Antwerp Maritime Academy

Aansturen van een BLDC motor met PID en fuzzy logic

Artiom Samoilov

Scriptie voorgedragen tot het behalen
van de graad van
Bachelor in de Scheepswerktuigkunde

Promotor: Raf Maes
Co-promotor: Pascal Bouquet
Academiejaar: 2021-2022

Voorwoord

Als eindwerk van het 3de bachelor jaar in de scheepswerktuigkunde voer ik deze scriptie uit. Deze scriptie gaat over het besturen van een brushless DC motor met PID en fuzzy logic regeling en de vergelijking tussen de twee.

Samen met mijn promotor Raf Maes is er beslist om een scriptie te schrijven over fuzzy logic. Na de beslissing is de brushless DC motor gekozen als medium om te regelen en de twee regeltechnieken te vergelijken.

Ik wil Raf Maes en Pascal Bouquet bedanken voor hun begeleiding en medewerking om deze scriptie tot een goed einde te brengen.

Bij deze hoop ik dat er meer aandacht komt voor fuzzy logic in de vorm van andere scripties.

Artiom Samoilov

Samenvatting

Deze scriptie gaat over het regelen van een brushless DC motor met PID en fuzzy logic regelaars. Brushless DC motoren worden in veel toepassingen gebruikt zoals huishoudtoestellen en dus worden zij ook geregeld op hun parameters. Fuzzy logic is gekozen als een van de regelstrategiën omdat het een minder bekende regeltechniek heeft die toch een interessante werking heeft. Het doel is om te zien welke regeltechniek het beste is om een BLDC motor te regelen en tegelijkertijd aandacht brengen voor fuzzy logic.

Om de vergelijking te maken tussen beide regeltechnieken is er een windtunnel gemaakt met daarin een sensor die het gegenereerde luchtdebiet zal meten en omzetten in een spanning. Deze spanning wordt gestuurd naar een Arduino microcontroller board die de berekeningen zal uitvoeren en de motor zal besturen. Een Arduino platform is ideaal voor deze scriptie omdat het goedkoop is en software bevat waarmee we de regelaars kunnen programmeren.

Uiteindelijk is gebleken dat beide types regelaar de motor met succes kunnen regelen. Uiteraard zijn er verschillen zoals dat de fuzzy logic regelaar sneller stabiliseert maar de motor meer zal belasten. De beste regeltechniek hangt dus af van de gewenste toepassing.

Abstract

This thesis is about controlling a brushless DC motor with PID and fuzzy logic controllers. Brushless DC motors are used in many applications such as household appliances. Thus, they will be subject to control systems. Fuzzy logic has been chosen as one of the control systems because it is a less known control system that still has an interesting way of operating. The goal is to see which control system is best to control a brushless DC motor and at the same time bring attention to fuzzy logic.

For comparison purposes, a wind tunnel was created containing an airflow sensor that will measure the generated air flow speed and convert it into a voltage. This voltage is sent to the Arduino which will perform the calculations and control the motor. The Arduino platform is ideal for this thesis because it is inexpensive and contains software that allows us to program the controllers.

In the end, it was found that both regulators can successfully regulate the motor. Of course, there are differences such as that the fuzzy logic controller stabilizes faster but will put more load on the motor. So the best control technique depends on the desired utilization.

Inhoud

Figurenlijst

Tabellenlijst

Inleiding

De BLDC motor is een van de motor typen die tegenwoordig meer en meer gebruikt wordt. Deze motor type wordt toegepast in onder andere huishoudtoestellen, luchtvaart, ziekenhuizen en industriële automatisatie (Yedamale, 2003). Voor deze toepassingen is het nodig om gebruik te maken van regeltechnieken. PID en fuzzy logic regeling verschillen sterk van elkaar maar hebben hetzelfde einddoel namelijk het systeem te regelen. Er moet dus een keuze gemaakt worden tussen de twee. Volgens (Knospe, 2006) gebruiken over 90% van de regelkringen PID. Dit betekent dat fuzzy logic regeling weinig gebruikt wordt. De reden hiervoor is omdat het minder bekend is dan PID regeling en het is ook complexer. Om te kijken hoe complex het is en welke regeling beter is voor een BLDC motor is er in deze scriptie een regelkring ontworpen. In deze regelkring is de motor bestuurd door een Arduino microcontroller met twee verschillende regelaars namelijk PID en fuzzy logic. De motor zal geregeld worden op basis van de luchtstroom die de motor genereert. De sensor meet de luchtstroom en stuurt de informatie naar de Arduino die vervolgens de motor snelheid zal aanpassen. Een vergelijking maken tussen de twee regelaars is mogelijk door naar het gedrag van de motor en sensorspanning te kijken.

Probleemstelling

We weten niet wat de beste regelaar is voor de BLDC motor in onze opstelling. We kunnen daardoor niet weten wat het meest stabiele en correcte regeling zal zijn. De uitdaging zal zijn om onze regelaars zodanig te programmeren dat we een stabiele regeling krijgen van de luchtstroom. Eenmaal we de luchtstroom kunnen regelen is het nog altijd nodig om een vergelijking te maken tussen de twee regelaars. Alleen dan kunnen we een antwoord krijgen op al onze onderzoeksvragen namelijk:

Wat is de beste regeltechniek om een BLDC motor te regelen?

In welke mate verschilt de PID en fuzzy logic regeling bij een BLDC motor?

Doelstelling

De doelstelling van deze scriptie is om een beter inzicht te krijgen bij het regelen van een BLDC motor. We willen ook een beter begrip krijgen over PID en fuzzy logic regeling. Er zal onderzocht worden of we onze gewenste waarde krijgen en hoe de motor dat levert. De evolutie van de gewenste waarde zal vergeleken worden bij de twee regelaars om het verschil te zien in stabiliteit van het systeem.

Methode

We vetrekken vanuit het idee of we een BLDC motor kunnen sturen met fuzzy logic. Er werd bekeken of het wel mogelijk is om met een Arduino zowel PID als fuzzy logic te implementeren.

Om een effectieve regeling te krijgen hebben we metingen nodig. Er werd gekozen voor een luchtstroom sensor. Een BLDC kan een hoog toerental bereiken. Als we daarop twee rotorbladen plaatsen krijgen we een propeller die een luchtstroom genereert.

De sensorspanning werd gemeten met een multimeter om te weten wat de grenzen zijn van de sensor. Een anemometer is ook gebruikt om de effectieve windsnelheid te meten die de motor kan leveren. Na een zeker toerental gebeurt de stijging van windsnelheid traag in vergelijking met het begin. Er is besloten om tot dat bepaald toerental te werken.

Beide regelstrategieën zijn geschreven in twee programma's in Arduino software. De programma's krijgen aanpassingen om de stabiliteit van de sensorspanning te verhogen. De twee regel strategieën geven een trend van de sensorspanning en de puls breedte voor de motor. Een vergelijking tussen de twee regel strategieën is dan mogelijk.

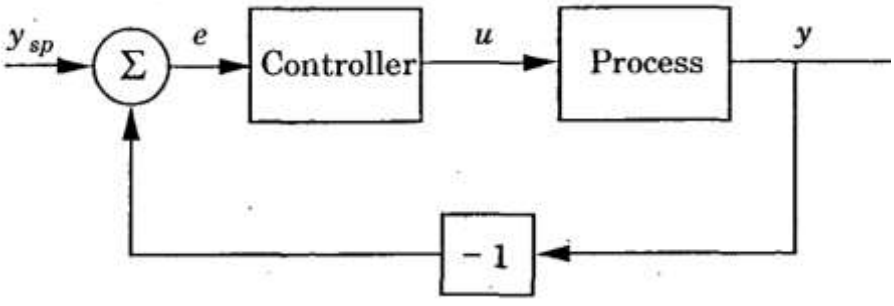
1 PID

Vooraleer we de vergelijking kunnen maken tussen de twee regeltechnieken moet er eerst een fundamenteel begrip tot stand komen van PID regelaars en fuzzy logic regelaars. In dit hoofdstuk wordt uitgelegd wat PID is en hoe het in zijn werking gaat.

PID staat voor Proportioneel, Integrerend, Differentiërend

1.1 Feedback systeem

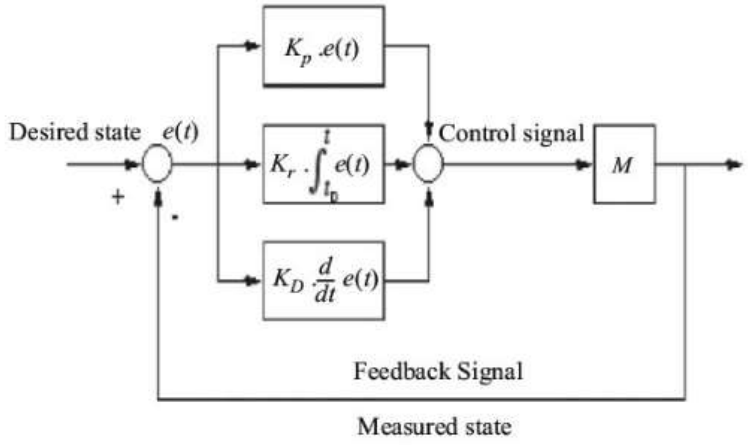
De basis van alle PID controllers ligt in het feit dat het een feedback systeem heeft. Afgebeeld in figuur 1 is een blokdiagram van een feedback systeem. In dit systeem is de relatie tussen de blokken getekend met pijlen. Het proces heeft een input aangeduid door *u.* De output van het proces is aangeduid door *y* en heet de *process variable* (PV). Dit is de waarde die wordt gemeten door de sensor. De uiteindelijke waarde die we willen bereiken is de *setpoint* (SP), aangeduid door y_{sp}. De fout of het verschil tussen de PV en de SP is aangeduid door *e* waaruit volgt dat $e = y_{sp} - y$.
Dankzij de feedback is dit een gesloten systeem. Dit geeft het systeem de mogelijkheid om de process variabele zo dicht mogelijk te houden bij de setpoint. Stel dat wanneer het systeem zich in evenwicht bevindt en er treedt een storing op waarbij de gemeten waarde groter wordt dan de setpoint. We krijgen een negatieve fout en de controller zal de fout wegwerken door de input *u* van het proces te verminderen wat als gevolg geeft dat de gemeten waarde verkleint en de afwijking dichter bij nul komt (Åström & Hägglund, 1995).



Figuur 1 Blokdiagram feedback systeem
Bron: Åström & Hägglund (1995)

1.2 PID regelaar

De PID regelaar is een uitbreiding op de simpele feedback systeem zoals te zien is op figuur 2. Het is de meest gebruikte regeltechniek in de wereld met meer dan 90% van regelaars die het gebruiken (Knospe, 2006). De eerste reden waarom het zo veel gebruikt wordt is omdat het vrij is eenvoudig is. Een werknemer hoeft alleen maar de essentie van PID te verstaan en heeft geen wiskundige achtergrond nodig om ermee te werken. De tweede reden: De PID controller is geen nieuwe technologie. Voor de Tweede Wereldoorlog heeft (Minorsky., 1922) geschreven over PID regelaars die gebruikt kunnen worden aan boord van Amerikaanse oorlogsschepen. Dat wil zeggen dat voor de oudere generatie van ingenieurs PID regelaars niet nieuw zijn en kunnen zij met deze regeltechniek vertrouwd zijn.



Figuur 2 Blokdiagram PID regelaar
Bron: Bansal (2009)

1.2.1 De drie parameters

Dit zijn de parameters dat de PID regelaar onderscheidt van een gewone feedback systeem.

- Proportioneel
- Integrerend
- Differentiërend

De proportionele parameter zorgt voor stabiliteit. De parameter kijkt naar de fout op het moment zelf en onmiddellijk zal reageren. Het wordt ook de proportionele versterkingsfactor genoemd.

De integrerende parameter zorgt voor nauwkeurigheid. Zoals de naam het zegt gaat deze parameter een integraal nemen tot het tijdstip *t* wat het heden is. Deze variabele is dus een accumulatie van de fout van het verleden.
De differentiërende parameter zorgt voor snelheid. Het neemt een afgeleide van de fout op het tijdstip *t*. Met de afgeleide kunnen we een voorspelling maken van hoe de fout zich kan evolueren als er niks gaat veranderen. Met deze informatie zal de differentiërende variabele de fout preventief verkleinen (Unbehauen, 2009).

Met deze drie parameters betekent het dat de PID regelaar een regelaar is die het verleden, het heden en de toekomst in acht neemt.
Met de drie parameters en met formule (1) is de output van de PID berekenen mogelijk.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt} \quad (1)$$

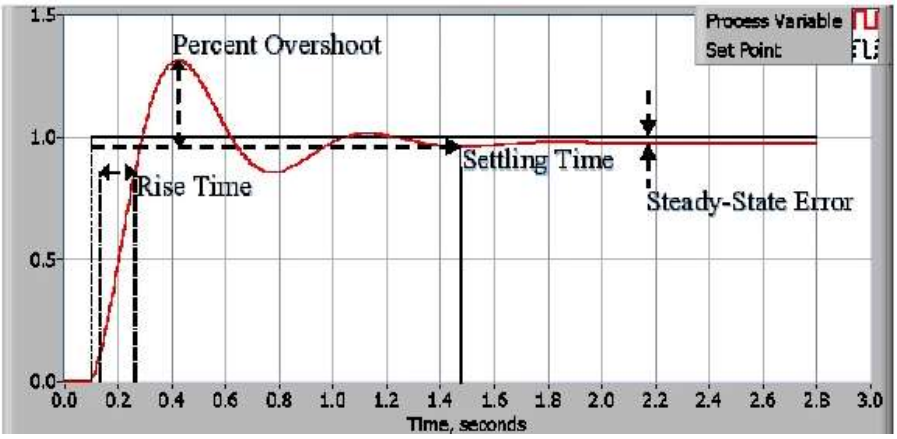
Bron: ctms (z.d.)

Waarbij:

u	stuursignaal
e	foutsignaal
t	tijd
K_p	proportionele versterkingsfactor
K_i	integrerende versterkingsfactor
de	foutsignaal verandering
dt	tijd verandering

1.2.2 Effect van de drie parameters op het systeem

De gevolgen weten van de drie parameters is essentieel voor het *tunen* van een PID regelaar. Vooraleer het effect van de drie parameters kan worden besproken is het nodig om de begrippen van de respons curve te begrijpen. Die worden besproken door (Tay, Mareels, & Moore, 1998).



Figuur 3 Respons curve
Bron: NI (2020)

- Reactie tijd (*Rise time*): De tijd dat de gemeten waarde de setpoint voor de eerste keer bereikt. Dit kan men zien als de reactietijd van het systeem. Een korte rise time betekent een systeem dat snel reageert tegenover een systeem dat traag reageert met een lange rise time.
- Doorschot (*Overshoot*): De maximale waarde van de respons curve.
- Uitslingertijd (*Settling time*): De tijd die nodig is zodat de gemeten waarde 5% afwijkt van de setpoint.
- Stabiele-toestand fout (*Steady-State Error*): De blijvende afwijking wanneer het systeem in stabiele toestand bevindt bij een tijd die tot oneindig gaat.

Nu de begrippen uitgelegd zijn kunnen we de gevolgen van de drie parameters bekijken in tabel 1 als we ze individueel veranderen.

Tabel 1 Effect van het veranderen van de PID parameters
Bron: bewerkt van Kiam Heong Ang, Chong, & Yun Li (2005)

Parameter	Reactie tijd	Doorschot	Uitslingertijd	Stabiele-toestand fout	Stabiliteit
K_p vergroten	Vermindering	Vergroting	Kleine vermindering	Vermindering	Verlaging
K_i vergroten	Kleine vermindering	Vergroting	Vergroting	Grote vermindering	Verlaging
K_d vergroten	Kleine vermindering	Vermindering	Vermindering	Kleine verandering	Verbetering

1.2.3 Tuning

Tabel 1 geeft duidelijk weer welk effect elke parameter individueel heeft op het systeem. Het doel is natuurlijk om een stabiel systeem te krijgen die zo goed als geen fout heeft. Om dat te krijgen moeten we de juiste waarden hebben voor onze drie parameters. Dit proces heet *tuning*. Een van de simpelste tuning methode is de *trial-and-error* methode, de waarden beetje bij beetje veranderen totdat je een goed resultaat krijgt. Een andere methode is de Ziegler-Nichols methode.
Bij de Ziegler-Nichols methode moet je eerst je integrerende- en differentiërende versterkingsfactor op nul zetten. Vervolgens moet de proportionele versterkingsfactor verhoogd worden totdat die zijn waarde bereikt waarbij de respons curve oscilleert, aangeduid met K_u. Je meet dan de periode van je respons curve, aangeduid met T_u. Tabel 2 geeft de nodige berekeningen weer voor de regelaar die je wilt tunen (Ellis, 2012).

Tabel 2 Ziegler-Nichols Methode
Bron: bewerkt van Ellis, (2012)

	K_p	K_i	K_d
P regelaar	$0.5*K_u$	0	0
PI regelaar	$0.45*K_u$	$T_u/1.2$	0
PID regelaar	$0.6*K_u$	$T_u/2$	$T_u/8$

De methode zal ervoor zorgen dat je systeem stabiliseert. Maar er moet rekening gehouden worden dat dit een zeer agressieve tuning methode is en kan als gevolg leiden tot een grotere doorschot. Het is dan een minder ideale tuning methode voor regelaars in de industrie waar een grote doorschot niet geaccepteerd wordt.

1.2.4 Transfer functie

Als we een systeem stabiel willen houden met een PID regelaar moeten we de drie variabelen weten. We kunnen de best passende parameters vinden door ze beetje per beetje veranderen of een stabilisatie methode toe te passen (*tuning*). Maar dat kan niet altijd gebeuren. Het kan zijn dat het systeem waarin de regelaar werkt te gevaarlijk is om te kijken of dat de parameters het systeem tot stabiliteit brengt. Daarom is er de transfer functie.

Transfer functies zijn wiskundige modellen die een uitkomst geeft voor elke mogelijke inputwaarde (Laughton & Warne, 2003). Dit betekent dat de transfer functie kan zeggen wanneer het systeem stabiel zal zijn en er geen noodzaak is aan *trial and error*, zeker als het een systeem is waar er absoluut geen onstabiliteit toegelaten is.
Om aan een transfer functie te komen moeten we een Laplace transformatie uitvoeren. De Laplace transformatie transformeert een differentiaal vergelijking om naar een algebraïsche vergelijking. Als we dit toepassen bij vergelijking (1) krijgen we:

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \quad (2)$$

Bron: ctms (z.d.)

2 Fuzzy logic

Vooraleer we ons gaan verdiepen in fuzzy logic is het best om een voorbeeld te geven waar het handig kan zijn.
Stel je voor dat er een kraan is en die neemt zandkorrels een per een van een berg zand en smijt die weg. De vraag is nu wanneer kunnen we zeggen dat de berg geen berg is. Bij booleaanse logic stopt de berg zandkorrels een berg te zijn wanneer alle zandkorrels weggesmeten zijn. Maar als je het aan een mens vraagt dan zal die antwoorden dat de berg maar bijvoorbeeld voor de helft een berg is, of het is een beetje een berg en meer een heuvel geworden (Bhattacharyya & Dutta, 2012). Een computer verstaat woorden zoals: beetje, veel, half vol niet omdat ze niet bestaan in de booleaanse logic. Daar heb je maar twee uitkomsten: Waar of Niet waar.
Maar met fuzzy logic kunnen we een computer leren denken zoals een mens en met gevolg ook regelen met een menselijke denkwijze.

2.1 Fuzzy set theorie

Fuzzy logic werd eerst voorgesteld door professor Zadeh in zijn geschreven onderzoek *Fuzzy sets* (1965). De basis van fuzzy logic regelaars ligt in de fuzzy sets of wazige verzamelingen. In de wiskunde heb je verzamelingen. Ofwel hoort een getal bij die verzameling ofwel niet. Bij

een wazige verzameling behoort een getal maar voor een deel bij een verzameling. Het is een vrij breed concept, maar hier houden we het bij de basis.

Een fuzzy set wordt als volgt genoteerd:

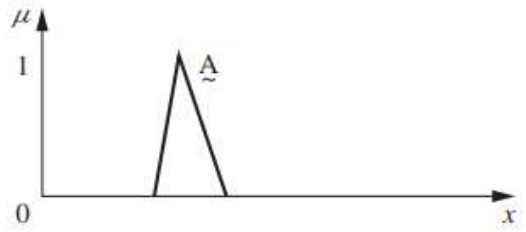
$$\underset{\sim}{A} = \left\{\frac{\mu_{\underset{\sim}{A}}(x_1)}{x_1} + \frac{\mu_{\underset{\sim}{A}}(x_2)}{x_2} + \cdots\right\} = \left\{\sum_i \frac{\mu_{\underset{\sim}{A}}(x_i)}{x_i}\right\} \quad (3)$$

Bron: Ross (2010)

Waarbij:

$\underset{\sim}{A}$	Fuzzy set
$\mu_{\underset{\sim}{A}}$	Lidmaatschapsfunctie
$\mu_{\underset{\sim}{A}}(x_1)$	Toebehorengraad

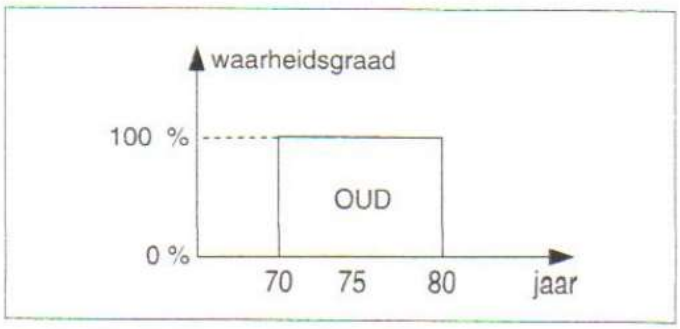
Wat we hier bedoelen met lidmaatschapsfunctie is de waarheidsgraad of toebehorengraad. Het is een getal van 0 tot 1 waarbij als het 0 is dan valt het element helemaal niet onder de wazige verzameling. Als het 1 is, behoort het element helemaal tot de wazige verzameling. Als $\mu_{\underset{\sim}{A}}(x_1)$tussen 0 en 1 is, betekent dat het element deels bij de wazige verzameling behoort. Als je deze verzameling in een grafiek zet krijg je een lijn zoals afgebeeld op figuur 4



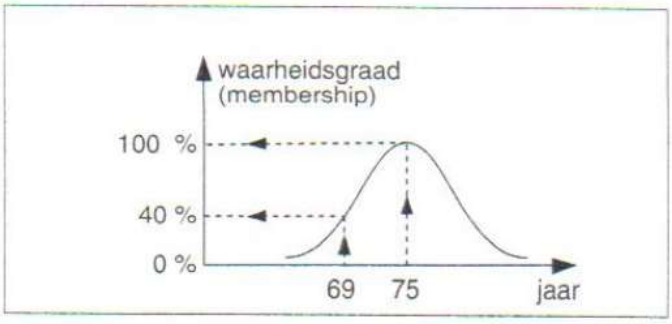
Figuur 4 Fuzzy set $\underset{\sim}{A}$
Bron: Ross (2010)

2.1.1 Voorbeeld

Laten we dit allemaal verduidelijken met een voorbeeld. We hebben aan twee groepen gevraagd vanaf wanneer je oud bent. Groep 1 is een computer die ingesteld is dat vanaf 70 jaar ben je oud en vanaf 80 jaar heel oud. Het resultaat is de grafiek in figuur 5. Groep 2 is een groep mensen die allemaal een verschillende definitie van oud hebben. 1 persoon zegt dat als je 69 jaar oud bent dan ben je “niet zo oud” en bij 75 jaar ben je “zeker” oud. Je krijgt dan de grafiek in figuur 6.



Figuur 5 Membershipfunctie van een normale set
Bron: Maes (z.d.)

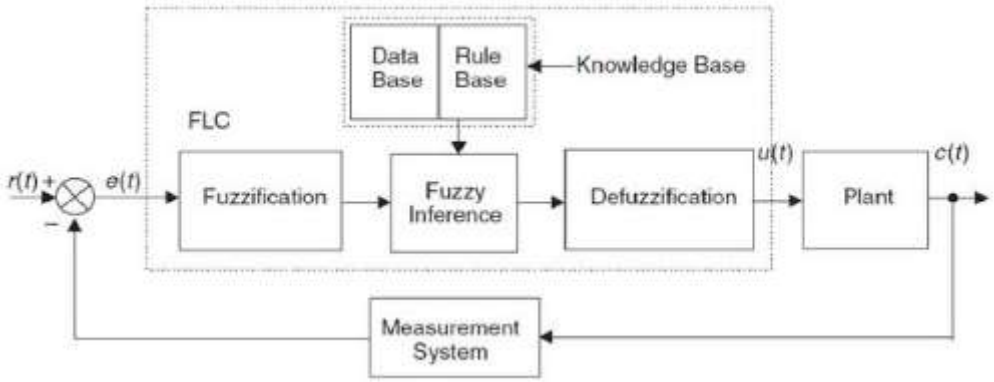


Figuur 6 Membershipfunctie van een fuzzy set
Bron: Maes (z.d.)

In figuur 6 is te zien dat 69 jaar 40% deel uitmaakt van de fuzzy set terwijl 75 jaar helemaal bij de fuzzy set behoort. Dit kan je niet zeggen bij figuur 5 waar het ofwel het er wel deel uitmaakt van de verzameling ofwel niet.
Nu we de basis van de fuzzy logic verstaan kunnen we verdergaan naar de fuzzy regelaars en hoe ze te werk gaan.

2.2 Fuzzy regelaar

De fuzzy regelaar maakt ook gebruik van een feedback systeem. De meest bekende techniek voor fuzzy regelaars is die van Mamdani. Mamdani heeft zijn fuzzy systeem voor het eerst beschreven in *Applications of fuzzy algorithms for control of a simple dynamic plant* (Mamdani, 1974). Het systeem is afgebeeld in figuur 7.



Figuur 7 Blokdiagram fuzzy regelaar
Bron: Khedr, Ammar, & Moustafa Hassan (2013)

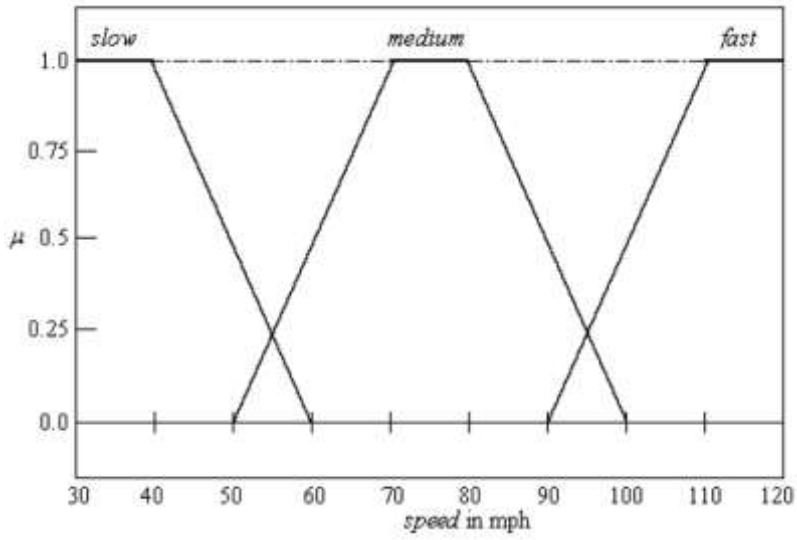
Vergeleken met figuur 2 zien we dat binnen de fuzzy regelaar de berekeningen in serie gebeuren tegenover bij de PID regelaar waar de 3 parameters tegelijkertijd het uitgangssignaal beïnvloeden. De drie belangrijke stappen bij een fuzzy regelaar volgens het Mamdani systeem zijn:

- Fuzzificatie
- Fuzzy inferentie
- Defuzzificatie

2.2.1 Fuzzificatie

De eerste stap bij een fuzzy regelaar. In deze stap geven we alle crisp values[1] zijn eigen toebehorengraad. Als we dit doen voor alle mogelijke getallen krijgen we een lidmaatschapsfunctie. De lidmaatschapsfunctie kan in verschillende vormen komen bijvoorbeeld: een driehoek, een trapezium, een bel curve,…
De meest gebruikte vormen zijn symmetrische en convexe vormen. De vorm hangt af van het systeem waar je gaat regelen(Bhattacharyya & Dutta, 2012). Na de fuzzificatie krijgen we een grafiek zoals in figuur 8.

1 Exacte waarden. Crisp values behoren tot crisp sets waar het een toebehorengraad heeft van alleen 0 of 1



Figuur 8 Lidmaatschapsfunctie
Bron: Bhattacharyya & Dutta (2012)

In figuur 8 heeft elke getal tussen 30 en 120 op de x-as een toebehorengraad gekregen. Bijvoorbeeld 55 mph behoort 25% bij de fuzzy set *slow* en 25% bij de fuzzy set *medium*. De fuzzificatie is nu voltooid.

2.2.2 Fuzzy inferentie

De belangrijkste en moeilijkste stap. De fuzzy inferentie systeem maakt de keuze wat de wazige output zal zijn met de regels. Deze regels kunnen opgesteld worden met booleaanse operatoren zoals OR (figuur 9), AND (figuur 10) en NOT (figuur 11). Ze komen overeen met MAX, MIN en 1-x respectievelijk. De drie fuzzy operatoren hebben elk een wiskundige formule. Stel dat A en B fuzzy sets zijn:

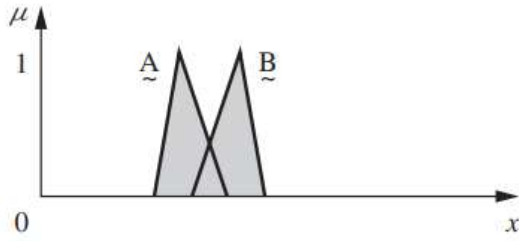
$$(A \cup B)(x) = max[A(x), B(x)] \quad (4)$$

$$(A \cap B)(x) = min[A(x), B(x)] \quad (5)$$

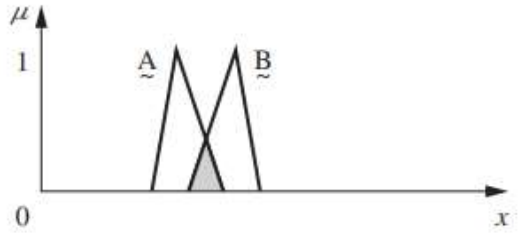
$$\overline{A}(x) = 1 - A(x) \quad (6)$$

Bron:Klir & Yuan (1995)

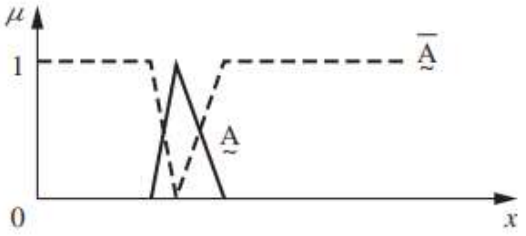
Dus als je een regel opstelt zoals IF *speed is slow* OR *distance is great* THEN accelerate *fast* dan zal de fuzzy inferentie een unie nemen van de fuzzy set *speed: slow* en fuzzy set *distance: great*. Deze regels kan snel complex worden vanaf je meerdere inputs of outputs hebt.



Figuur 9 Unie van fuzzy set A en B
Bron: Ross (2010)



Figuur 10 Doorsnede van fuzzy set A en B
Bron: Ross (2010)



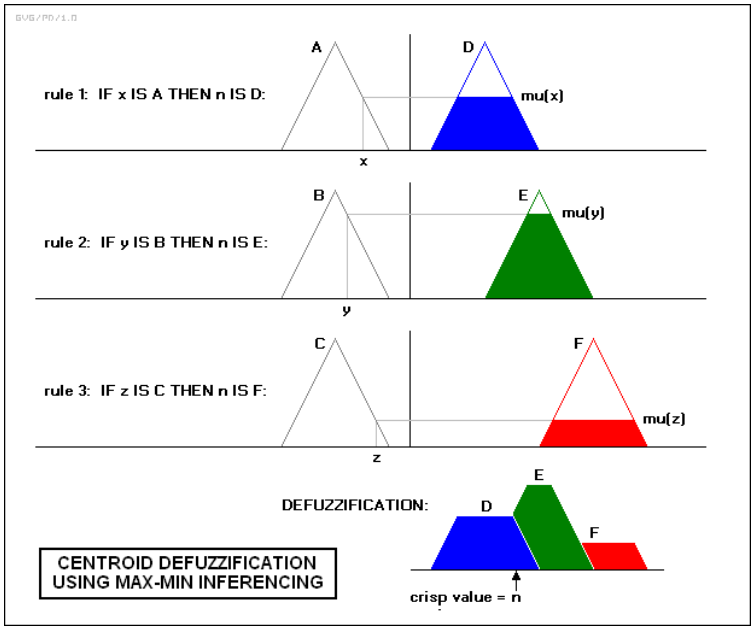
Figuur 11 Complement van fuzzy set A
Bron: Ross (2010)

De fuzzy inferentie kan met de regels en de output lidmaatschapsfunctie een antwoord geven. Maar het probleem is dat het antwoord nog altijd een wazig getal is. We hebben een exact antwoord nodig dus schiet er nog één stap over.

2.2.3 Defuzzificatie

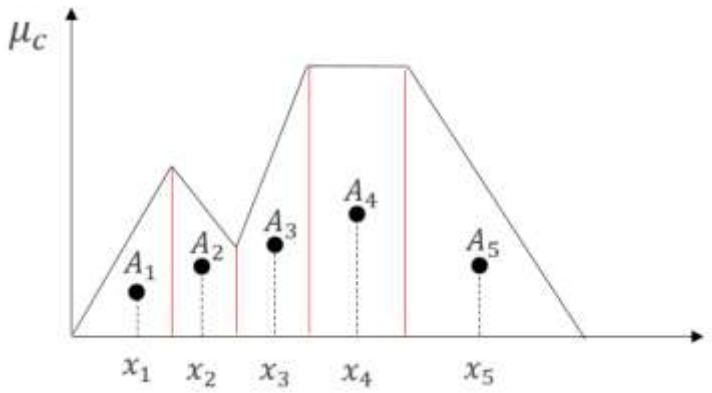
Stel ons antwoord van de fuzzy inferentie is dit: 24% temperatuur verlagen, 15% temperatuur verhogen en 61% temperatuur verhogen. Het logische antwoord zou zijn

temperatuur verhogen omdat het meer waar is dan verlagen. Dit betekent wel dat we de 24% gewoon negeren alsof die er nooit was. Dit geeft ons een onnauwkeurig antwoord. Een van de methodes voor defuzzificatie is het zwaartepunt nemen van de gecombineerde figuur (center of area). Als de fuzzy inferentie voltooid is krijgen we één geheel vorm, afgebeeld in figuur 12.



Figuur 12 Zwaartepunt defuzzificatie
Bron: Wikimedia Commons (2003)

De fuzzy regelaar gaat dan de gehele vorm in n aantal segmenten delen zoals in figuur 13 afgebeeld.



Figuur 13 Verdeelde lidmaatschapsfuncite
Bron: Codecrucks (2021)

De formule om de center of area te berekenen is:

$$x^* = \frac{\sum_{i=1}^{n} A_i * x_i}{\sum_{i=1}^{n} A_i} \quad (7)$$

Bron: Novák & Perfilieva (1999)

Waarbij:

A_i De oppervlakte van het segment

x_i De center van het segment

n Aantal segmenten

Als we de formule hebben gebruikt krijgen we het zwaartepunt van de gecombineerde lidmaatschapsfuncties. Dit is het antwoord op de regels en de originele input. Het antwoord wordt gestuurd naar het proces en er zal terug een meting gebeuren. De gemeten waarde wordt teruggestuurd naar de input van de controller en de procedure begint opnieuw.

2.3 Toepassingen van fuzzy logic

Fuzzy logic is minder bekend dan PID maar het is nog steeds veel gebruikt. Het wordt gebruikt bijvoorbeeld in stofzuigers, gezichtsherkenning software, air conditioners, remsystemen, weervoorspellingen,… (Singh e.a., 2013). Door de stijgende energieprijzen zijn er zelfs energiebesparing modellen gemaakt met fuzzy logic regelaars (Parvin, Al-Shetwi, Hannan, & Ker, 2021). Cybersecurity gebruikt ook fuzzy logic om het risico van data verlies bijvoorbeeld te verminderen (Bhattacharyya & Dutta, 2012). Fuzzy logic is hiermee een zeer waardevol deel van het geheel Automatisatie.

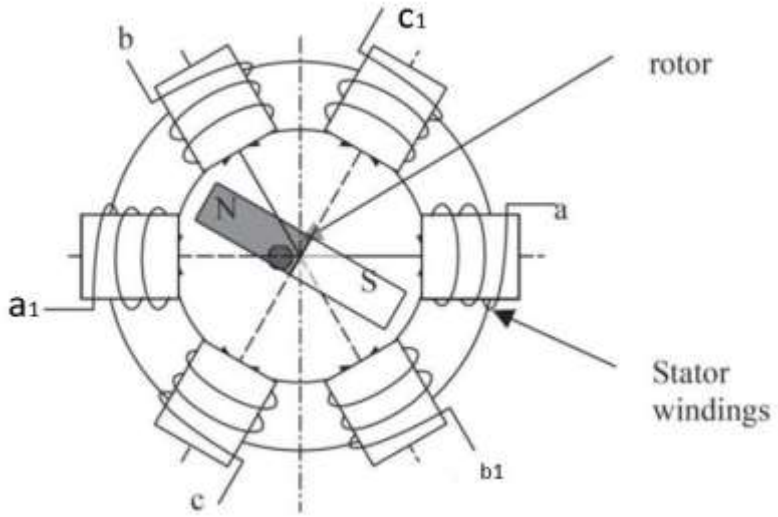
3 BLDC motor

BLDC motor staat voor *Brushless Direct Current* motor of een motor zonder bortels. In dit hoofdstuk gaan we ons verdiepen in de werking van deze motor.

3.1 Stator en rotor

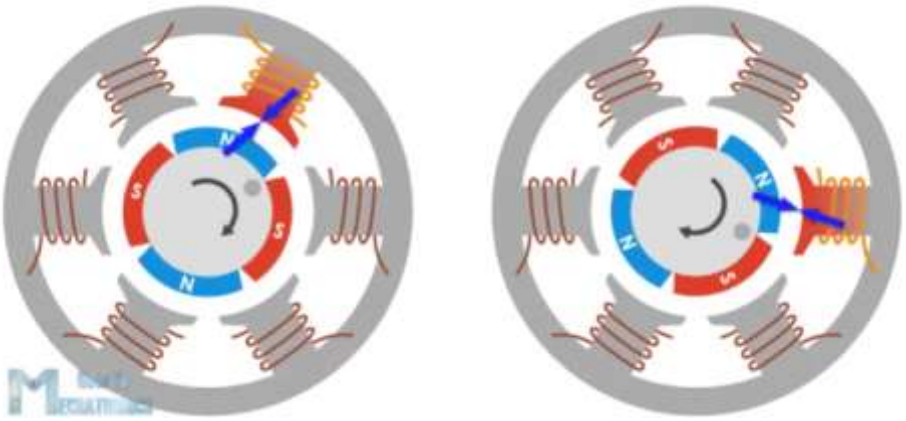
Als er door een draad die gewikkeld is rond een ijzeren kern stroom vloeit dan genereert die een magnetisch veld (Hambley, 2018). Dit veld kan gebruikt worden in combinatie met een ander permanente magneet, die altijd een magnetisch veld heeft, om een draai beweging te krijgen. Magneten trekken elkaar bij tegenovergestelde polen en stoten elkaar af bij de zelfde polen. De BLDC motor maakt gebruik van magnetisme om een koppel te krijgen. Door

het koppel resulteert in een roterende beweging. In een motor is het deel dat stationair blijft de stator en het deel dat beweegt de rotor.



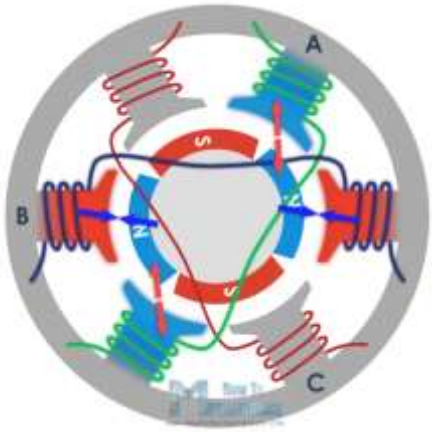
Figuur 14 BLDC motor schema
Bron: bewerkt van Halvaei & Shahrbabak (2014)

In figuur 14 is een BLDC motor te zien met twee polen. Dit kunnen er soms meerdere zijn. Dus om een beweging te krijgen geven we stroom aan b en krijgen we een magnetisch veld die het noorden zal aantrekken. Dan zal de rotor met zijn noordelijke pool naar b bewegen. Net als de rotor bij b komt stoppen we b met stroom te geven en geven we c1 stroom. De beweging gaat nu voort. Als we dit niet deden zou de rotor stoppen met draaien omdat het noordelijke magnetisch veld van de rotor en het zuidelijke magnetisch veld van b elkaar blijven aantrekken.
We kunnen een rotor hebben met meerdere polen. Dan zal de draaibeweging vloeiender worden.



Figuur 15 BLDC motor met een vierpolige rotor
Bron: Dejan (2019a)

Om de efficiëntie nog meer te verbeteren kunnen we de tegenovergestelde pool ook laten bekrachtigen door ze met elkaar te verbinden. Wat er nog kan worden toegepast is dat we twee winding paren tegelijkertijd bekrachtigen, maar niet in dezelfde richting. Dus zodat één winding de rotorpool aantrekt en de andere winding dezelfde rotorpool wegduwt. Deze samenstelling is te zien in figuur 16.



Figuur 16 Efficiënte BLDC motor
Bron: Dejan (2019a)

Op het eerste zicht lijkt dit een interessante motor zonder problemen. Maar er zijn wel twee vragen die we ons moeten stellen. Hoe weet de motor wanneer de juiste wikkeling stroom moet krijgen en hoe weet de motor de positie van de rotor bij het opstarten? Op beide vragen is er een antwoord en het heet de ESC.

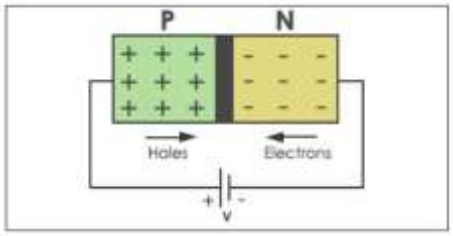
3.2 ESC

De Electrical Speed Control zal het toerental van de motor bepalen. Dit gebeurt met behulp van zes MOSFET's en PWM.

Figuur 17 ESC
Bron: Eigen werk

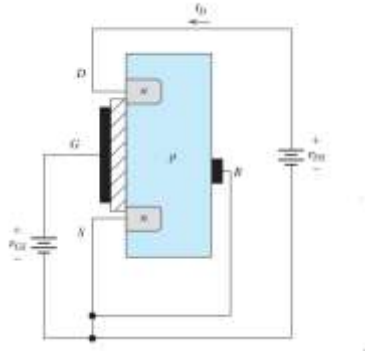
3.2.1 MOSFET

MOSFET staat voor *metal-oxide-semiconductor field-effect transistor*. Het is een semiconductor of halfgeleider. Een halfgeleider is een stof die beter geleidt dan een isolator maar minder goed geleidt dan een geleider. Silicium is meestal gebruikt als een halfgeleider. Door het silicium te vullen met onzuiverheden krijgen we twee soorten halfgeleiders: Het *p-type* en *n-type*. Bij het n-type heb je een grotere concentratie van elektronen. Bij het p-type heb je een grotere concentratie van positief geladen deeltjes of "gaten" (Hambley, 2018). In het midden heb je een grenslaag of junctie. De stroomzin van elektronen is van min naar plus. Dus als we een positieve spanning op het p-gedeelte zetten dan gaan de elektronen van het n-gedeelte de gaten opvullen van het p-gedeelte. Als dit gebeurt en de spanning is groter om de scheiding te overwinnen zal de stroom door de hele halfgeleider geleiden. Maar als we positieve spanning zetten op het n-gedeelte zal de barrière breder worden en kunnen er geen elektronen overgebracht worden dus heb je geen geleiding. Zo werkt een diode en dit is te zien in figuur 18.



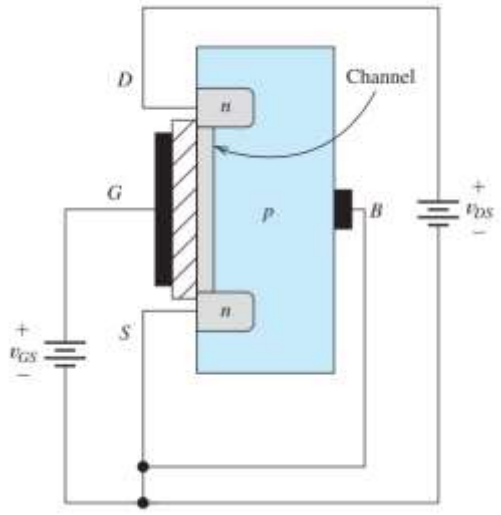
Figuur 18 Werkingsschema Diode
Bron: Nasir (2018)

Een MOSFET gebruikt hetzelfde werkprincipe als een diode, maar met het feit dat we drie geladen zones hebben in plaats van twee. We krijgen dan een npn en pnp configuratie. Hiermee vergroten de mogelijkheden. We kunnen een MOSFET gebruiken als een schakelaar bijvoorbeeld. In figuur 19 is een n-channel MOSFET in een schema afgebeeld.



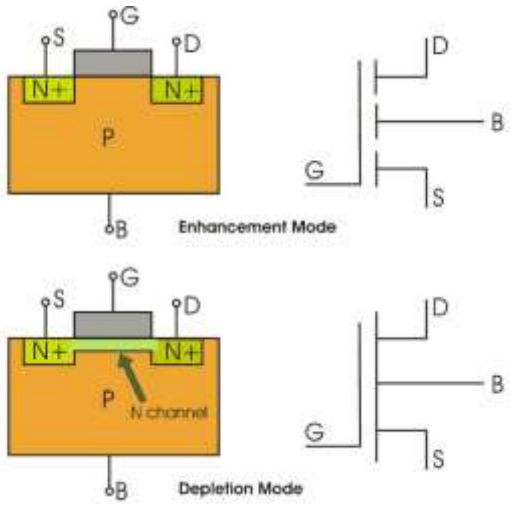
Figuur 19 N-channel MOSFET schema
Bron: Hambley (2018)

De verbindingen van de MOSFET zijn de *drain* (D), de *source* (S), de *gate* (G) en de *body* (B). De body en de source worden samen verbonden om het aantal verbindingen te verlagen naar 3. De gate is gescheiden van p-type materiaal door een laag siliciumdioxide. We willen dat er elektronen vloeien van de source naar de drain. We weten als we een positieve spanning sturen naar een n-deel dat de scheiding zal vergroten en er geen geleiding plaatsvindt. Om dit op te lossen moeten we een positieve spanning sturen naar de gate. We weten dat in het p-type materiaal een grotere concentratie van gaten zijn, maar er zijn nog altijd elektronen aanwezig. Als we een positieve spanning sturen naar de gate trekken we alle elektronen in het p-type materiaal naar het siliciumdioxide laagje. Het gevolg is dat er een hoge concentratie van elektronen onder het laagje gevormd is. Deze concentratie transformeert in een n-type kanaal die zich bevindt tussen de source en de drain. Dit is goed afgebeeld in figuur 20.



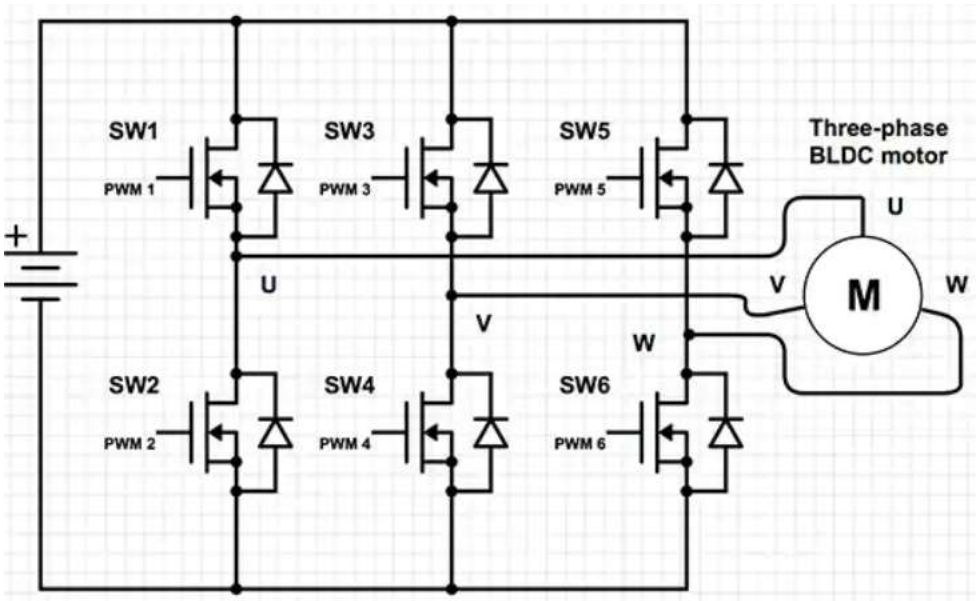
Figuur 20 N-channel MOSFET schema met kanaal
Bron: Hambley (2018)

Als we V_{ds} bekrachtigen zal er stroom geleiden van de drain, door het kanaal en naar de source. Deze n-channel MOSFET is uiteindelijk dus een normaal open schakelaar in.
Er zijn verschillende MOSFET's. Zo hebben we de p-channel MOSFET waar de werking hetzelfde is alleen zijn het p- en n-type materiaal verwisseld. We hebben ook voor elke MOSFET twee modes, verrijking (enhancement) en verarming (depletion). In figuur 21 zijn ze beiden afgebeeld voor een n-channel MOSFET.



Figuur 21 Verrijkende en verarmende n-channel MOSFET
Bron: electronicsforu (2019)

De MOSFET in figuur 20 is een verrijkende MOSFET. Het verschil tussen de twee is dat je een positieve spanning stuurt naar de verrijkende om een geleiding te krijgen tussen de source en de drain en bij een verarmende heb je al een kanaal tussen source en drain. Als je dat wil stoppen moet je een negatieve spanning sturen naar de gate om het kanaal te verbreken en met gevolg de geleiding stoppen tussen source en drain. Met andere woorden een verarmende MOSFET is een normaal gesloten elektronische schakelaar tegenover een verrijkende MOSFET die een normaal open schakelaar is. De mode die je gebruikt hangt af van je doel. In de ESC gebruiken we n-channel verrijkende MOSFET's.
In de ESC gebruiken we zes MOSFET's in een brug structuur, afgebeeld in figuur 22.

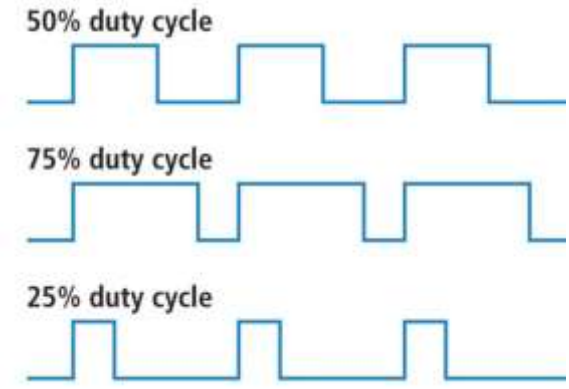


Figuur 22 ESC schema
Bron: Digi-Key (2016)

Om de motor te laten draaien moeten de MOSFET's in een volgorde bekrachtigd worden. Tegelijkertijd moet er ook een MOSFET bekrachtigd worden zodat de stroom naar de massa kan gaan. Dit betekent dat er op elk moment twee MOSFET's bekrachtigd worden. Als we een signaal geven naar SW1 zal er stroom geleiden naar spoel U. Van spoel U gaat de stroom via de gemeenschappelijke neutraal naar spoel W bijvoorbeeld omdat SW6 ook bekrachtigd is, wat een geleiding geeft naar de massa. Bij elke MOSFET in figuur 22 zit een diode over body en drain. Dit is de body diode.
Wanneer we met een inductieve last werken wordt de last bekrachtigd. Als we geen stroom meer sturen naar de inductieve last moet de energie ergens naartoe (Wilcher, 2012). Deze stroom kan te hoog zijn voor onze elektronische componenten. We willen ook niet dat de MOSFET deze stroom in omgekeerde richting stuurt want dat kan onze MOSFET beschadigen (etechnog, 2019). Met een body diode zal de terugstroom geleiden via de diode en niet de MOSFET.
De bekrachtiging van de MOSFET's gebeurt door een microcontroller die een PWM signaal genereert.

3.2.2 PWM

Pulsbreedtemodulatie (Pulse-width modulation) is een manier om analoge circuits te regelen met een digitale output (Ibrahim, 2014). Een PWM signaal is rechthoekig in vorm. De belangrijkste parameter van een PWM signaal is de breedte. Dit heet de arbeidscyclus (duty cycle) en die bepaald hoe lang je toestel aan of uit is. Een arbeidscyclus van 100% betekent dat het toestel altijd aan is terwijl één met 50% zal maar werken voor de helft van de tijd. Dit is goed te zien op figuur 23.

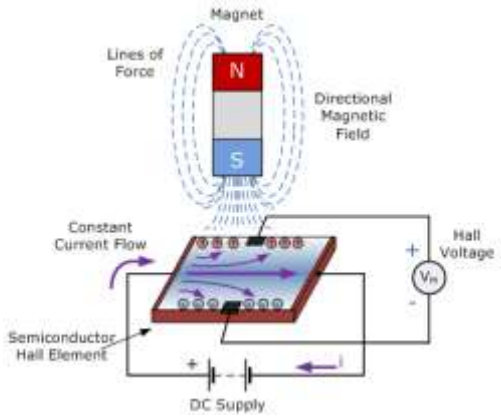


Figuur 23 PWM arbeidscyclus
Bron: sparkfun (z.d.)

Een microcontroller zoals de Arduino kan een PWM signaal genereren. In de ESC zal de breedte van het PWM signaal de snelheid van het aan en uitschakelen van de MOSFET bepalen. Hoe breder het signaal, hoe sneller de MOSFET's schakelen. Als ze rapper schakelen zal de rotor sneller het opgewekte magnetisch veld van de stator willen volgen omdat die sneller opgewekt en uitgeschakeld wordt. Zo kunnen we de snelheid van de BLDC motor regelen. Maar hoe weten we op welke positie we zitten als de motor uitgeschakeld is?
Er zijn BLDC motoren met sensoren die de positie van de permanente magneten kunnen detecteren. Deze sensoren heten Hall effect sensoren.

3.2.3 Hall effect sensor

Dit zijn sensoren die op basis van het Hall effect werken. Het Hall effect is de opgewekte spanning die wordt gecreëerd wanneer een magnetisch veld een metaal nadert die stroom geleid (Ramsden, 2006). De spanning die hierdoor ontstaat, noemt de Hall spanning. Om de Hall spanning te meten moet die loodrecht staan tegenover de draden die stroom geleiden zoals afgebeeld op figuur 24. Als we het magnetisch veld omdraaien zal de Hall spanning van polariteit veranderen.



Figuur 24 Hall effect
Bron: Collins (2021)

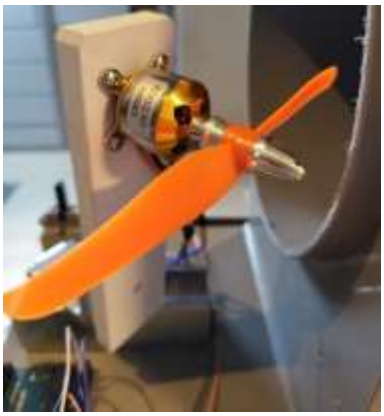
We gaan deze spanning gebruiken om de positie te bepalen van de windingen. In de BLDC motor zit een Hall sensor voor elke spoelwinding. Bij het draaien van de motor passeren de magnetische velden langs de sensoren. De sensor meet een positieve of negatieve spanning (Yedamale, 2003). Met alle sensoren weet de ESC de positie van de motor. De motor die we in onze scriptie gebruiken heeft geen Hall sensoren. Er wordt gebruikt gemaakt van tegen-elektromagnetische krachten.

3.2.4 Tegen-elektromagnetische kracht

We weten dat wanneer er stroom vloeit door een draad die gewikkeld is rond bijvoorbeeld ijzer dat die een magnetisch veld opwekt (Hambley, 2018). Dit geldt ook omgekeerd. Als je een magnetisch veld introduceert bij een draad die gewikkeld is rond een ijzeren kern zal die een spanning genereren. We weten dat bij een BLDC er altijd 2 spoelen bekrachtigd worden en één niet. Omdat de niet-bekrachtigde spoel in een magnetisch veld zit zal die een spanning opwekken. De controller van de ESC ziet dat die een spanning krijgt van een spoel en weet zo op welke positie de motor zit (Gamazo-Real, Vázquez-Sánchez, & Gómez-Gil, 2010).

3.3 Bevestiging van de BLDC motor

De motor die wij gebruiken is de A2212/6T 2200KV
Op de as van de motor komt een rotor met twee vleugels. Die moet in de buis passen dus moet de binnendiameter van de buis groter zijn dan de rotor diameter zodat de vleugels de buis niet raken. Door het draaien van de vleugels genereren wij een luchtstroom.



Figuur 25 Bevestiging van de motor
Bron: Eigen werk

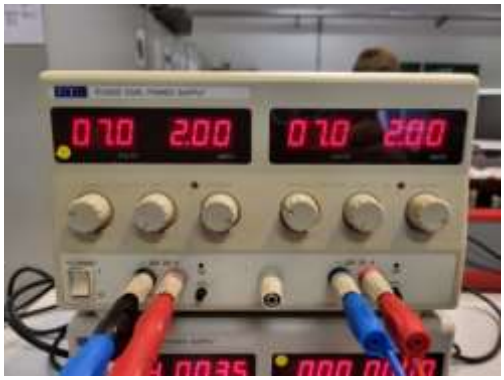
De rotorvleugels zijn 15,2 cm in diameter en de binnendiameter van de tunnel is 17,25 cm. Wanneer we de bladen handmatig laten draaien raken ze de tunnel niet. De motor is bevestigd op een witte plaat gemaakt van kunststof met vier schroeven. De witte plaat zit vast op een ijzeren hoek met twee inbusbouten. De ijzeren hoek zit vast aan een DIN rail met een bout die te zien is in figuur 26. We kunnen de bout lossen om de motor dichter of verder van de tunnel te plaatsen.

Figuur 26 Bevestiging van de motor op DIN rail
Bron: Eigen werk

3.4 Voeding

Voor de voeding van de motor gebruiken we een regelbare DC voeding die tot 30V en 10A kan leveren. Een voeding met regeling laat ons toe om te kijken wat de beste spanning en stroom is en wat de gevolgen op de motor zijn als je de spanning en de stroom verandert. De positieve- en de negatieve kabel van de voeding zitten vast in de klemmen. Deze zijn vastgemaakt op een DIN rail. De ESC is dan verbonden met deze klemmen, afgebeeld op figuur 27.
Op school is er een voeding van 24V en 2A. De motor met deze voeding starten bleek een probleem want hij startte niet. 2A was dus niet genoeg om de motor te laten draaien dus zijn er twee voedingen gepakt en werden die parallel gezet zodat je 4A kreeg, afgebeeld op figuur 28. Dit heeft de motor geholpen om te kunnen draaien. Bij het opstarten van de motor maakt hij geluid en geeft de motor een paar rukbewegingen. Dit doet de motor om zijn positie van de polen te bepalen en om te weten welke spoel hij moet bekrachtigen.

Figuur 27 Voeding van de motor
Bron: Eigen werk



Figuur 28 Schoolvoeding
Bron: Eigen werk

Vervolgens was het nodig om te bekijken bij welke spanning de motor zou draaien. Hoe hoger je de spanning zette, hoe sneller de motor draaide. Als je de spanning onder 7V zet valt de motor direct stil. Op tabel 3 is er een datasheet van het type motor die wij hebben te zien met daarop 11,1V als constante. Uiteindelijk is dan 10V als spanning besloten om te gebruiken voor de motor. De reden hiervoor is omdat hogere spanning te motor te snel laten draaien, terwijl de sensorspanning zijn maximum behaald heeft.

Tabel 3 Datasheet A2212 BLDC motor
Bron: abra-electronics (z.d.)

MODEL	KV (rpm/V)	Voltage (V)	Prop	Load Current(A)	Power (W)	Pull (g)	Efficiency (g/W)	Lipo Cell	Weight (g)
A2212	930	11.1	1060	9.8	109	660	6.1	2-4S	52
	1000		1047	15.6	173	885	5.1		
	1400		9050	19.0	210	910	4.3		
	1800		8060	20.8	231	805	3.5		
	2200		6030	21.5	239	732	3.1	2-3S	
	2450		6030	25.2	280	815	2.9		

Er was nog steeds een probleem namelijk de motor stopte met draaien na 1 of 2 minuten. En op de schoolvoeding kan je niet hoger dan 4A met twee voedingen dus is er besloten om een

nieuwe voeding aan te schaffen met meer stroom. Uiteindelijk is er gekozen voor om op 5A en 10V te werken.

3.5 Kv verhouding

Op de motor staat er 2200Kv maar wat betekent dat? Kv is een verhouding en wilt zeggen dat voor 1V de motor 2200rpm of toeren per minuut kan draaien. Dus als ons motor op 10V draait en we geven de ESC een arbeidscyclus van 100% dan zal die 22000 toeren per minuut draaien. Op de motor staat ook A2212/6T. De 22 staat voor de motor diameter (22mm), de 12 staat voor de hoogte van het motoras (12mm) en 6T zegt hoeveel keer de draad is gewikkeld rond een pool (Minipro, 2018).

3.6 Toepassingen van BLDC motor

De BLDC motor wordt bij huishoudelijke toestellen vaker en vaker gebruikt. Bijvoorbeeld in de compressor van je koelkast, airconditioning, DVD/Cd-spelers, koeling ventilatoren in je computer,…(Xia, 2012) Ze zijn bestendig tegen lage- en hoge temperaturen, vibraties en shock. BLDC motoren produceren ook weinig geluid wat ze perfect maakt voor een koeling ventilator in de computer. Deze motor is gekozen voor ons proef omdat het klein en compact was, hoge toerental kan leveren en het kan bestuurd worden met een Arduino microcontroller met PWM.

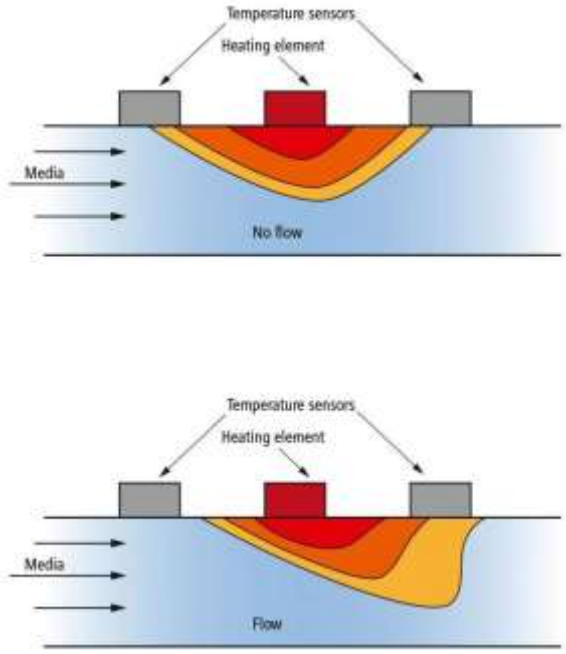
4 Sensor

De sensor die wordt gebruikt is de F2 flow sensor van SEIKOM. Vergeleken met andere anemometers die groot zijn en bewegende delen hebben is de F2 flow sensor klein, compact en stationair. In bijlage A staat genoteerd dat de sensor meet volgens het calorimetrisch meetprincipe meet. Als we willen weten hoe de F2 sensor meet moeten we weten hoe het calorimetrisch meetprincipe werkt.

Figuur 29 F2 flow sensor
Bron: SEIKOM (z.d. -a)

4.1 Calorimetrisch meetprincipe

Het calorimetrisch meetprincipe is gebaseerd op calorimetrie, het meten van warmtecapaciteit met behulp van calorimeters. Er zijn veel verschillende calorimeters maar wij gaan ons richten op de calorimeter voor luchtstroom.
De sensor heeft een thermisch element en wordt warmer. Het zal warmte afgeven aan de omgeving. Doordat er lucht zal stromen in de tunnel gaat het thermisch element afkoelen. Er is namelijk een relatie van luchtsnelheid en de hoeveelheid afgekoelde warmte. Hoe hoger de luchtstroom, hoe meer het thermisch element zal afkoelen. De sensor meet dit temperatuur verschil en zet het om in een spanning. Dit meetprincipe is afgebeeld in figuur 30.



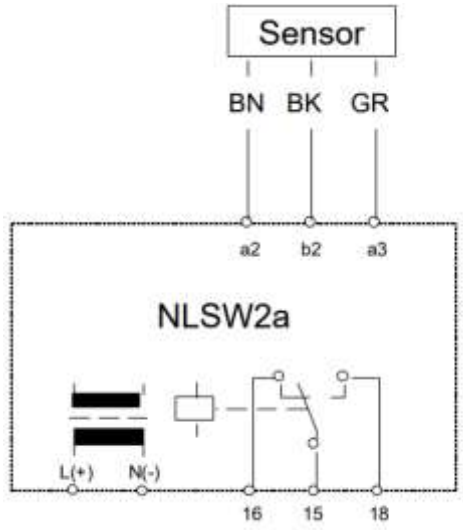
Figuur 30 Calorimetrisch meetprincipe voor luchtstroom
Bron: Baumer (z.d.)

4.2 Luchtstroom monitor

Als we het analoge signaal willen meten hebben we een luchtstroom monitor nodig. De functie van deze monitor is de sensor stroom te geven en de analoge spanning meten die de sensor krijgt door het calorimetrisch meetprincipe. Er werd gekozen voor de NLSW2A van SEIKOM die voor de F2 flow sensor past.

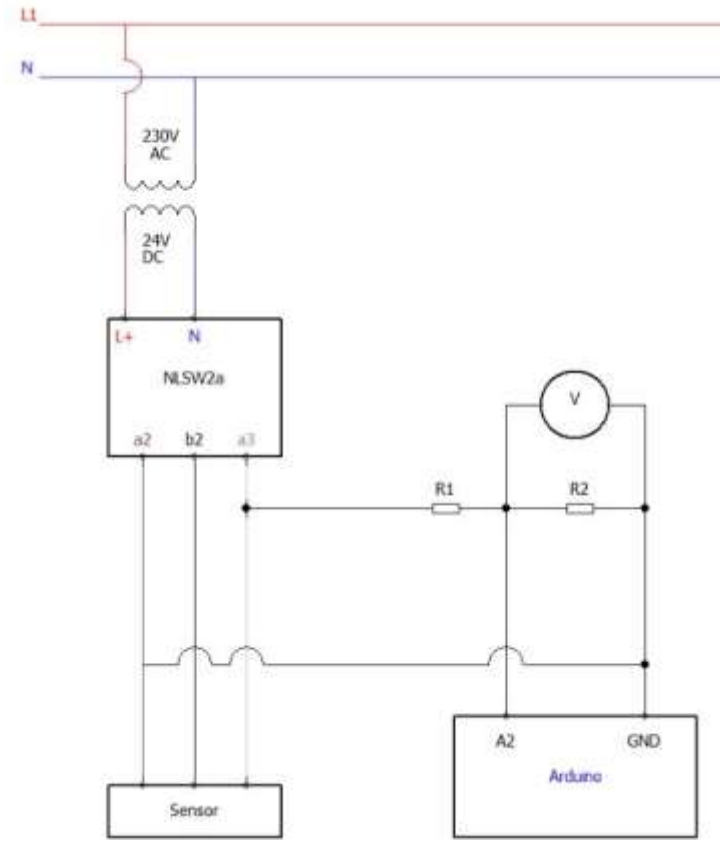
Figuur 31 NLSW2a
Bron: SEIKOM (z.d. -b)

Jammer genoeg is uiteindelijk gebleken dat deze luchtstroom monitor geen analoge signalen kan sturen. Het enige wat dit model kan doen is schakelen wanneer dat het wind detecteert zoals te zien op figuur 32. Dit is natuurlijk niet nuttig voor onze proef dus is er een manier gevonden om toch een analoog signaal te krijgen en die naar de Arduino te sturen.



Figuur 32 Elektrische schema NLSW2a
Bron: SEIKOM (z.d. -b)

De sensor heeft drie kabels. Als we de spanning over twee kabels meten terwijl er wind stroomt, zou er meer spanning moeten zijn dan als er geen wind is. Bij het meten tussen alle kabels blijkt dat tussen de grijze en bruine kabel een spanning is die vergroot als de sensor meer luchtstroom meet. We kunnen deze spanning als analoog signaal gebruiken voor de Arduino om onze motor te regelen. Op figuur 33 is het elektrische schema te zien van de sensor, monitor en Arduino.



Figuur 33 Elektrische schema sensor en monitor
Bron: Eigen werk

De Arduino is in dit schema simplistisch voorgesteld. Alleen de relevante verbindingen zijn hier getekend. De NLSW2a heeft 24V DC voeding nodig. We gebruiken een transformator die 230V AC naar 24V DC omvormt. Deze transformator is afgebeeld op figuur 34. R1 en R2 zijn weerstanden van 5,5kΩ en samen vormen ze een spanningsdeler. De spanningsdeler en het nut ervan wordt uitgelegd bij de Arduino. A2 is de analoge input van de Arduino.

Figuur 34 230VAC/24VDC transformator
Bron: Eigen werk

Nu we kunnen meten is het mogelijk om de windsnelheid te weten en de spanning die zal stijgen bij verhoogde luchtstroom. Een multimeter is geplaatst over R1 en R2 om de spanning van de sensor te meten.

4.3 Metingen van sensor

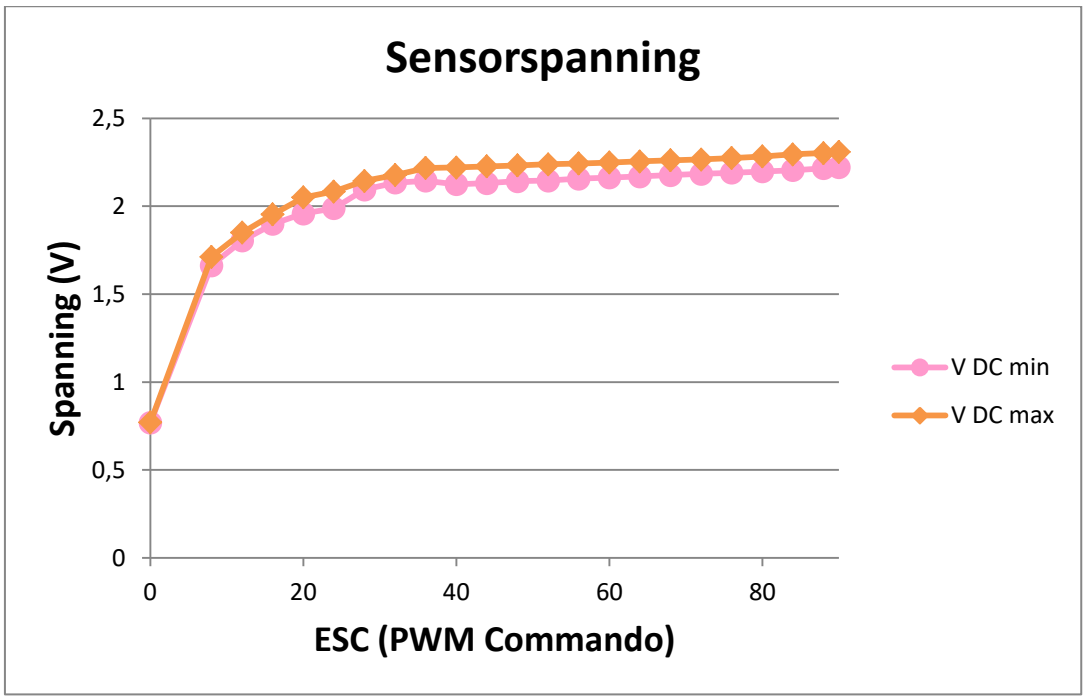
De sensor stuurt een spanning naar de Arduino, maar die weet niet hoeveel de windsnelheid is bij een bepaalde sensorspanning. We hebben dus een digitale anemometer nodig.

4.3.1 Digitale anemometer

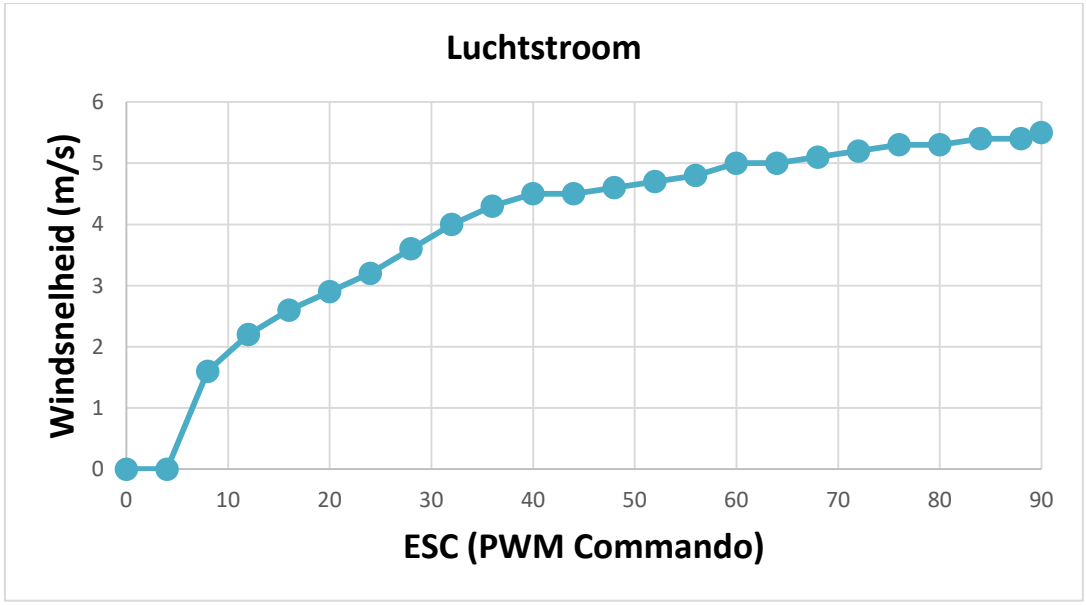
We gaan een digitale anemometer gebruiken om te weten welke windsnelheid we hebben bij welke PWM signaal en zodat we een trend krijgen van hoe de windsnelheid evolueert. De anemometer wordt tegen de sensor geplaatst.

Figuur 35 Digitale anemometer
Bron: Eigen werk

Figuur 36 en figuur 37 zijn de resultaten van de metingen. De x-as is het signaal dat naar de ESC wordt gestuurd. In het Arduino programma gebruiken we de Servo library om de ESC te besturen. De ESC maakt gebruikt van puls breedtes tussen 1 milliseconde en 2 milliseconden. Maar de Servo library verstaat alleen getallen tussen 0 en 180 dus moeten we die omzetten. Als we de motor testen blijkt dat zijn maximumsnelheid behaald wordt bij een commando van 90. Dus gaan we maar tot 90 meten.
De sensorspanning oscilleert veel. Daarom is er besloten om de minimum sensorspanning en de maximum sensorspanning apart te meten. We krijgen dan een zone tussen de twee lijnen waar de sensorspanning altijd zal bevinden.



Figuur 36 Sensorspanning grafiek
Bron: Eigen werk



Figuur 37 Luchtstroom grafiek
Bron: Eigen werk

Als we figuur 36 en figuur 37 bekijken zien we eerst dat er een zone is waar er geen windsnelheid is maar de sensor meet nog steeds een kleine spanning. In deze situatie draait de motor niet. Zodra de ESC een signaal krijgt van 8 zal de motor starten. We zien dat zodra de motor draait er een sterke stijging is. Daarna blijven de waarden stijgen naarmate de motor sneller draait. Maar vanaf de ESC een commando krijgt van 40 beginnen de waarden minder snel te stijgen. Dit is zeer goed te zien bij de sensorspanning. De lijn begint af te vlakken. Er is dus weinig effect voor de regelaar als we de ESC een hoger commando geven

dan 40 omdat de sensorspanning te traag stijgt. Deze sensorspanning is de meting en feedback voor onze regelaar. Daarom is er besloten om de ESC te laten werken op commando's van 0 tot 40.

4.4 Soorten stromingen

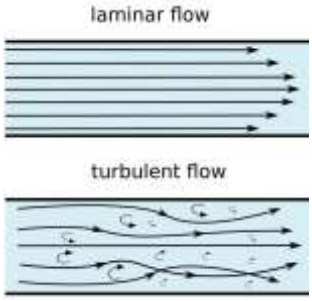
Als we bezig zijn met het meten van een gas of vloeistof in een buis is het belangrijk om de twee typen stromingen te bekijken want dit kan invloed hebben op onze metingen.

Laminaire stroming

Bij een laminaire stroming gebeurt de stroming glad en recht. De stroming is verdeeld in lagen en wordt ook zo verplaatst (Crowe & Crowe, 2009). Dit gebeurt meestal bij vloeistoffen met lage viscositeit en in buizen met kleine diameters.

Turbulente stroming

Bij een turbulente stroming gebeurt de stroming chaotisch, onstabiel en niet in lagen. In deze stroming kunnen er gas- of vloeistofdeeltjes geneigd zijn om in de tegenovergestelde richting te stromen. Als we dit in onze tunnel hebben, zal de sensor meting onstabiel zijn doordat de afkoeling van het thermisch element niet optimaal gebeurt.



Figuur 38 Soorten stromingen
Bron: CFD support (z.d.)

4.4.1 Reynoldsgetal

We kunnen het type van de stroming berekenen met het Reynoldsgetal. Het is een dimensieloos getal dat zegt wanneer een stroming laminair of turbulent is.

Laminair: $Re < 2300$

Turbulent: $2300 < Re < 4000$

Transitie: $Re > 4000$

$$Re = \frac{\rho * v * d}{\mu} \quad (8)$$

Waarbij:

ρ densiteit (kg/m^3)

v windsnelheid (m/s)

d binnendiameter buis (m)

μ dynamische viscositeit (Pa*s)

De binnendiameter van de buis is 0,1725m

De densiteit voor 21 graden lucht is 1,2 kg/m^3

De dynamische viscositeit voor 21 graden lucht is 18,17*10^{-6} Pa*s (Engineering Toolbox, z.d.)

De snelheid is variabel. We kunnen nu het Reynoldsgetal berekenen voor onze gemeten snelheden die we gaan gebruiken.

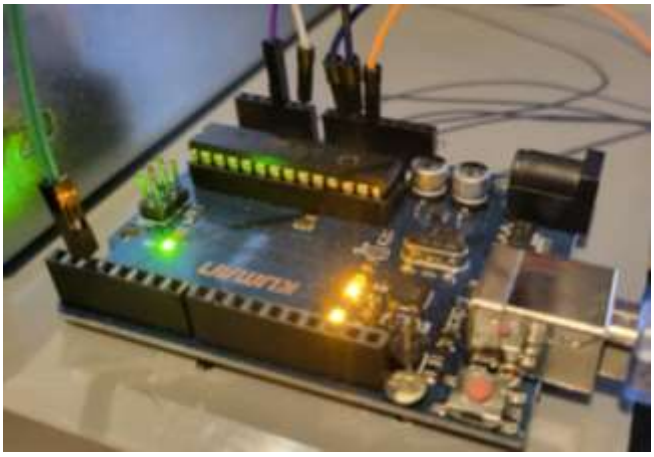
Tabel 4 Reynoldsgetal van gemeten windsnelheden
Bron: Eigen werk

ESC commando	Windsnelheid	Reynoldsgetal
0	0	0
4	0	0
8	1,6	18289
12	2,2	25147
16	2,6	29719
20	2,9	33148
24	3,2	36577
28	3,6	41149
32	4	45722
36	4,3	49151
40	4,5	51437

In tabel 4 is duidelijk te zien dat de stroming turbulent verloopt in de buis. Zelfs bij de allerlaagste snelheid die de motor levert bij 10V is het nog steeds turbulent. Als we het Reynoldsgetal willen verlagen dan moeten we de windsnelheid verlagen of de buis diameter verkleinen. Voor onze scriptie moeten we weten dat dit invloed heeft op onze metingen. De turbulente stroming kan het oscillerende sensor signaal nog meer verslechteren wat als gevolg heeft dat de regelaars meer werk hebben om de sensorspanning op de setpoint te krijgen.

5 Arduino

Arduino is een open source platform voor hardware en software (Arduino, 2017). Er zijn verschillende Arduino borden. De meest bekende is de Arduino Uno die we gebruiken in deze scriptie zoals te zien op figuur 39. De Arduino kan ook een PWM signaal genereren die we gaan gebruiken voor het besturen van onze motor.



Figuur 39 Arduino Uno
Bron: Eigen werk

We hebben voor Arduino gekozen omdat het goedkoop is en gemakkelijk mee te werken is. Het is open source wat betekent dat je veel informatie vindt over deze microcontroller. De forums van Arduino bevatten veel voorbeelden van programma's die nuttig kunnen zijn.
De Arduino kan analoge signalen meten en die naar digitale signalen omzetten. De Arduino zet het analoge signaal van 0 tot 5V om in een 10-bit getal van 0 tot 1023. Dit betekent dat Arduino geen analoge signalen kan meten die hoger zijn dan 5V. Een te hoge spanning kan de Arduino beschadigen. In werkelijkheid zijn alle sensorspanning waarden in figuur 36 gedeeld door twee met een spanningsdeler.

5.1 Spanningsdeler

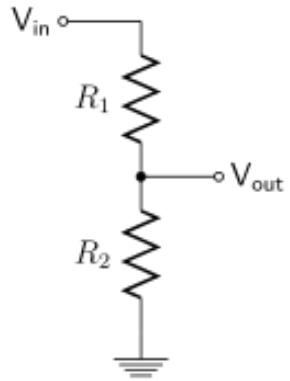
De reden waarom we een spanningsdeler gebruiken is omdat de sensor oscilleert. Als de sensor oscilleert bij de maximum motorsnelheid dan zijn er metingen boven de 5V. Om potentiële beschadigingen te voorkomen maken we gebruik van een spanningsdeler om onze sensorspanning door twee te delen. Op figuur 40 is de spanningsdeler te zien. De formule om de spanning te delen is:

$$V_{out} = \frac{R2}{R1+R2} * V_{in} \quad (9)$$

Als we twee keer dezelfde weerstand gebruiken dan krijgen we

$$V_{out} = \frac{1}{2} * V_{in} \quad (10)$$

Met V_{out} de spanning naar de Arduino en V_{in} de spanning afkomstig van de sensor.



Figuur 40 Spaningsdeler
Bron: Krishnavedala (2014)

Dankzij de spanningsdeler kan er nooit een analoog signaal hoger dan 5V naar de Arduino gaan en is de veiligheid van onze microcontroller gegarandeerd.

6 Programma

In bijlage B is de volledige Arduino schakeling met breadboard te zien. De sensor zelf is niet getekend op het schema, de luchtstroom monitor wel. Op het schema is een potentiometer en een knop te zien. De potentiometer is aanwezig voor manuele controle over de motor. Dit was noodzakelijk om de sensorspanning en windsnelheid te meten. De knop heeft als functie om de motor te resetten.

6.1 Resetknop

We weten dat bij het starten van het programma de ESC eerst moet weten op welke positie de motor zit. Dit werd vermeld bij de tegen-elektromagnetische kracht. Elke keer als de ESC de positie van de motor verloren is moet die dit proces opnieuw doen. Dit gebeurt als de ESC een te lage spanning krijgt of geen stroom krijgt. Het is nu zo dat als de ESC een commando krijgt dat boven de minimum puls breedte is terwijl de spoelpositie van de motor onbekend is ontstaat er een probleem. Namelijk de motor zal niet starten. De motor zal rukbewegingen maken en een biep geluidssignaal geven. Als we de motor toch willen starten is het noodzakelijk dat we het commando onder de minimum puls breedte zetten, bijvoorbeeld naar 1ms pulsbreedte. Als dit gebeurt zal de ESC terug de positie van de spoelen bepalen en

nu kunnen we de motor laten draaien en regelen. Dit is een veiligheidsfunctie van de ESC (Dejan, 2019b). Als deze functie niet aanwezig was zou de motor direct starten vanaf het moment dat die stroom krijgt. Dit kan tot een situatie leiden waar je verwondingen kan oplopen. Dus verplicht de ESC je om hem eerst een 0 commando te geven. Maar voor ons programma kan dit voor een probleem zorgen.
Bij het testen van de PID programma viel de motor veel stil wegens te lage spanning of door slechte contacten. Het PID programma detecteerde een groot verschil met het setpoint en vroeg aan de ESC om het maximum te geven (in ons programma is dat 40). Maar omdat de motor stilviel verloor de ESC de positie van de spoelen en moest hij ze terugvinden. De PID gaf een commando van 40 en de motor startte niet. Dit gebeurde ook bij het fuzzy logic programma. De enige manier om dit op te lossen was door het potentiometer programma te uploaden en de potentiometer naar 0 zetten. Maar dit was te onhandig voor het testen van de programma dus werd er besloten om een resetknop te plaatsen zoals op figuur 41.

Figuur 41 Resetknop
Bron: Eigen werk

Als men deze resetknop indrukt dan krijgt de ESC een commando van 0. De bedoeling van de knop is als er een probleem zou zijn met de motor en die valt stil dan druk je op de knop en wacht even voordat de ESC de positie bepaald heeft. Dan laat je de knop los en de motor zal terugdraaien. Deze knop heeft voorrang op beide regelaars.

6.2 PID programma

In bijlage C is de code voor de PID regelaar terug te vinden onder PID programma. We gebruiken de PID_v1 library voor de regelaar die geschreven is door (Beauregard, z.d.). Eerst definiëren we onze drie parameters. We maken een PID object die de drie parameters gebruikt en de input, output en setpoint. De ESC is ook als object gedefinieerd en wordt gelinkt met pin 9 met een puls breedte tussen 1ms en 2ms. We moeten de output limiteren

met een getal van 0 tot 40. Dit getal zal het commando zijn voor de ESC. We kunnen de library zelf niet veranderen. Volgens de library moet de input een getal zijn van 0 tot 255 dus zullen we die moeten omzetten. De setpoint zal ook een getal zijn van 0 tot 255. We gebruiken een IF ELSE structuur waarbij er eerst gekeken wordt of de knop ingedrukt is. Als dat het geval is stopt de motor met draaien. Als dat niet het geval is dan zal de PID regelaar de output sturen naar de ESC.

6.3 Fuzzy logic programma

De code van de fuzzy regelaar is te vinden in Bijlage C onder fuzzy logic programma. De fuzzy logic library is geschreven door (Alves, z.d.) De library gebruikt de Mamdani structuur en de *center of area* methode voor defuzzificatie. We definiëren terug onze ESC en resetknop.
De eerste stap in dit programma is de lidmaatschapsfuncties opstellen voor de input. We kiezen voor een driehoek vorm omdat dat het gemakkelijkste is om mee te werken. De fuzzy set in deze library heeft altijd vier punten. Om een driehoekvorm te krijgen moeten we dus de twee middelste punten gelijkmaken. We voegen deze functies toe aan onze fuzzy regelaar. Vervolgens stellen we lidmaatschapsfuncties op voor de output. Daarna is het tijd voor de fuzzy inferentie en regels. We stellen de regels op met een IF…THEN structuur voor onze lidmaatschapsfuncties. We zijn klaar met de setup en kunnen overgaan naar de fuzzificatie en defuzzificatie. Het antwoord van de defuzzificatie zal het commando zijn voor de ESC. De regels die we gaan gebruiken in de fuzzy logic programma zijn weergegeven in tabel 5

Tabel 5 Fuzzy logic regels
Bron: Eigen werk

IF	wind	=	geen	THEN	ESC	=	zeer snel
IF	wind	=	weinig	THEN	ESC	=	snel
IF	wind	=	gemiddeld	THEN	ESC	=	medium
IF	wind	=	veel	THEN	ESC	=	traag
IF	wind	=	maximum	THEN	ESC	=	minimum

7 Resultaten

Alle metingen worden genomen in de Arduino software. Helaas heeft dit programma zijn nadelen. De grafiek blijft lopen en we kunnen die niet stopzetten. Een ander probleem is dat de spanning van de sensor sterk oscilleert. Dit betekent een extra uitdaging voor de regelaar om stabiel te blijven.

7.1 PID regelaar

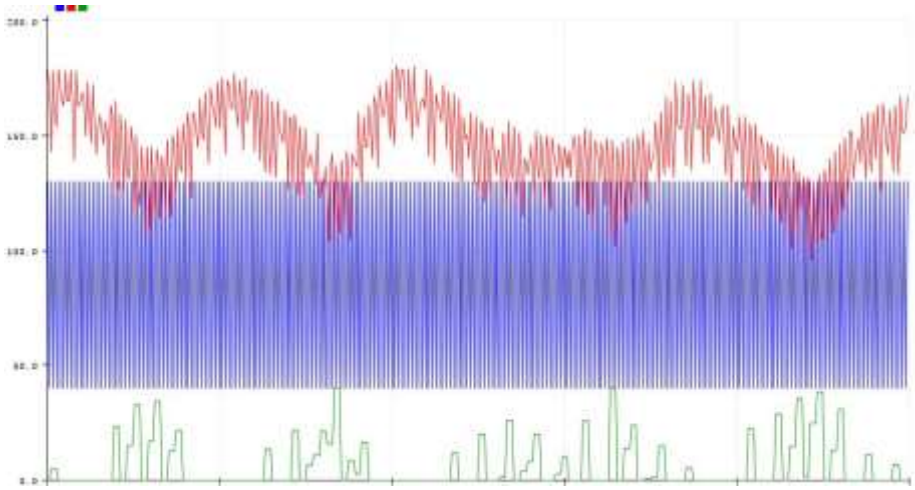
Voor onze drie parameters gebruiken we:

P=1

I=0,03

D=0,02

We gebruiken deze waarden omdat ze ons het beste resultaat geven. Deze waarden zijn gevonden door herhaaldelijk de waarden te veranderen en kijken hoe het systeem reageert. De I en D waarden zijn sterk verkleint omdat het systeem te onstabiel was en te snel reageerde. De P is op het einde ook verkleind omdat het systeem terug onstabiel werd.
We bepalen welke setpoints mogelijk zijn om mee te kunnen regelen. De setpoint is een getal van 0 tot 255. Als we de setpoint lager dan 100 zetten, wat ongeveer overeenkomt met 0,98V, zal de motor niet draaien omdat de piek van de oscillatie boven deze spanning zit. Zetten we de setpoint tussen 100 en 150 zal de motor het setpoint proberen te behalen door te draaien en direct stoppen. Dit gebeurt omdat de sensor al een spanning van 1,6V krijgt als de motor op zijn laagste toerental draait. Dus zal er altijd een doorschot aanwezig zijn met gevolg dat we een golf beweging krijgen van onze sensorspanning zoals te zien op figuur 42.



Figuur 42 PID met setpoint 130
Bron: Eigen werk

De betekenis van de lijnen is als volgt:

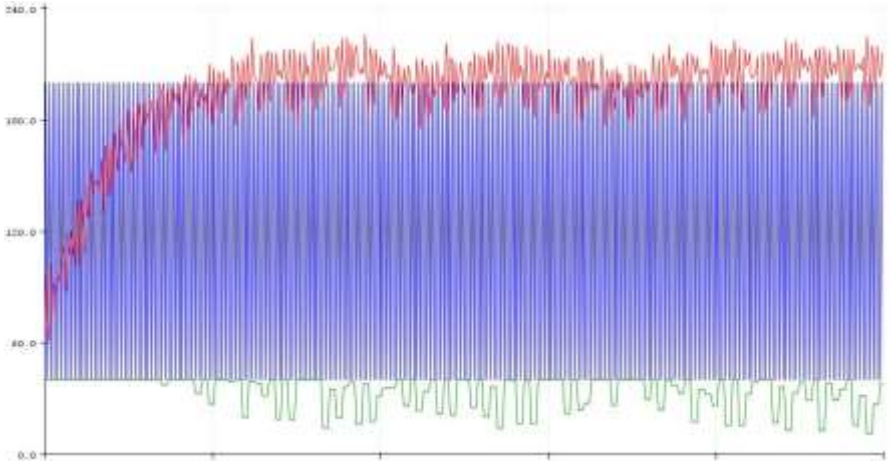
De rode lijn is de sensorspanning.

De groene lijn is de ESC commando.

De bovengrens van de blauwe balk is de setpoint.

De ondergrens van de blauwe balk is de maximum ESC commando (40).

Maar als we de setpoint naar bijvoorbeeld 200 zetten krijgen we een beter resultaat. Dit is afgebeeld op figuur 43.

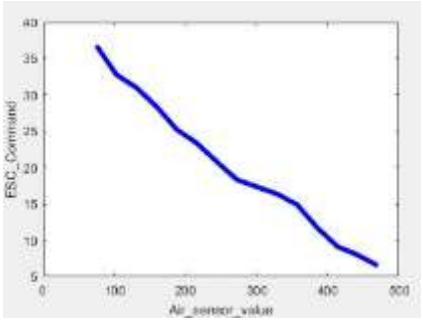


Figuur 43 PID met setpoint 130
Bron: Eigen werk

De reactietijd is tussen 1,7 seconden en 2 seconden. Ons doorschoot is zeer klein wat heel goed is. Het systeem is ook vrij stabiel. Na 10 tot 20 seconden krijgen we wel een stabiele-toestand fout met een fout van 10 tot 20 wat ongeveer overeenkomt met 0,1V tot 0,2V verschil. De regelaar kan dus vrij goed regelen met setpoints van 150 tot 210. Het enige probleem is dat de motor soms uitvalt en niet meer kan starten. Om dat op te lossen drukken we op de resetknop en zal de motor terugdraaien.

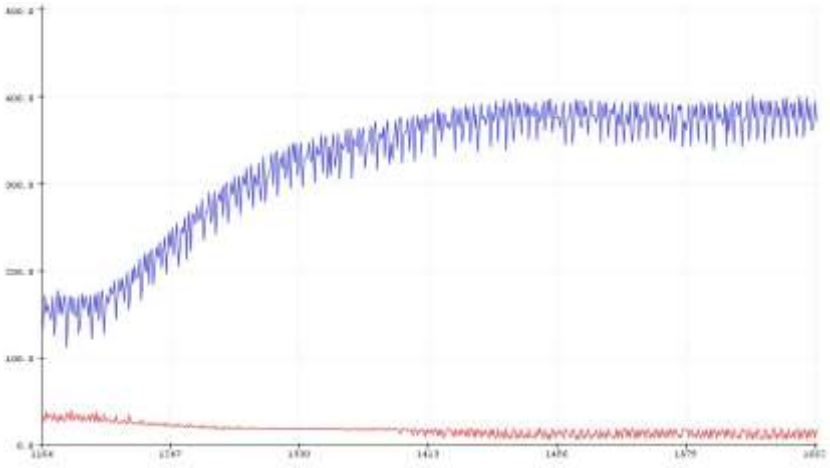
7.2 Fuzzy logic regelaar

Bij de fuzzy regelaar hebben we geen setpoints maar lidmaatschapsfuncties. Om ze te visualiseren zijn die in MATLAB gemaakt. Ze zijn te vinden in bijlage D. We gebruiken driehoeken voor onze lidmaatschapsfuncties. Dit is de gemakkelijkste vorm voor fuzzy logica. De driehoeken overlappen elkaar met een reden. Dit zorgt ervoor dat de regeling recht evenredig gebeurt. Dit is te zien op figuur 44.



Figuur 44 Relatie van ESC en sensorspanning
Bron: Eigen werk

Was dit niet het geval dan krijgen we meer horizontale lijnen wat betekent dat voor verschillende wind metingen de motor op dezelfde snelheid blijft draaien.
Jammer genoeg valt de motor hier ook stil. De resetknop komt weer van pas. Maar bij deze programma gebeurt dat vaker dan bij de PID programma dus zullen we de spanning verhogen naar 12V om dat te vermeiden. Het eerste wat ons opvalt, is dat er zo goed als geen doorschot is. Het stabiliseert ook veel sneller tegenover de PID. Op figuur 45 is de grafiek te zien van de fuzzy regelaar. De blauwe lijn is de sensorspanning in een schaal van 0 tot 1024 en de rode lijn is de ESC commando. Als we de ESC commando vergelijken met figuur 42 en 43 zien we een groot verschil. Bij de PID regelaar is die trapeziumvormig terwijl bij de fuzzy regelaar is het ESC commando driehoekvormig. Dit betekent dat de puls breedte constant verandert en nooit dezelfde breedte heeft. Dit is ook de reden waarom de motor vaker bij de fuzzy regelaar uitvalt. De motor krijgt constant verschillende draaisnelheden van de ESC op een korte tijd. Een intensieve sturing zoals deze kan ervoor zorgen dat de motor in de toekomst sneller stukgaat.


Figuur 45 Grafiek fuzzy logic regelaar
Bron: Eigen werk

8 Conclusie

We hebben in deze scriptie gezien hoe we een BLDC motor kunnen regelen met PID en fuzzy logic regeling door een gewenste luchtstroom te vragen. We hebben dit mogelijk gemaakt door middel van een luchtstroom sensor die de gemeten luchtstroom naar de Arduino stuurt. Om de sensor te beschermen van buitenlucht werd die geplaatst in een tunnel. Op deze opstelling zijn de motor, luchtstroom monitor, transformator en Arduino bevestigd. Helaas hebben we de verkeerde luchtstroom monitor besteld die geen analoog signaal kan uitsturen dus was het noodzakelijk om de sensorspanning rechtstreeks naar de Arduino sturen. De gemeten waarde oscilleerde sterk. Dit heeft een invloed op de regeling. De motor zal ook oscilleren om het constant veranderende signaal op de setpoint te krijgen. Als we kijken naar het verloop van de sensorspanning na regeling zien we dat bij beide regelaars het lukt om rond de setpoint te blijven.
De PID regelaar is makkelijker om te ontwerpen en te veranderen. Het kan de luchtstroom grotendeels stabiel krijgen als de setpoint niet te laag is. Er is een stabiele-toestand fout aanwezig en de sensorspanning slingert af en toe. De motor valt zelden uit.
Het opstellen van de fuzzy logic regelaar is moeilijker. We moeten kiezen welke lidmaatschapsfuncties voor ons systeem past. De regels zijn hier simpel omdat er 1 input en 1 output is. Het verloop van de sensorspanning gebeurt hier zonder doorschot. Het systeem is hier stabieler als je die vergelijkt met de PID regeling. De motor valt hier vaker uit dan bij de PID omdat de ESC commando bij de fuzzy logic regelaar constant verandert. We kunnen dat voorkomen door de spanning te verhogen.

De beste regeltechniek om de BLDC te regelen hangt uiteindelijk af van de toepassingen. Als je de motor wilt gebruiken in een omgeving waar het toerental constant moet zijn en altijd stabiel is gebruik je best een fuzzy logic regelaar. Er moet rekening gehouden worden dat de operator het systeem goed moet kennen om de correcte lidmaatschapsfuncties op te stellen. De PID regelaar is goed voor motoren die in veranderende systemen zitten. Het aanpassen van de regelaar aan het systeem gaat gemakkelijker dan bij de fuzzy logic controller. Dus als men het systeem goed kent en het verandert niet gebruikt men best fuzzy logic regelaars. Als men het systeem niet kent of het varieert wel gebruikt men een PID regelaar.

Bibliografie

abra-electronics. (z.d.). A2212/6T 2200KV Brushless DC Motor For RC Quadcopters Planes Boats Vehicles and DIY Kits. Geraadpleegd van https://abra-electronics.com/quadcopters/motors/mot-qc-02.html

Alves, A. J. (z.d.). *Alvesoaj/eFLL*. C++, . Geraadpleegd van https://github.com/alvesoaj/eFLL

Arduino. (2017, 29 augustus). Arduino—Introduction. Geraadpleegd van https://web.archive.org/web/20170829015201/https://www.arduino.cc/en/Guide/Introduction

Åström, K. J., & Hägglund, T. (1995). *PID controllers* (2nd ed.). Research Triangle Park, N.C: International Society for Measurement and Control.

Bansal, H. (2009). Tuning of PID Controllers using Simulink. *International Journal of Mathematical Modeling, Simulation and Applications*.

Baumer. (z.d.). Functionality of calorimetric flow sensors. Geraadpleegd van https://www.baumer.com/us/en/a/Know-how_Function_calorimetric_Flow-sensors

Beauregard, B. (z.d.). Arduino Playground—PIDLibrary. Geraadpleegd van https://playground.arduino.cc/Code/PIDLibrary/

Bhattacharyya, S., & Dutta, P. (2012). Fuzzy Logic: Concepts, System Design, and Applications to Industrial Informatics (pp. 33–71).

CFD support. (z.d.). Node334—CFD SUPPORT. Geraadpleegd van https://www.cfdsupport.com/openfoam-training-by-cfd-support/node334.html

Codecrucks. (2021, 21 augustus). Center of Gravity (CoG) method for defuzzification. *CodeCrucks*. Geraadpleegd van https://codecrucks.com/center-of-gravity-method-for-defuzzification/

Collins, D. (2021, 21 september). How do Hall effect sensors work and where are they used? Geraadpleegd van https://www.motioncontroltips.com/how-do-hall-effect-sensors-work-where-are-they-used-in-motion-applications/

Crowe, C. T., & Crowe, C. T. (Red.). (2009). *Engineering fluid mechanics* (9th ed.). Hoboken, NJ: Wiley.

ctms. (z.d.). Control Tutorials for MATLAB and Simulink—Introduction: PID Controller Design. Geraadpleegd van https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

Dejan. (2019a, 4 februari). How Brushless Motor and ESC Work. *How To Mechatronics*. Geraadpleegd van https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

Dejan. (2019b). Arduino Brushless Motor Control Tutorial | ESC | BLDC. *How To Mechatronics*. Geraadpleegd van https://howtomechatronics.com/tutorials/arduino/arduino-brushless-motor-control-tutorial-esc-bldc/

Digi-Key. (2016, 7 december). How to Power and Control Brushless DC Motors. *Digi-Key Electronics*. Geraadpleegd van https://www.digikey.be/nl/articles/how-to-power-and-control-brushless-dc-motors

electronicsforu. (2019, 5 januari). MOSFET Basic: Working Principle & Applications. *Electronics For You*. Geraadpleegd van https://www.electronicsforu.com/technology-trends/learn-electronics/mosfet-basics-working-applications

Ellis, G. (2012). *Control system design guide: Using your computer to understand and diagnose feedback controllers* (Fourth edition.). Amsterdam: Elsevier/BH.

Engineering Toolbox. (z.d.). Air—Dynamic and Kinematic Viscosity. Geraadpleegd van https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d_601.html

etechnog. (2019). Why diode is connected across IGBT and MOSFET? Body or Intrinsic Diode. *ETechnoG - Electrical, Electronics and Technology*. Geraadpleegd van https://www.etechnog.com/2019/07/body-diode-connected-mosfet.html

Gamazo-Real, J. C., Vázquez-Sánchez, E., & Gómez-Gil, J. (2010). Position and Speed Control of Brushless DC Motors Using Sensorless Techniques and Application Trends. *Sensors*, *10*(7), 6901–6947. doi:10.3390/s100706901

Halvaei, A., & Shahrbabak, M. (2014). Direct Power Control of Brushless DC Motor Drive. *Journal of Power Electronics & Power Systems*.

Hambley, A. R. (2018). *Electrical engineering: Principles and applications* (Seventh edition.). NY NY: Pearson.

Ibrahim, D. (2014). Chapter 8—Advanced PIC32 Projects. In D. Ibrahim (Red.), *Designing Embedded Systems with 32-Bit PIC Microcontrollers and MikroC* (pp. 359–442). Oxford: Newnes. doi:10.1016/B978-0-08-097786-7.00008-7

Khedr, S., Ammar, M., & Moustafa Hassan, M. (2013). *Multi Objective Genetic Algorithm Controller's Tuning For Non-linear Automatic Voltage Regulator",* (p. 863).

Kiam Heong Ang, Chong, G., & Yun Li. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, *13*(4), 559–576. doi:10.1109/TCST.2005.847331

Klir, G. J., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: Theory and applications*. Upper Saddle River, N.J: Prentice Hall PTR.

Knospe, C. (2006). PID control. *Control Systems Magazine, IEEE*, *26*, 30–31. doi:10.1109/MCS.2006.1580151

Krishnavedala. (2014). *English: Voltage divider network using resistors*. Geraadpleegd van https://commons.wikimedia.org/wiki/File:Resistive_divider2.svg

Laughton, M. A., & Warne, D. J. (2003). *Electrical engineer's reference book* (16th ed.). Oxford: Newnes.

Maes, R. (z.d.). *Automatisatie*. (Cursus, Hogere Zeevaartschool, Antwerpen, België).

Mamdani, E. (1974). Applications of fuzzy algorithms for control of a simple dynamic plant. doi:10.1049/PIEE.1974.0328

Minipro. (2018, 12 november). Understanding Motor Turns & Kv Rating. *MINIPRO®*. Geraadpleegd van https://www.minipro.com/blogs/articles/motor-turns-kv-rating

Minorsky., N. (1922). Directional Stability of Automatically Steered Bodies. *Journal of the American Society for Naval Engineers*, *34*(2), 280–309. doi:10.1111/j.1559-3584.1922.tb04958.x

Nasir, B. (2018). *Power Electronics 15 May 2018*.

National, I. (2020). PID Theory Explained. Geraadpleegd van https://www.ni.com/nl-be/innovations/white-papers/06/pid-theory-explained.html

Novák, V., & Perfilieva, I., Močkoř, Jiří. (1999). Mathematical Principles of Fuzzy Logic.

Parvin, K., Al-Shetwi, A., Hannan, M. A., & Ker, P. J. (2021). Modelling of Home Appliances Using Fuzzy Controller in Achieving Energy Consumption and Cost Reduction. *Elektronika ir Elektrotechnika*, *27*, 15–25. doi:10.5755/j02.eie.27000

Ramsden, E. (2006). *Hall-effect sensors: Theory and applications* (2nd ed.). Amsterdam ; Boston: Elsevier/Newnes.

Ross, T. J. (2010). *Fuzzy logic with engineering applications* (3rd ed.). Chichester, U.K: John Wiley.

SEIKOM. (z.d.-a). SEIKOM-Electronic GmbH & Co. KG. Geraadpleegd a van https://www.seikom-electronic.de/F2/en

SEIKOM. (z.d.-b). SEIKOM-Electronic GmbH & Co. KG. Geraadpleegd b van https://www.seikom-electronic.de/NLSW2A/en

Singh, H., Gupta, M. M., Meitzler, T., Hou, Z.-G., Garg, K. K., Solo, A. M. G., & Zadeh, L. A. (2013). Real-Life Applications of Fuzzy Logic. *Advances in Fuzzy Systems*, *2013*, e581879. Hindawi. doi:10.1155/2013/581879

sparkfun. (z.d.). Pulse Width Modulation—Learn.sparkfun.com. Geraadpleegd van https://learn.sparkfun.com/tutorials/pulse-width-modulation/all

Tay, T.-T., Mareels, I., & Moore, J. B. (1998). *High performance control*. Systems & control. Boston: Birkhäuser.

Unbehauen, H. (Red.). (2009). *Control systems, robotics, and automation*. Encyclopedia of life support systems. Oxford, U.K: Eolss Publishers.

Wikimedia Commons, B. b at. (2003). *English: Centroid defuzzification using max-min inferencing in a fuzzy control system.* Geraadpleegd van https://commons.wikimedia.org/wiki/File:Fuzzy_control_-_centroid_defuzzification_using_max-min_inferencing.png

Wilcher, D. (2012). *Learn electronics with Arduino*. [Berkeley, CA] : New York, NY: Apress ; Distributed to the book trade worldwide by Springer-Verlag New York.

Xia, C. (2012). *Permanent magnet brushless DC motor drives and controls*. Hoboken, N.J: Wiley-Science Press.

Yedamale, P. (2003). AN885, Brushless DC (BLDC) Motor Fundamentals, 20.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. doi:10.1016/S0019-9958(65)90241-X

Lijst van bijlagen

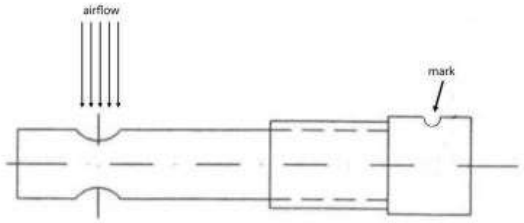
Installation Instruction:
Before setting up the switching point, the device should have been active for at least 2 minutes in normal conditions. To set up the switching point please attend the following steps:

- The sensors tip should be placed in the duct's middle and has to be flowed around completely by the medium.
- The flow in vertical-ducts needs to be upwards.
- To assure maximum reliability the sensor needs a length of the inlet path of 5xD (inside pipe diameter) and 3xD (inside pipe diameter) of the outlet path.
- The sensor is to be mounted only with its own hex-head screw.
- The sensor must be connected to the evaluation unit as described in its manual. Incorrect connection leads to malfunctioning and can destroy both!
- If the sensor's cable is laid in a conduit with other live cables (motor-, solenoid valve-cables, …) we recommend shielding it.
- If the length of the cable needs to be changed it needed to be done with a.w.g. 16 (1.5mm²) and must not be longer than 50m!

Maintenance information:
In order to avoid malfunction the sensor should be maintained in regular distances according to its pollution. Cleaning the sensor pay attention to following steps:

- Dismantle the sensor.
- Insert the sensor in slightly warm and soaped water carefully for about 10 minutes.
- Carefully rinse off the airflow sensor with lukewarm water.
- Assemble the airflow sensor.

Attention: Do not use screwdrivers or equal to clean the sensor!



Mistakes and misprints are not to be excluded. All information „without guarantee". 03/2022

♦SEIKOM-Electronic GmbH & Co. KG♦Fortunastr.20♦D-42489 Wülfrath♦
♦Telefon: +49(0) 20 58/20 44 ♦ Fax: +49(0) 20 58 / 79 111♦
♦E-Mail: info@seikom-electronic.com ♦ Internet: http://www.seikom-electronic.de♦

Airflow monitoring

Installation and operating instruction
Airflow sensor F2, F3, F4.2 F4.3

Our products correspond to the requirements of the European guidelines
WEEE 2012/19/EU - RoHS 2011/65/EU

UK CA

♦SEIKOM-Electronic GmbH & Co. KG♦Fortunastr.20♦D-42489 Wülfrath♦
♦Telefon: +49(0) 20 58/20 44 ♦ Fax: +49(0) 20 58 / 79 111♦
♦E-Mail: info@seikom-electronic.com ♦ Internet: http://www.seikom-electronic.de♦

For measuring the airflow of gaseous media in the media temperature range from -10… +80°C, -20… +120°C and –10… +90°C.
The influence of the media temperature in this range is compensated.
The Seikom airflow sensor measures airflow velocities in the range of 0.1..30m/s based on the calorimetric measuring principle.

Technical Data

Type ***Article-No.***	***F2 / F2M16*** ***50272 / 50272M16***	***F3 / F3M16*** ***50276 / 50276M16***
Range of media temperature	-10..+80°C	-20..+120°C
Thermal gradient	30K/min	30K/min
Immersion depth approx.	50mm */48mm	50mm */48mm
Process onnection	PG 7 */M16x1,5*	PG 7 */M16x1,5*
sensor material	MS58, nickel-plated	MS58, nickel-plated
pressure resistance	10bar	10bar
Connecting cable	2,5m / 3x0,5mm2	2,5m / 3x0,5mm^2
Protection class	IP67	IP67
electronics	NLSW2a.NLSW45-3 NLSW75-A, NLSW75-TE	NLSW2a.NLSW45-3 NLSW75-A, NLSW75-TE

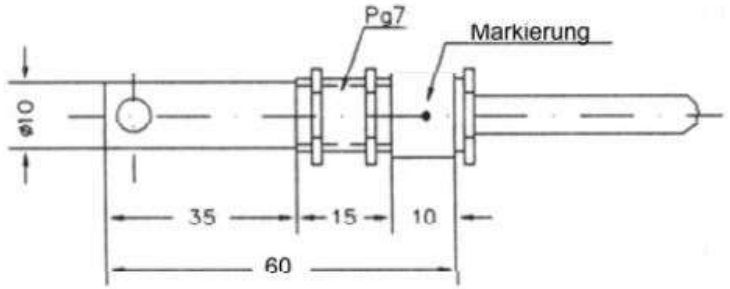
*** optional available in stainless-steel**

Type **Article-No.**	**F4.2** **50311**	**F4.3** **69829**
Range of media temperature	-20..+90°C	-20..+90°C
Thermal gradient	30K/min	30K/min
Immersion depth approx.	60mm	115mm
Process onnection	M12x1	12mm
sensor material	Teflon	Teflon
pressure resistance	4bar	6bar
Connecting cable	2,5m / 3x0,5mm²	2,5m / 3x0,5mm²
Protection class	IP67 **black/brown/grey* * grey replaces blue since 04/01/06**	IP67
electronics	NLSW2A/45-3/75A/75TE	NLSW2A/45-3/75A/75TE

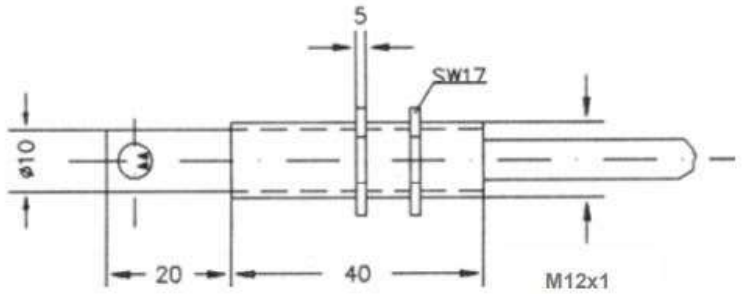
Type examination TÜV Nord DIN EN 61010-1:2011-07

Please attend to our appurtenances! Depths of immersion also available as: 130/165/240/300mm!
*diameter F2M16, F3M16 = 12mm
F4.2: Delivery without 2 teflon screw nut / optional accessory

Dimension F2, F3:



Dimension F4.2:



Sensor accessory

Reducing adapter	Art.-No.: 80399 - G1/2" to PG7
	Art.-No.: 80402 - M20 x 1.5 to PG7
	Art.-No.: 80403 - M16 x 1.5 to PG7
Flange	Art.-No.: 79781 - 10 mm Ø
Teflon screw nut	Art.-Nr.: 50311/M12 – Set with 2 pieces optional accesorry

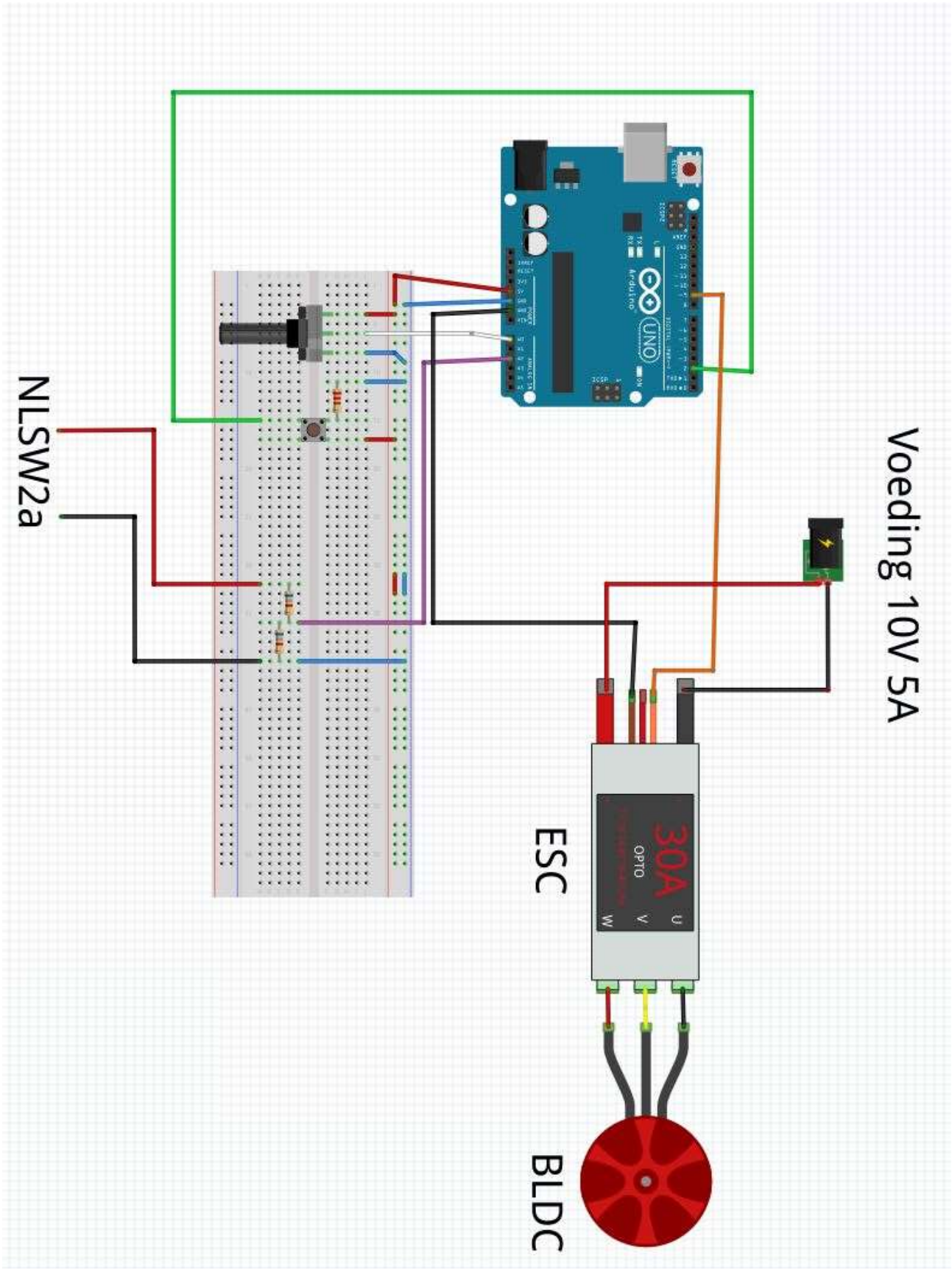
Technical data sensorcable (in accordance to DIN VDE 0281, 0293, 0295)

Temperature range	flexing	-15°C…+80°C
	fixed installation	-40°C…+80°C
Minimum bending radius	flexing	7,5 x cableØ
	fixed installation	4 x cableØ

For use without tensile stress or forced movements.
Not for outdoor use or direct burial.

Bijlage B

Schema Arduino



Bijlage C

[PID programma]

```
//  PID_v1 library geschreven door Brett Beauregard

#include <PID_v1.h>
#include <Servo.h>

Servo ESC;  //Maak een nieuwe servo object met naam "ESC"
const int buttonPin = 2;  //Definieer de pin van de Arduino voor de resetknop
int buttonStatus = 0;

double Setpoint; //De setpoint wordt uitgedrukt in een 10-bit getal
double Input; //De input is de sensorspanning van de F2 flow sensor
double Output; //De commando aan de ESC in een getal van 0 tot 40

double Kp=1, Ki=0.03, Kd=0.02;  //De 3 PID parameters die bepaalt hoe onze regelaar
reageert

PID AirPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);  //Maak een nieuwe PID object
met de 3 parameters en de SP, Input, Output

void setup() {
  ESC.attach(9,1000,2000); //Link de ESC met pin 9, definieer de pulsbreedtes=1000µs tot
2000µs
  Serial.begin(9600); //Start seriele monitor met een snelheid van 9600 bits per seconde

   Setpoint = 200;
   //Turn the PID on
   AirPID.SetMode(AUTOMATIC); //Laat de PID zelf regelen
   AirPID.SetTunings(Kp, Ki, Kd); //Gebruik de drie parameters om de PID te tunen
   AirPID.SetOutputLimits(0, 40); //Zet de minimum en maximum commando's voor de ESC
hier

}

void loop() {
  buttonStatus = digitalRead(buttonPin);
  int Air_sensorValue = analogRead(A2);
  Input = map(Air_sensorValue, 75, 470, 0, 255); //De gemeten digitale signaal omzetten naar
0 tot 255 signaal
  //PID calculation
  AirPID.Compute(); //PID voert berekeningen uit

  if (buttonStatus == HIGH) {  //Als resetknop ingedrukt is stopt de motor
     ESC.write(0);
  } else {
     analogWrite(9,Output);
```

```
    ESC.write(Output);    //Stuur de berekende Output als commando naar ESC
  }
  //Maak een grafiek
  Serial.print(40);
  Serial.print(' ');
  Serial.print(Input);
  Serial.print(' ');
  Serial.println(Output);
  Serial.print(' ');
  Serial.println(Setpoint);
}
```

[Fuzzy logic programma]

```
//Fuzzy library geschreven door A.J. Alves

#include <Fuzzy.h>
#include <Servo.h>

Servo ESC;
const int buttonPin = 2;
int buttonState = 0;

// Instantiating a Fuzzy object
Fuzzy *fuzzy = new Fuzzy();

void setup()
{
  ESC.attach(9,1000,2000);
  // Set the Serial output
  Serial.begin(9600);
  randomSeed(analogRead(0));

  // Instantiating a FuzzyInput object
  FuzzyInput *wind = new FuzzyInput(1);
  // Instantiating a FuzzySet object
  FuzzySet *geen_wind = new FuzzySet(25, 75, 75, 175);
  // Including the FuzzySet into FuzzyInput
  wind->addFuzzySet(geen_wind);
  // Instantiating a FuzzySet object
  FuzzySet *weinig = new FuzzySet(75, 175, 175, 275);
  // Including the FuzzySet into FuzzyInput
  wind->addFuzzySet(weinig);
  // Instantiating a FuzzySet object
  FuzzySet *gemiddeld = new FuzzySet(150, 275, 275, 395);
  // Including the FuzzySet into FuzzyInput
  wind->addFuzzySet(gemiddeld);
  //new set
  FuzzySet *veel = new FuzzySet(220, 370, 370, 500);
  wind->addFuzzySet(veel);
  //new set
  FuzzySet *maximum = new FuzzySet(370, 470, 470, 500);
  wind->addFuzzySet(maximum);
  // Including the FuzzyInput into Fuzzy
  fuzzy->addFuzzyInput(wind);

  // Instantiating a FuzzyOutput objects
  FuzzyOutput *ESC = new FuzzyOutput(1);
  FuzzySet *minimum = new FuzzySet(-20, 0, 0, 8);
  ESC->addFuzzySet(minimum);
  FuzzySet *traag = new FuzzySet(3, 14, 14, 25);
```

```
ESC->addFuzzySet(traag);
FuzzySet *med = new FuzzySet(5, 20, 20, 35);
ESC->addFuzzySet(med);

FuzzySet *snel = new FuzzySet(20, 30, 30, 40);
ESC->addFuzzySet(snel);
// Including the FuzzyOutput into Fuzzy
FuzzySet *zeer_snel = new FuzzySet(30, 40, 40, 50);
ESC->addFuzzySet(zeer_snel);
fuzzy->addFuzzyOutput(ESC);

// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *ifWindGeen = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
ifWindGeen->joinSingle(geen_wind);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenESCZeerSnel = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenESCZeerSnel->addOutput(zeer_snel);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule01 = new FuzzyRule(1, ifWindGeen, thenESCZeerSnel);
// Including the FuzzyRule into Fuzzy
fuzzy->addFuzzyRule(fuzzyRule01);

// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *ifWindWeinig = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
ifWindWeinig->joinSingle(weinig);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenESCSnel = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenESCSnel->addOutput(snel);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule02 = new FuzzyRule(2, ifWindWeinig, thenESCSnel);
// Including the FuzzyRule into Fuzzy
fuzzy->addFuzzyRule(fuzzyRule02);

// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *ifWindGemiddeld = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
ifWindGemiddeld->joinSingle(gemiddeld);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenESCMed = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenESCMed->addOutput(med);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule03 = new FuzzyRule(3, ifWindGemiddeld, thenESCMed);
// Including the FuzzyRule into Fuzzy
fuzzy->addFuzzyRule(fuzzyRule03);
```

```
  FuzzyRuleAntecedent *ifWindVeel = new FuzzyRuleAntecedent();
  ifWindVeel->joinSingle(veel);
  FuzzyRuleConsequent *thenESCTraag = new FuzzyRuleConsequent();
  thenESCTraag->addOutput(traag);
  FuzzyRule *fuzzyRule04 = new FuzzyRule(4, ifWindVeel, thenESCTraag);
  fuzzy->addFuzzyRule(fuzzyRule04);

  FuzzyRuleAntecedent *ifWindMax = new FuzzyRuleAntecedent();
  ifWindMax->joinSingle(maximum);
  FuzzyRuleConsequent *thenESCMinimum = new FuzzyRuleConsequent();
  thenESCMinimum->addOutput(minimum);
  FuzzyRule *fuzzyRule05 = new FuzzyRule(5, ifWindMax, thenESCMinimum);
  fuzzy->addFuzzyRule(fuzzyRule05);
}

void loop()
{
  buttonState = digitalRead(buttonPin);
  int input = analogRead(A2);

  fuzzy->setInput(1, input);
  // Running the Fuzzification
  fuzzy->fuzzify();
  // Running the Defuzzification
  float output = fuzzy->defuzzify(1);

  if (buttonState == HIGH) {
    ESC.write(0);
  } else {
    analogWrite(9,output);
    ESC.write(output);    // Send the signal to the ESC
  }
  // Printing something
   // Serial.println("\n\n\nEntrance: ");
 // Serial.print("\t\t\tWind: ");
  Serial.print(input);
  Serial.print(' ');
  Serial.println(output);
  Serial.print(' ');
}
```

Bijlage D

Lidmaatschapsfuncties