KU LEUVEN kulak

Groep Wetenschap & Technologie Kulak
Ingenieurswetenschappen
P&O 2

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 vastgemaakt aan het plafond van het gebouw en zijn aan de waterleiding aangesloten om bij brand water te doen neerdalen. Ze werken heel efficiënt, 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. Bovendien is het bestrijkingsgebied bij kleine branden niet proportioneel met de grootte van de brand waardoor er ook meer waterschade kan optreden dan nodig is.

Daarom zijn wij op zoek gegaan naar een efficiëntere manier om branden te blussen. Een **Smart Fire Extinguisher**, die zelf de branden kan detecteren, lokaliseren 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.

De klant wenst van zelfsprekend een prototype waarbij de kost zo laag mogelijk ligt en het totale volume van de brandblusser zo klein mogelijk is.

1.2 Ontwerpspecificaties

De brandblusser moet de gesimuleerde branden kunnen detecteren en blussen. De branden worden gesimuleerd door drie cilinders met een hoogte van 50 cm en een diameter van 20 cm, waarop 3 rode ledlampen gemonteerd zijn die constant branden. De lampen bevinden zich op 20, 30 en 40 centimeter van de grond. De extinguisher moet dus in staat zijn om per cilinder de rode lampen te detecteren en daarbij ook de afstand tot aan die lamp te meten. De cilinders bevinden zich in een rechthoek van 6m x 7m en de brandblusser staat op 3 m van deze rechthoek.

De maximale uitwijking van onze arm is zowel in de horizontale als de verticale richting 90°, alhoewel dit in de verticale richting absoluut niet nodig zal zijn.

Er zal een 10L water beschikbaar zijn in een jerrycan waarmee we de 'branden' moeten blussen. In elke cilinder moet minstens ... cl water geschoten worden.

Daarnaast mag de robot maximaal 20 kg wegen, water inbegrepen. De effectieve brandblusser mag dus maximaal een kleine 10kg wegen.

De werking van het blusapparaat moet volledig geautomatiseerd zijn, maar het is belangrijk dat er mogelijkheid is tot manuele override. Hierbij moeten we zelf kunnen richten en spuiten met de robotarm via de computer.

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 wordt vastgesteld door een webcam. 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 gebeuren nog enkele berekeningen om de hoek van de arm te bepalen.

Voor de **detectie** van de brand maken we gebruik van dezelfde webcam (USB Webcam 1080P), in Python wordt een programma geschreven op een laptop die aan branddetectie en -lokalisatie 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. De motoren worden bestuurd door een microcontroller (Arduino nano 33 iot).

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. We zullen dus een smallere spuit gebruiken, met een diameter van 3 mm om precies te zijn.

Figuur 1: Ontwerp in Solid Edge

2 Berekeningen en code

2.1 Hoek van de waterstraal

2.1.1 Berekening

Als eerste zullen we de maximaal mogelijke afstand tussen het brandblusapparaat en de cilinder berekenen. We berekenen dus de afstand tussen het blusapparaat en de hoek van het terrein in de onderstaande formule. De maximale afstand die we dus moeten overbruggen is $10,45m$

$$x^2 = 3^2 + 10^2 \Rightarrow x \approx 10,45 \tag{1}$$

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 $h_2 = lsin(\theta)$ met l de lengte van de arm. y is de hoogte van de cilinder die moet worden opgevuld. Dan krijgen we:

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

Bij het gebruik van de brandblusser zullen we met behulp van de camera de afstand x al kennen, dankzij de waterflowsensor zullen we de snelheid van het water v ook accuraat kennen. Daarnaast staat h_1 vast en hangt h_2 af van de hoek θ. Uit de twee vergelijkingen kunnen we θ en t halen en zo correcte instructies doorgeven aan onze robotarm.

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.

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

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 Arduino

De Arduino code werkt met twee functies: de setup en de loop. Eerst wordt in een Arduino programma de variabelen die bovenaan in het programma staan gedeclareerd/geïnitialiseerd. Daarna wordt de setup-functie één keer aangeroepen. Alle code die in deze functie is geschreven, wordt uitgevoerd. Meestal worden hier de verschillende pins benoemd, waarden aan variabelen gegeven,enz. Daarna wordt de setup-functie afgesloten. Dan wordt de loopfunctie aangeroepen en wordt die code uitgevoerd, en op het einde begint hij opnieuw bij de loopfunctie en zo blijft het doorgaan.

2.2.1 setup-functie

In onze setup-functie worden de verschillende pins gedeclareerd, enkele variabelen gelijkgesteld aan 0 en wordt een serialconnectie gestart met de pythoncode. De verschillende pins bestaan uit de motoren, pomp en waterflowsensor. Er zijn twee motors en om onderscheid te maken worden ze de horizontale en de verticale genoemd, de horizontale (verticale) motor voert een horizontale (verticale) beweging uit.

2.2.2 loopfunctie

In de loopfunctie zit een stuk code die elke seconde het waterdebiet en het totale hoeveelheid gepasseerd water berekend aan de hand van de waterflowsensor. Deze code blijft constant lopen en start direct.

In de loopfunctie gebeurt er niks tot het startsignaal komt van de pythoncode. Als dat aangekomen is, start de horizontale motor te draaien aan een gegeven snelheid. Dan wordt er gewacht tot er een signaal (dat niet start is) komt van de pythoncode, die bevat de hoek die de arm met de waterslang moet maken. Als dit signaal gekregen is, start een if-lus, waarin het volgende gebeurt. Er wordt intern bijgehouden hoeveel branden er zijn gedetecteerd, er wordt dus een brand bijgeteld. De horizontale motor wordt ook gestopt, zodat onze arm in de juiste richting is georiënteerd. Daarna draait de verticale motor een aantal (milli)seconden, afhankelijk van de grote van de hoek. Ten slotte begint de pomp met water op te pompen. De pomp moet stoppen nadat een bepaalde hoeveelheid water is gepasseerd, dit is de voorwaarde van een andere if-lus in de loopfunctie. Deze hoeveelheid komt van de bovenvermelde code van de waterflowsensor. De lus start met de pomp uit te zetten en de hoeveelheid gepasseerd water weer op nul te zetten. Daarna draait de verticale motor terug naar zijn startpositie, door evenveel (milli)seconden te draaien aan dezelfde snelheid maar in tegengestelde zin. Hierna staat een sub-if-lus met als voorwaarde dat het aantal branden kleiner moet zijn dan 3. Als deze voorwaarde voldaan is, start de horizontale motor weer met draaien en is het weer wachten tot een signaal komt van de pythoncode. Als er al meer

branden zijn gedetecteerd, gebeurt er niks. Heel het systeem zal blijven stilstaan en alleen de code van de waterflowsensor zal blijven doorgaan. Wij hebben voor 3 gekozen omdat de de opdracht is om drie branden te blussen, maar dit kan makkelijk aangepast worden naar 1 of een keuze die afhangt van de hoeveelheid beschikbaar water.

2.3 Camera

2.3.1 Berekening

Omdat het gezichtsveld van de camera niet het volledige bestrijkingsgebied omvat (zie figuur 4), hebben we besloten de camera te laten meedraaien met het draaiende platform en het bestrijkingsgebied 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 dor 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.

$$d = f * \frac{b_w}{b_s} \tag{4}$$

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:

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

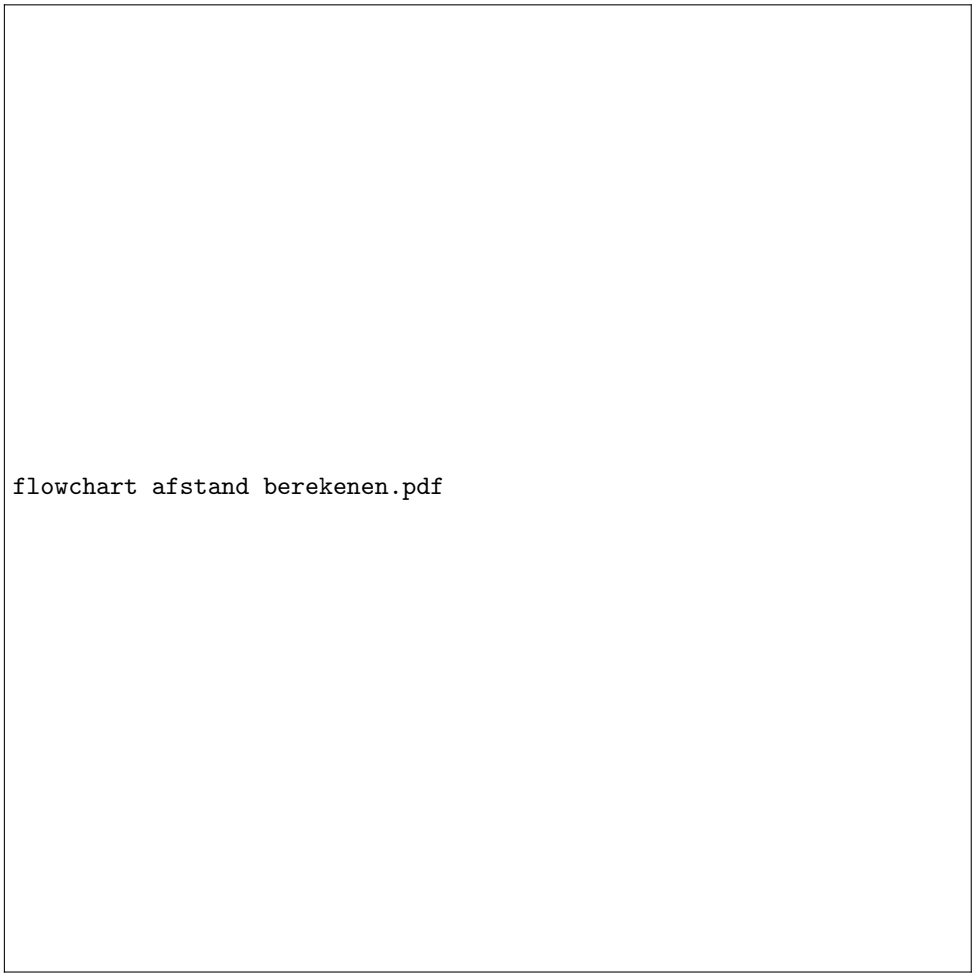
Figuur 4: Schematische voorstelling van het bestrijkingsgebied (rood) met het blusapparaat zwart gekleurd

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 bestrijkingsgebied van links naar rechts. Wanneer een object wordt opgemerkt en de hoek waaronder het object zich bevind 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.



Figuur 5: Flowchart van de functie `berekenen_afstand()`

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. We hopen met een prototype klaar te zijn tegen 28 april 2023, zodat we daarna kunnen beginnen met testen en nodige aanpassingen kunnen maken. width=!,height=!,