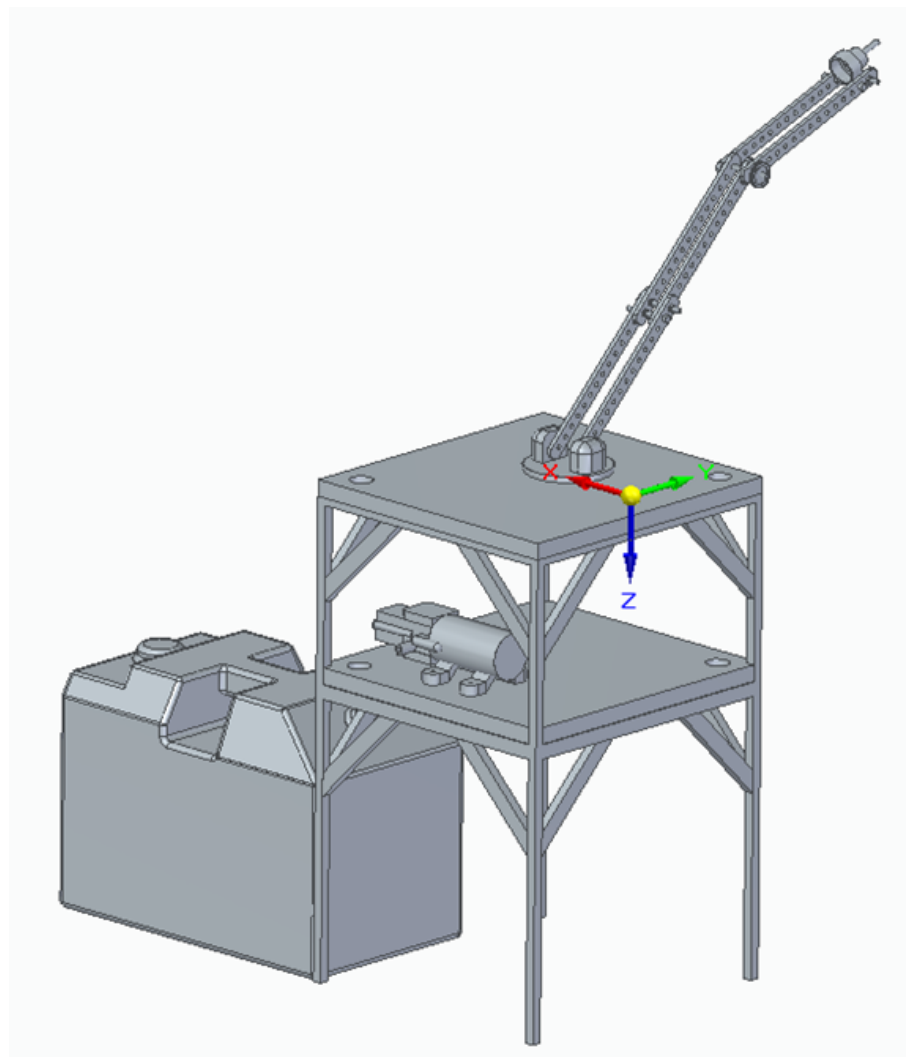
We opteren voor een **stationair brandblusplatform** dat gebruik maakt van een draaiend platform en een arm om het water in de juiste richting en onder de juiste hoek weg te spuiten. Water uit een waterreservoir zal dan met behulp van een pomp op de gewenste plaats terecht komen. De plaats zal zijn vastgesteld door een camera. Deze detecteert de brand en aan de hand van de beelden zal er berekend

worden waar en hoe ver de brand zich bevindt ten opzichte van het brandblusplatform. Daarna zullen nog een hoop berekeningen de hoek van de twee armen bepalen.

Voor **de detectie** van de brand maken we gebruik van een webcam (USB Webcam 1080P), in Python wordt een programma geschreven op een laptop die aan branddetectie en -localisatie doet. De camera is vastgemaakt op het roterend platform.

Het richten van het draaiend platform en de arm gebeurt met behulp van twee motoren (Micro Metal Gear Motor 100:1 HP), deze zullen dankzij een motor driver met de gewenste snelheid en in de juiste richting draaien. Volgens de berekeningen gedaan door een laptop en webcam zullen de armen juist gepositioneerd worden.

Wanneer de arm correct gericht is, is de laatste stap **het blussen** van de brand. Hiervoor zal de pomp (Membraanpomp 12V 4.8 bar) water uit het waterreservoir (jerrycan 10L) door een slang (met diameter 10mm) pompen, om zo weggespoten te worden richting het doelwit. De benodigde afstand halen om het volledige bestrijkingsgebied te kunnen bestrijken is mogelijk door het aanpassen van het mondstuk.



Figuur 1: Ontwerp in Solid Edge

2 Berekeningen en code

2.1 Hoek van de waterstraal

2.1.1 Berekening

De afstand zoals te zien in afbeelding 2 tussen het brandblusapparaat en de cilinder is x waarbij x maximaal gelijk kan zijn aan $10,45m$.

$$x^2 = 3^2 + 10^2 \Rightarrow x \approx 10,45 \quad (1)$$

Om deze afstand te halen berekenen we de hoek waaronder het water moet worden weggespoten. De hoogte van het platform is hier h_1 en de hoogte van de arm is $h_2 = l \sin(\theta)$ met l de lengte van de arm. y is de hoogte van de cilinder die moet worden opgevuld. Dan krijgen we:

$$\begin{cases} x = \cos(\theta)vt \\ y = h_1 + h_2 + \sin(\theta)vt + \frac{-1}{2}gt^2 \end{cases} \quad (2)$$

Bij het gebruik van de brandblusser zullen we met behulp van de camera x al kennen, dankzij de waterflowsensor zullen we v ook accuraat kennen, h_1 staat vast en h_2 hangt af van de hoek θ . Uit de twee vergelijkingen kunnen we θ en t halen en zal het doelwit gevuld geraken.

2.1.2 Code

De functie `hoekV(waterDebiet, afstandBeker)` (zie figuur 3) berekent de hoek θ die nodig is tussen het platform en de arm. Deze functie heeft daarvoor het waterdebiet, in l/min , nodig en de verticale afstand, in m , tot het doelwit. Het waterdebiet wordt meegegeven door de waterflowsensor en de afstand door de webcam. Er worden ook niet-variabele parameters gebruikt die op voorhand zijn vastgelegd, zoals de straal van het spuitgat, de hoogte van het platform, de lengte van de arm, de hoogte van het doelwit en de afstand tussen het beginpunt van de arm en de camera.

In deze functie wordt dan de snelheid van het water berekend en dat wordt in een stelsel gebruikt met twee vergelijkingen. Dit stelsel wordt, met gebruik van de `scipy`-package, opgelost naar de hoek θ en de tijd t .

$$\begin{cases} afstandBeker = \cos(\theta) * lengteArm - afstandCamera + \cos(\theta) * snelheid * t \\ hoogteBeker = hoogtePlatform + \sin(\theta) * lengteArm + \sin(\theta) * snelheid * t - 1/2 * 9.81 * t^2 \end{cases} \quad (3)$$

De hoek θ wordt omgezet van radialen naar graden en is dan de output van deze functie en kan dan gebruikt worden voor één van de motoren.



Figuur 2: Schematische voorstelling van de opstelling



Figuur 3: Flowchart van de functie `hoekV(waterDebiet, afstandBeker)`

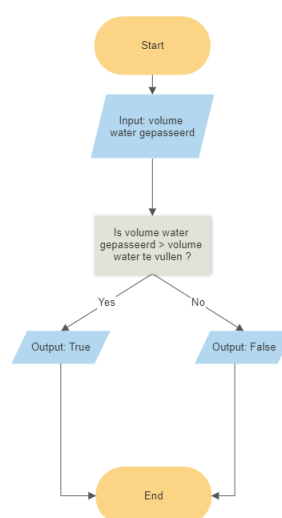
2.2 Stoppen met water spuiten

2.2.1 Berekening

Wanneer de hoeveelheid water nodig om een brand te blussen bekend wordt gemaakt en we een foutenmarge uit een handvol experimenten hebben berekend zullen we de hoeveelheid te spuiten water aan onze code kunnen toevoegen. De foutenmarge zal een gemiddelde zijn van water die naast het doelwit terecht komt bij de proeven.

Om zo exact mogelijk te weten hoeveel water het apparaat al heeft uitgespoten, maken we gebruik van de waterflowsensor.

2.2.2 Code



Figuur 4: Flowchart van de functie `stoppenPompen()`

2.3 Camera

2.3.1 Berekening

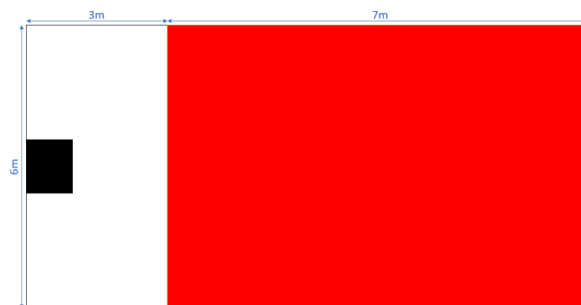
Omdat het gezichtsveld van de camera niet het volledige bestrijingsgebied omvat (zie figuur 5), hebben we besloten de camera te laten meedraaien met het draaiende platform en het bestrijingsgebied van links naar rechts te analyseren.

We kunnen de afstand d van een object tot de camera vinden met de formule 4, waarbij b_w de werkelijke breedte van een object is en b_s de breedte van het object op het beeld van de camera in cm (berekend door het aantal pixels in de x-richting te vermenigvuldigen met 0.02645833333). f is de focal lengte van de camera, dit is eigen aan ieder toestel en werd door ons experimenteel vastgesteld door dezelfde formule 4 te gebruiken, maar ze om te vormen naar f en een object op een gekende afstand te plaatsen en van meerdere waarnemingen het gemiddelde te nemen.

$$d = f * \frac{b_w}{b_s} \quad (4)$$

Een andere manier om de afstand d te berekenen is door de hoek te meten tussen de middelste pixel p_m en buitenste pixel p_b van het beeld. Dit doen we door de absolute waarde te nemen van het verschil van p_m en p_b . Deze waarde vermenigvuldigen we met gpp . gpp staat voor graden per pixel en wordt bekomen door de breedte van het gezichtsveld van de camera (hier 60°) te delen door de breedte van het beeld (640 pixels). De uiteindelijke uitkomst is de hoek α . Om daarmee de afstand te berekenen gebruiken we de formule:

$$d = \frac{b_w}{2} * \tan(\alpha) \quad (5)$$



Figuur 5: Schematische voorstelling van het bestrijingsgebied (rood) met het blusapparaat zwart gekleurd

2.3.2 Code

In python werd een programma geschreven dat de afstand tot een object kan inschatten aan de hand van camerabeelden. Het kan de beelden vastgelegd door een webcam analyseren dankzij het gebruik van cv2. Daarna zet het het beeld om van BGR (blue-green-red) naar HSV (hue-saturation-value), waardoor we een boven- en onderlimiet kunnen opleggen voor het kleur die we wensen te behouden, hier is dit rood. De output is een zwart beeld waarbij enkel objecten met de gewenste kleur met wit omlijnd zijn.

De camera is vastgemaakt op het roterend platform (zie figuur 1) waarbij het beeld parallel staat met de arm. We laten dus de camera meedraaien en analyseren zo het volledige bestrijingsgebied van links naar rechts. Wanneer een object wordt opgemerkt en de hoek waaronder het object zich bevindt tussen twee bepaalde waarden ligt (nu nog -1° en 1°), dit wil zeggen dat het in het midden van het beeld staat, wordt de afstand d tot het object berekend. Dit gebeurt op 2 manieren:

- We maken gebruik van formule 4
- We maken gebruik van formule 5

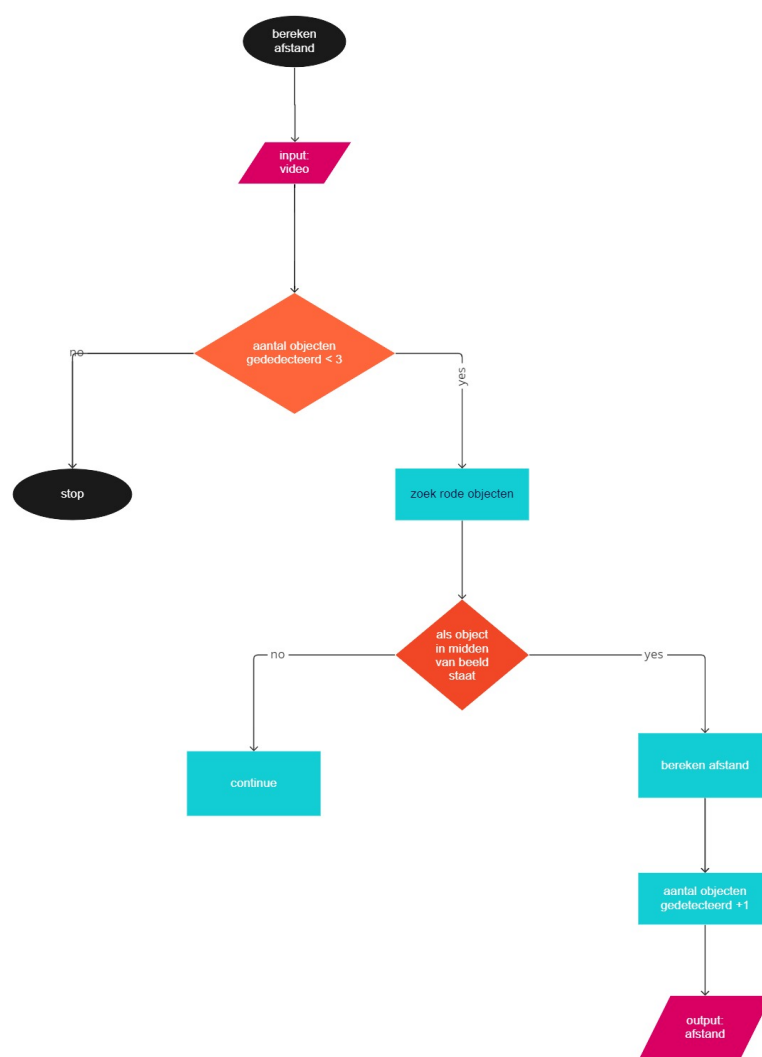
We benutten beide methoden en nemen het gemiddelde van de twee uitkomsten voor d en verkrijgen op die manier de meest accurate waarde voor d .

3 Planning

Op de pagina hieronder is onze taakstructuur te vinden. De eerste taak, het verkennen van de opdracht, is al voltooid. Ons CAD model is bijna af en we zijn al begonnen met het coderen van de microcontroller en PC alsook met de bouw van het geraamte van ons apparaat.

Code	Taak
1	OPDRACHT VERKENNEN
1.1	Brainstormen
1.2	Taakstructuur opstellen
1.3	Verantwoordelijkheidsstructuur opstellen
1.4	Teamkalender opstellen
1.5	Gantt chart opstellen
1.6	1ste vergaderverslag
1.7	Kostenraming opstellen
1.8	Onderzoeken materialen
1.9	Materiaalselectie
2	CAD MODEL
2.1	3D modellen
2.1.1	Jerrycan
2.1.2	Webcam
2.1.3	Slang
2.1.4	Motoren
2.1.5	Membraanpomp
2.1.6	Armen
2.1.7	Platform
2.1.8	Printplaat
2.1.9	Waterflowsensor
2.1.10	Powerbank
2.2	Assemblage
2.3	Technisch tekenen
2.4	Stuklijst
3	BOUW
3.1	Coderen microcontroller
3.1.1	Brandlocalisatie
3.1.2	Beweging armen
3.1.3	Relatie brandlocalisatie - beweging van armen
3.1.4	Water spuiten
3.1.5	Relatie juist gericht - water spuiten
3.2	Communicatie met PC
3.2.1	Automatische werking
3.2.2	Manuele override
3.3	Testen van onderdelen en mogelijke aanpassingen
3.3.1	Detectie brand
3.3.2	Locatievaststelling brand
3.3.3	Beweging arm 1 richting brand
3.3.4	Beweging arm 2 afstand brand
3.3.5	Spuitdruk
3.3.6	Stoppen wanneer cilinders gevuld
3.4	In elkaar steken
3.5	Testen geheel
3.6	Aanpassingen maken

4	RAPPORTERING/VERSLAG
4.1	Feedback P&O 1 lezen
4.2	Inleiding
4.3	Probleem Schetsen
4.3.1	Probleem
4.3.2	Oplossing
4.4	Ontwerp en Materialen
4.4.1	Ontwerpproces
4.4.2	Materiaalselectie
4.4.3	Solid Edge
4.5	Elektronisch Circuit
4.6	Programmeercode
4.6.1	LabView
4.6.2	Python
4.6.3	Raspberry PI
4.7	Resultaten
4.7.1	Prototype
4.7.2	Resultaten Demo
4.8	Financieel rapport
4.9	Mogelijke verbeteringen
4.10	Besluit
4.11	Bibliografie
5	POWERPOINT LATEX
5.1	Inleiding
5.2	Probleem Schetsen
5.3	Ontwerp en Materialen
5.3.1	Ontwerpproces
5.3.2	Materiaalselectie
5.5	Elektronisch Circuit
5.6	Programmeercode
5.7	Resultaten
5.7.1	Prototype
5.7.2	Resultaten Demo
5.8	Financieel rapport
5.9	Mogelijke verbeteringen
5.10	Besluit
5.11	Bibliografie
6	LABVIEW
6.1	Sensoren
6.2	Webcam
6.3	Overige code
6.4	Handmatige override



miro

Figuur 6: Flowchart van de functie `berekenen_afstand()`