



# Projectverslag

## Air Filtration Control System

**Stagementor**

Ruben Mangelschots

**Stagecoördinator**

Filippo Bagnoli

Academiejaar 2023-2024

Campus Geel, Kleinhoefstraat 4  
BE-2440 Geel

**Student**

Harold De Ridder



# Voorwoord

Als kind droomde ik ervan om op een dag uitvinder te worden. Ik zou machines en systemen bouwen die het leven van de mens aangenamer en makkelijker zouden maken. Gedurende mijn kindertijd – en nu misschien mogelijks nog meer – werd mijn reeds hoge nieuwsgierigheid het meeste geprikkeld door technologie, innovatie, gadgets, en alles wat hier wat bij aansloot. Over de jaren heen ben ik dit dus altijd gaan opzoeken, waardoor ik eerst een opleiding Industrieel Ingenieur ben gestart aan Groep T in Leuven. De combinatie van de ruime waaier aan uitgaansmogelijkheden, totale vrijheid en de lager dan verwachte interesse voor heel wat vakken horende bij deze bachelor, zorgde ervoor dat ik deze opleiding jammer genoeg niet heb afgerond. Hierna waagde ik een tweede poging; de opleiding Toegepaste Informatica van Thomas More in Geel.

Na een algemeen vormend jaar opteerde ik in het tweede jaar voor het deeltraject Digital Innovation, een ietwat atypische richting aangezien hier nagenoeg alles in projectvorm werd gedaan en beoordeeld. Hier stak ik heel wat kennis op wat betreft elektronica, web development, artificial intelligence en nog zo veel meer. Desondanks ontbrak het mij nog steeds wat aan discipline, en slaagde ik uiteindelijk niet voor Digital Innovation. Waarom vertel ik dit allemaal? Omdat ik graag een eerlijk beeld geef van wie ik ben, ik niet ga ontkennen dat ik niet de beste student ben, maar wel ben blijven proberen om iets te zoeken wat me lag en waar ik mee verder wou. Dit parcours heeft me uiteindelijk gebracht bij een ander deeltraject, genaamd Internet of Things (IoT). Hier werden er wel gewoon verschillende vakken en lesmomenten ingepland, naast heel wat projecten van allerlei allooi. Hier voelde ik me, na lang zoeken, echt op mijn plek. Het was een goede balans tussen theorie en praktijk, met een overwicht aan de praktijkzijde. Aangezien ik altijd iemand ben geweest die graag wist hoe de zaken werkten en in elkaar zaten, maar vooral hoe je ze kon gebruiken en toepassen was dit een goede fit.

Voor mijn stageplek ging ik dan ook op zoek naar een bedrijf dat me interesseerde, maar vooral ook een stageopdracht aanbood die aansloot bij mijn kennis en vaardigheden. Tijdens mijn zoektocht moest ik plots denken aan het bedrijf genaamd CERcuits. Tijdens één van de lessen Embedded Devices Essentials had onze docent namelijk Frederik Luppens – De CEO van dit bedrijf – uitgenodigd om te komen vertellen waar zij mee bezig waren. Dit bedrijf specialiseert zich in het ontwerpen en maken van keramische printplaten, bedoelt voor de ontwikkeling van chips en andere elektronica die gebruikt zal worden in omgevingen met extreme temperaturen of omgevingen waar de elektronica over het algemeen tegen een stootje moet kunnen. Op een mooie dag besloot ik om er gewoon aan te kloppen en eens te horen of zij een stagestudent konden gebruiken.

Ik werd dadelijk enthousiast onthaald door Ruben Mangelschots, één van de werknemers van CERcuits. Hij gaf me een rondleiding doorheen het bedrijf en vertelde wat hij er al had gedaan in de periode waarin hij werkte bij CERcuits. We wisselden contactgegevens uit en na goedkeuring van Frederik, werkte Ruben een stageopdracht voor mij uit en nam hij de rol van stagementor op zich.

Deze opdracht bestond er in grote lijnen uit een systeem te ontwikkelen dat de lasers - die bij het productieproces van de printplaten gebruikt werden - te monitoren en in functie hiervan een stofafzuigstelsysteem automatisch aan- of uit te zetten. Hoofddreden hiervoor was het jaarlijks energieverbruik en de daarbij horende factuur naar beneden te krijgen. Analooq hieaan wou men graag de gemiddelde temperatuur in deze ruimte verlagen (het intensief gebruik van de blowers die dienden voor de afzuiging wekten veel warmte op aangezien deze quasi de hele dag aanstonden). Ik was onmiddellijk enthousiast om hiermee aan de slag te gaan, het sloot mooi aan bij de automatisatie- en monitoringopstellingen die we voor schoolprojecten hadden gebouwd en ik stond dus te popelen om te beginnen.

Ik nodig u uit om in dit uitgebreid realisatieverslag horende bij mijn stage te duiken en mijn evolutie doorheen dit project te zien. Vooraleer ik verder ga had ik wel graag nog enkele mensen bedankt aangezien zij er mee verantwoordelijk voor geweest zijn om dit alles tot een goed einde te brengen. In de eerste plaats Ruben Mangelschots, mijn stagementor, voor zijn enthousiasme, zijn flauwe maar zeer welkome humor op de werkvloer, de antwoorden op al mijn vragen en de vele ondersteuning die ik tijdens de volledige stage van hem gekregen heb. Daarnaast had ik ook graag Tim Christiaensen bedankt, voor het “zoeken naar gremlins (korsluitingen en dergelijke)” na het soldeer- en aansluitwerk, zijn technische ondersteuning bij zowel het coderen als het bouwen van mijn systeem en de tips en tricks die ik van hem kreeg tijdens mijn stage. Ook Frederik Luppens verdient mijn dankbaarheid, dit dan vooral voor het vertrouwen, de carte blanche wat betreft het bestellen van componenten en de kans om bij te leren die hij mij gaf, door mij te verwelkomen als stagestudent en dus te mogen proeven van het bedrijfsleven. Tot slot had ik graag mijn stagecoördinator, Filippo Bagnoli, ook mijn dank getoond voor zijn immer aangename tussentijdse evaluaties, zijn bijsturing waar nodig en de beschikbaarheid bij vragen of opmerkingen gedurende het hele semester. Ik ben enorm blij dat ik met hem en alle andere genoemde mensen heb mogen samenwerken en kijk dus met veel plezier terug op deze leerrijke periode!

# Inhoudsopgave

Voorwoord .....	2
Inleiding .....	6
Begrippenlijst .....	8
Figurenlijst .....	10
Technische beschrijving .....	12
Startsituatie .....	12
Systeemvereisten .....	13
Hardware .....	14
Bill of Materials (BOM) .....	14
Extra benodigdheden: .....	21
ESP32 Devkit .....	22
Breadboard .....	23
Jumper Wire Kit .....	24
Breakout Board ESP32 .....	25
Perfboard .....	26
Optocoupler (2-kanaals) .....	27
Relaisbord (4-kanaals & 1-kanaals) .....	28
Schakelaar (3 inputs) .....	30
Signaallamp + zoemer .....	31
Contactor (Siemens 3RT1015-1BB41) .....	32
Voeding 24V + 5V .....	33
Voedingskabel 230V .....	34
WAGO Verbindingsklem .....	35
Differentiële druksensor (LWLP5000-5XD) .....	36
Rubberen slang .....	37
Potentiometer 10k .....	38
Draaiknop .....	38
Aftakdoos .....	39
DIN-5 Panel mount .....	39
Wartels .....	40
Dubbele jack 3.5mm (5m & 10m) .....	40
Jack 3.5mm Panel mount .....	41
Stekkerhulskit + krimptang .....	41

Adereindhulskit + krimptang .....	42
Schema's .....	43
Globaal schema .....	43
Elektrisch schema .....	43
Aansluittabel ESP32 .....	44
Schema plaatsing componenten .....	45
Schema onderkant aftakdoos .....	45
Schema voorzijde aftakdoos .....	46
Software .....	47
Confluence .....	47
EasyEDA .....	47
Het programma zelf .....	48
Troubleshooting .....	58
Verbetervoorstellen .....	59
Noodstop .....	59
Internetverbinding AFCS .....	60
Online dashboard .....	60
Extra .....	60
Besluit .....	60
Bibliografie .....	60

# Inleiding

Zoals ik reeds aanhaalde, bestond mijn stageopdracht eruit een systeem te ontwerpen én te bouwen om zowel de kostprijs wat betreft energie te verlagen, als het comfort van de werknemers te verhogen. Dit systeem zou enerzijds dienen ter monitoring van het gebruik van de lasers, anderzijds voor het automatisch aan- en uitzetten van de stofafzuiginstallatie. De naam die eraan werd toegekend luidt als volgt: Air Filtration Control System, kortweg AFCS. Bij aanvang van mijn stage is de situatie de volgende: de bestaande stofafzuiginstallatie werkt, maar staat bijna 24/24 aan. Vaak is de laser nog operationeel of wordt er zelfs een nieuwe bewerking gestart tegen het einde van de werkdag. Zo winnen de werknemers bij CERcuits aan tijd, want dan is er tegen de dag nadien een printplaat klaar zonder dat ze er tijd mee verliezen tijdens de werkdag. Helaas kent dit ook enkele nadelen:

- De blowers zijn hierdoor ook nog operationeel nadat de bewerking is afgerond en blijven tot de ochtend nadien blazen, wat onnodig energieverbruik en dus een onnodig hoge energiefactuur betekent.
- Doordat deze blowers veel langer aanstaan dan nodig, verslijten deze heel wat sneller en heeft men dus op termijn ook veel meer kosten om telkens nieuwe blowers aan te schaffen.
- 4 blowers van elk 600W genereren elk ook serieus wat warmte. In de zomermaanden is het dus geen pretje voor de werknemers om de hele tijd te werken in een ruimte waar het lijkt alsof de verwarming opstaat terwijl het buiten 25-30 graden is. In de winter valt er iets te zeggen over deze extra warmte omdat het dan net welkom is om minder te moeten verwarmen met traditionele verwarming. Deze analyse heb ik echter niet uitgevoerd, al kan het wel interessant zijn om hier in de toekomst eens naar te kijken. Op deze warmtecreatie ga ik nog verder in bij het hoofdstuk “Verbetervoorstellen”, onderaan dit verslag.
- Daarnaast genereren deze blowers gezamenlijk ook heel wat lawaai, waardoor het voor de werknemers zeker wenselijk is als het volume soms wat verlaagt doordat niet alle blowers continu ingeschakeld zijn.

Laat mij u leiden doorheen het hele ontwikkelingsproces van zowel mezelf als IT’er, als van het systeem dat op dit moment operationeel is bij CERcuits. Elk project start uiteraard met een fase van onderzoek en dat was hier dus niet anders. Door de verschillende projecten die ik door mijn opleiding reeds achter de kiezen had, had ik al snel een idee van hoe AFCS er in grote lijnen zou kunnen uitzien. Dit betreft welk type microcontroller we zouden kunnen gebruiken, welke sensoren het beste aansloten bij onze noden en zo voort. Het was een goede balans tussen de vrijheid om zelf keuzes te maken wat essentieel was en zaken toe te voegen of weg te laten na een bespreking met Ruben, mijn stagementor. We staken onze koppen regelmatig bijeen, waardoor bestellijsten verbeterd werden, functionaliteiten werden toegevoegd en het uiteindelijke systeem robuuster werd. Ik heb enorm veel bijgeleerd van Ruben en andere collega’s gedurende het gehele proces, voornamelijk naar het einde toe. Al denk ik dat ook Ruben hier en daar ook wel eens iets heeft opgestoken van mij – althans dat hoop ik toch.

Het verslag is zo ingedeeld dat er gesproken wordt over de gebruikte hardware en de verschillende opties die voor dit systeem in overweging zijn genomen. De functionaliteiten van het systeem en de daarvoor geschreven code wordt ook in detail toegelicht, gevolgd door de aansluitschema's van alle gebruikte componenten.

Enkele mogelijke problemen en de hiervoor gevonden oplossingen nam ik ook op in dit verslag, tot slot gevolgd door enkele verbetervoorstellen die ik nog in gedachten had maar waarvoor ik de tijd niet meer heb gevonden binnen de afgebakende stageperiode. Deze zaken hebben volgens mij weldegelijk toegevoegde waarde en kunnen in sommige gevallen zelfs in verband gebracht worden met elementen binnen het bedrijf die er op het eerste zicht niet onmiddellijk iets mee te maken hebben.

Het ultieme doel van dit hele rapport is dus aantonen waarom en op welke manier heel dit project een positieve impact heeft gehad voor CERcuits. Daarnaast hoop ik ook dat dit een inspiratie kan zijn voor ieder die misschien zelf al het idee had om zijn of haar eigen bedrijf een boost te geven en zelf een IoT-oplossing wil creëren voor een heersend probleem op de werkvloer of thuis. Veel succes alvast indien dat het geval is!



# Begrippenlijst

Internet of Things (IoT)	<p>Vrij vertaald: het internet der 'dingen'. Onder 'dingen' verstaat men meestal allerlei verschillende stukjes elektronica. Dit kan gaan over zaken als sensoren, motoren, microcontrollers, slimme toestellen vanuit domoticasystemen, ... .</p> <p>Het 'internet'-aspect komt dan weer neer op het feit dat al deze elementen met elkaar kunnen verbonden worden, meestal zelfs draadloos. Hierdoor kunnen sensoren op plaats A bijvoorbeeld iets meten en kunnen deze gegevens naar een lokaal netwerk of zelf 'het internet' worden gestuurd. Hier kunnen ze gebruikt worden voor analyses in de cloud of kunnen ze worden gebruikt om bijvoorbeeld op plaats B een toestel aan te zetten.</p> <p>Een praktisch voorbeeld:</p> <p>Ooit kreeg ik de opdracht om samen met mijn medestudenten uit de richting IoT een pompoenserre te automatiseren. Hiervoor installeerden we eerst sensoren die temperatuur, luchtvochtigheid en CO2 monitorden.</p> <p>Deze data werd gebundeld op een microcontroller en vervolgens over een Wi-Fi-verbinding doorgestuurd naar een lokaal draaiende server.</p> <p>Deze data werd gebruikt om het automatisch openen of sluiten van de ramen te triggeren. Ook konden gigantische ventilatoren worden ingeschakeld wanneer de lucht te vochtig werd.</p> <p>Steeg de temperatuur boven een bepaalde grenswaarde, dan gingen de ramen automatisch open en vice versa.</p> <p>Werd het te vochtig in de serre, dan gingen de deuren open en werden de ventilatoren ingeschakeld.</p> <p>Dit is slechts één voorbeeld uit de lijst van talloze mogelijkheden die er zijn met IoT-toestellen. Onthoud vooral dat het bouwen van een IoT-systeem meestal specifiek voor 1 'probleem' is, dat het monitoring , meestal (inter)netwerkverbinding vereist en er geautomatiseerde acties gekoppeld worden aan de gemeten gegevens.</p>
--------------------------	---

Sensor(en)	<p>Elektronica die in staat is om dingen te <b>detecteren</b>, te <b>meten</b> en deze meting te presenteren aan een ander systeem dat hier een actie aan kan koppelen.</p> <p>Denk aan 'to sense' in het Engels, wat 'voelen' betekent.</p>
Actor(en)	<p>Elektronica die in staat is om iets te <b>doen</b>. Dit zijn bijvoorbeeld zaken als: motoren, lampen, ...</p> <p>Denk aan 'to act' in het Engels, wat 'iets doen, handelen' betekent.</p>
Microcontroller	<p>Een soort minicomputer die niet zo krachtig is als bijvoorbeeld een gewone laptop, maar wel in staat is om verwerking van kleine hoeveelheden gegevens of berekeningen te doen. Zo kan het gegevens van sensoren verwerken en actoren aansturen.</p> <p>Kan worden geprogrammeerd en kan in heel wat toepassingen gebruikt worden. Bevat naast fysieke aansluitmogelijkheden ook soms draadloze opties zoals Bluetooth of Wi-Fi.</p> <p>Het wordt veelvuldig gebruikt in de IoT-wereld om zaken te monitoren of te automatiseren. Daarnaast zit het bijvoorbeeld ook vaak in andere toestellen waar vele knoppen op zitten, waar draadloze communicatie voor nodig is of waar een schermpje inzit. Denk aan afstandsbedieningen, Spelconsoles, ...</p>
GPIO pinnen	<p>GPIO is de afkorting voor: <b>G</b>eneral <b>P</b>urpose <b>I</b>nterface <b>O</b>utput. Deze pinnen vind je terug op bijvoorbeeld de ESP32 en zijn dus pinnen die dienst kunnen doen als Input óf Output. Waarvoor ze specifiek zullen gebruikt worden en in welke modus de ESP32 ze zal moeten interpreteren wordt programmatorisch bepaald. Soms zijn bepaalde pinnen al standaard geconfigureerd om een bepaalde functie uit te voeren. Desondanks kan je deze pinnen meestal ook nog een andere functie geven indien gewenst.</p>

# Figurenlijst

Figuur 1: Blower 600W .....	12
Figuur 2: Afzuigkap + afzuigbuis .....	12
Figuur 3: ESP32 .....	14
Figuur 4: Optocoupler .....	14
Figuur 5: Relaisbord 4-kanaals .....	15
Figuur 6: Relaisbord 1-kanaals .....	15
Figuur 7: Voedingskabel 230V .....	15
Figuur 8: Dubbele voeding (24V & 5V) .....	15
Figuur 9: Differentiële druksensor .....	16
Figuur 10: Schakelaars (3 inputs) .....	16
Figuur 11: Driekleurenlamp + zoemer .....	16
Figuur 12: Contactor .....	16
Figuur 13: Potentiometer .....	17
Figuur 14: Aftakdoos .....	17
Figuur 15: Kabel type DIN 5 .....	17
Figuur 16: DIN 5 Panel Mount .....	17
Figuur 17: Wartel .....	17
Figuur 18: Dubbele Jack (AUX) 5m .....	18
Figuur 19: Dubbele Jack (AUX) 10m .....	18
Figuur 20: Jack Panel Mount .....	18
Figuur 21: Stekkerhulskit .....	18
Figuur 22: Adereindhulzenkit .....	18
Figuur 23: Draaiknop .....	19
Figuur 24: WAGO-Klem .....	19
Figuur 25: Jumper Wire Kit .....	19
Figuur 26: Rubberen slang (6mm) .....	19
Figuur 27: Breakout Board ESP32 .....	20
Figuur 28: Koperdraad (rood) .....	20
Figuur 29: Koperdraad (zwart) .....	20
Figuur 30: ESP32 - groot .....	22
Figuur 31: Breadboard - groot .....	23
Figuur 32: Breadboard schematisch .....	23
Figuur 33: Jumper Wire Kit - groot .....	24
Figuur 34: Breakout board ESP32 - groot .....	25
Figuur 35: Perfboard leeg .....	26
Figuur 36: Perfboard met componenten .....	26
Figuur 37: Perfboard onderzijde .....	26
Figuur 38: Optocoupler - groot .....	27
Figuur 39: Werking Optocoupler .....	28
Figuur 40: Relaisbord 4-kanaals - groot .....	28
Figuur 41: Relais werking .....	29
Figuur 42: Schakelaar 3 inputs - groot .....	30
Figuur 43: Tuimelschakelaar .....	30
Figuur 44: Signaallamp + zoemer - groot .....	31
Figuur 45: Contactor - groot .....	32

Figuur 46: Voeding - groot.....	33
Figuur 47: Voedingskabel 230V - groot .....	34
Figuur 48: Druksensor - groot.....	36
Figuur 49: Wartel - groot.....	40
Figuur 50: Dubbele Jack - groot.....	40
Figuur 51: Jack Panel mount - groot.....	41
Figuur 52: Stekkerhulskit - groot .....	41
Figuur 53: Adereindhulskit - groot .....	42
Figuur 54: AFCS Schematisch .....	43
Figuur 55: Elektrisch Schema AFCS.....	43

# Technische beschrijving

## Startsituatie

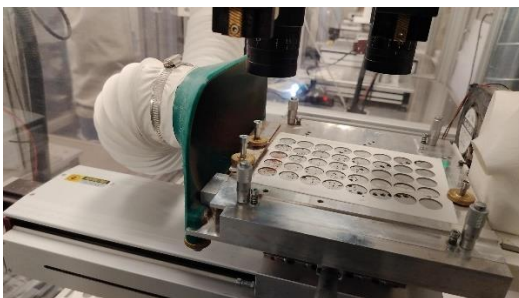
Bij aanvang van de stage zijn er reeds 4 productielijnen aanwezig, elk met 3 lasers (12 lasers in totaal). Ook een zelfgebouwd systeem voor de stofafzuiging is al aanwezig en operationeel.

Dit systeem bestaat uit:



Figuur 1: Blower 600W

- Een industriële blower van 600 Watt inclusief filterzak



Figuur 2: Afzuigkap + afzuigbuis

- Een 3D-geprinte afzuigkap.
- Een afzuigbuis voor de afvoer van stofdeeltjes richting de filterzak.

Per productielijn van 3 lasers is er dus 1 blower voorzien, die verbonden wordt aan de hand van afzonderlijke afzuigbuizen tot bij elke laser. Vooraleer men met een bewerking kan starten dient dit gehele systeem handmatig te worden opgezet. Hiervoor opent men de klep van de noodstop en drukt men op de groene knop. De blower wordt gestart en alle stofdeeltjes zullen nu dus worden weggezogen zo lang de blower ingeschakeld is.

## ***Systeemvereisten***

Mijn opdracht bestaat eruit 1 systeem per productielijn te voorzien met als doel de activiteit van de lasers te monitoren en in functie hiervan AFCS op te zetten. Dit systeem bevat 3 verschillende standen:

- 1) Uit-modus (0); de blower wordt handmatig uitgezet ongeacht de status van de lasers.
- 2) Manuele modus (I); de blower wordt handmatig opgezet ongeacht de status van de lasers.
- 3) Auto-modus(II); de blower wordt automatisch opgezet in functie van de status van de lasers.

Het systeem moet in modus II (Auto-modus) de volgende functionaliteiten bevatten:

- Detecteren wanneer een laser word opgezet of reeds met een bewerking bezig is.
- De blower die zorgt voor de afzuiging automatisch opzetten indien minstens één van de 3 lasers van desbetreffende productielijn actief is.
- Een timer starten wanneer alle 3 de aangesloten lasers inactief zijn.
- De blower van het systeem automatisch uitzetten wanneer de inactiviteit van alle 3 de lasers minstens 15 minuten bedraagt.

Het systeem moet in alle modi (Auto, Uit & Manueel) deze functionaliteiten bevatten.

- De status van de filterzakken monitoren en aangeven wanneer deze vol zitten en dus vervangen moeten worden.
- Signaleren aan de hand van licht en geluid wanneer de status van de blower zal wijzigen. Dit wil zeggen wanneer deze zal inschakelen of uitschakelen.



## Hardware





# Inleiding

Bij het opzoekingswerk wat betreft de nodige hardware, was (een zo laag mogelijke) kostprijs altijd een prioriteit. Dit kon zich ook uitdrukken in een verschil in verzendkosten. Als ik bijvoorbeeld component X aan 4 euro geprijsd zag op website A en aan 3 euro op website B, koos ik er toch soms voor om component X te bestellen op website A, als er nog iets anders was wat ik daar kon bestellen en daardoor vermeed om 2 keer afzonderlijke verzendkosten te betalen. Gebruiksgemak was zeker ook niet onbelangrijk, dus als er weinig prijsverschil was en ik 1 van de componenten al eens had gebruikt, koos ik voor die bepaalde component.





## Bill of Materials (BOM)






Hieronder vind je een overzicht van de benodigde materialen voor het bouwen van 1 AFCS.






Naam	Foto	Prijs/stuk (€)	Aantal	Totaal (€)
ESP32 Devkit	 <i>Figuur 3: ESP32</i>	8.55	1	8.55
Optocoupler 2-kanaals	 <i>Figuur 4: Optocoupler</i>	3.00	3	9.00



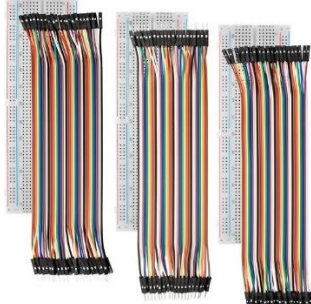

<b>Relaisbord</b> <b>4-kanaals</b>	 <p><i>Figuur 5: Relaisbord 4-kanaals</i></p>	7.00	1	7.00
<b>Relaisbord</b> <b>1-kanaals</b>	 <p><i>Figuur 6: Relaisbord 1-kanaals</i></p>	2.25	1	2.25
<b>Voedingskabel</b> <b>230V</b>	 <p><i>Figuur 7: Voedingskabel 230V</i></p>	4.00	1	4.00
<b>Voeding</b> <b>24V + 5V</b>	 <p><i>Figuur 8: Dubbele voeding (24V &amp; 5V)</i></p>	19.32	1	19.32



<b>Differentiële druksensor</b>  <b>LWLP5000-5XD</b>	 <p><i>Figuur 9: Differentiële druksensor</i></p>	39.73	1	39.73
<b>Schakelaar</b>  <b>3 inputs</b>  <b>(2-pack)</b>	 <p><i>Figuur 10: Schakelaars (3 inputs)</i></p>	4.89	1	4.89
<b>Signaallamp</b>  <b>3 kleuren</b>  <b>+ zoemer</b>	 <p><i>Figuur 11: Driekleurenlamp + zoemer</i></p>	17.68	1	17.68
<b>Contactor</b>  <b>Siemens</b>  <b>3RT1015-1BB41</b>	 <p><i>Figuur 12: Contactor</i></p>	57.75	1	57.75

<b>Potentiometer</b>  <b>10k</b>	  <i>Figuur 13: Potentiometer</i>	0.50	1	0.50
<b>Aftakdoos</b>	  <i>Figuur 14: Aftakdoos</i>	19.70	1	19.70
<b>DIN 5 Kabel (3m)</b>	  <i>Figuur 15: Kabel type DIN 5</i>	5.79	1	5.79
<b>DIN 5 connector</b>  <b>Inbouw</b>  <b>Vrouwelijk</b>	  <i>Figuur 16: DIN 5 Panel Mount</i>	1.09	2	2.18
<b>Wartel</b>	  <i>Figuur 17: Wartel</i>	0.63	1	0.63

<b>Kabel (5m)</b>  <b>Dubbele Jack</b>  <b>3.5 mm</b>	  <i>Figuur 18: Dubbele Jack (AUX) 5m</i>	5.29	2	10.58
<b>Kabel (10m)</b>  <b>Dubbele Jack</b>  <b>3.5 mm</b>	  <i>Figuur 19: Dubbele Jack (AUX) 10m</i>	10.99	1	10.99
<b>Jack connector</b>  <b>Inbouw</b>  <b>Vrouwelijk</b>	  <i>Figuur 20: Jack Panel Mount</i>	1.09	3	3.27
<b>Stekkerhulskit</b>  <b>+ tang</b>	  <i>Figuur 21: Stekkerhulskit</i>	28.91	1	28.91
<b>Adereindhulzenkit</b>  <b>+ tang</b>	  <i>Figuur 22: Adereindhulzenkit</i>	19.99	1	19.99

<b>Draaiknop</b>	 <p><i>Figuur 23: Draaiknop</i></p>	0.50	1	0.50
<b>Verbindingsklem WAGO (50st)</b>	 <p><i>Figuur 24: WAGO-Klem</i></p>	16.15	1	16.15
<b>Jumper wires</b>	 <p><i>Figuur 25: Jumper Wire Kit</i></p>	9.99	1	9.99
<b>Rubberen slang</b> <b>ø 2 mm binnen</b> <b>ø 6 mm buiten</b>	 <p><i>Figuur 26: Rubberen slang (6mm)</i></p>	2.75/m	2m	5.50

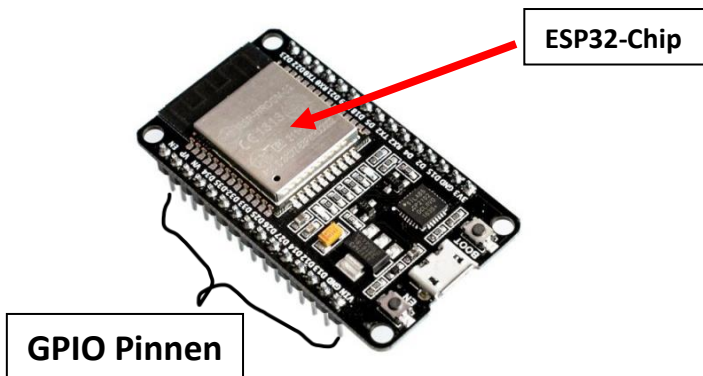
<b>Breakout bord</b>  <b>ESP32</b>	 <p><i>Figuur 27: Breakout Board ESP32</i></p>	3.90	1	3.90
<b>Koperdraad (10m)</b>  <b>Rood</b>	 <p><i>Figuur 28: Koperdraad (rood)</i></p>	1.95	1	1.95
<b>Koperdraad (10m)</b>  <b>Zwart</b>	 <p><i>Figuur 29: Koperdraad (zwart)</i></p>	1.95	1	1.950
<b>Weerstanden 10k</b>				
<b>Diode</b>				
<b>Boutjes (2mm)</b>				
<b>Moertjes (2mm)</b>				

## ***Extra benodigdheden:***

Naast bovenstaande onderdelen zijn er nog wat andere zaken die ik gebruikt heb tijdens mijn project. Ik heb ze niet opgenomen in de Bill of Materials omdat ze reeds aanwezig waren op mijn stagebedrijf en ze niet echt onderdelen vormen van mijn systeem. Desondanks heb je ze wel nodig om een AFCS te kunnen bouwen. Het zijn zaken die je meestal wel vindt in FabLabs of bij de doorgewinterde elektronicus. Hieronder vind je een lijstje van de dingen die ik zeker heb gebruikt.

- Multimeter
- Boormachine / schroefmachine
- Boorkoppen (2mm – 1cm)
- Soldeerbout & soldeertin
- Soldeerknecht
- Heat gun
- Krimpkousen
- Mini schroevendraaiersetje (0.2mm tot 0.6 mm, platte kop)
- Schroevendraaiers
- Lijmpistool & hot glue sticks
- Kniptang
- Striptang
- Liniaal
- Potlood
- Caliper
- Breekmes
- Figuurzaag of wipzaag

## ESP32 Devkit

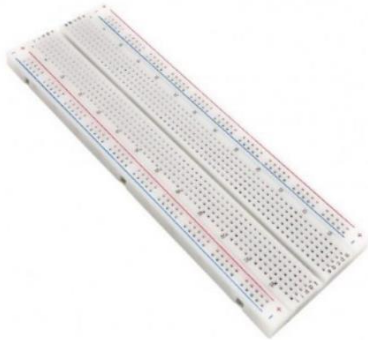


*Figuur 30: ESP32 - groot*

Als centraal element, ter aansturing van dit systeem, koos ik ervoor om een ESP32 Devkit microcontroller te gebruiken. Dit is een vrij goedkoop bordje (8.55 €) dat razend populair is in de wereld van IoT en de hobby-elektronicus. Het Bevat in hoofdzaak een ESP32-chip en allerlei aansluitmogelijkheden. Het is programmeerbaar in zowel C als in de programmeertaal van Arduino. Het heeft Wi-Fi en Bluetooth standaard aan boord, alsook een groot aantal (30+) GPIO pinnen die zowel als in- of uitgang gebruikt kunnen worden om sensoren of actoren op aan te sluiten. Als we op zoek gaan naar een Arduino bordje dat hetzelfde aantal bruikbare pinnen aan boord heeft, komen we meestal duurder uit dan de ESP32, en dienen we soms nog aparte shields te kopen als we ook Wi-Fi, bluetooth of extra aansluitmogelijkheden willen kunnen gebruiken. Daarnaast had ik zelf tijdens mijn opleiding ook altijd het advies gekregen om ESP32 te gebruiken voor projecten, dus was dat hier ook de logische keuze.

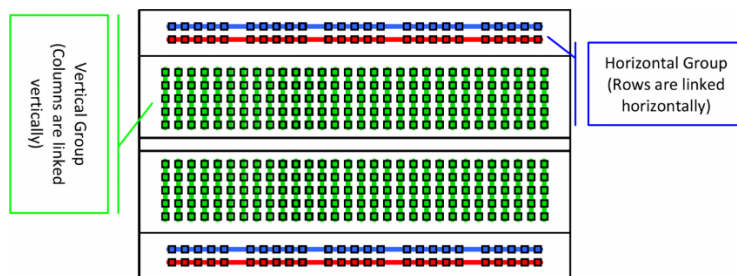
De ESP32 vereist een voedingsspanning van 5 Volt, die door een externe spanningsbron geleverd dient te worden. Deze spanning kunnen we verkrijgen door een spanningsbron met 5V uitgangsspanning aan te sluiten. Over de gekozen spanningsbron vindt u hieronder meer informatie terug. Alle andere pinnen van de ESP32 verdragen maximaal 3.3 Volt. Hier dienen we rekening mee te houden wanneer we sensoren en actuatoren kiezen om aan te sluiten op de ESP32.

## Breadboard



Figuur 31: Breadboard - groot

Vooraleer ik begon met solderen, boren en zagen, kortom: met het in elkaar steken van een permanent systeem, diende ik een testopstelling te bouwen. Een testopstelling is zelden van de eerste keer juist en is onderhevig aan veel verandering. Componenten worden al eens fout aangesloten of blijken niet compatibel te zijn met andere elementen in de opstelling. Pinnen van sensoren worden aan de verkeerde pinnen van de ESP32 aangesloten, noem maar op. In deze fase van het project is het zeer handig om gebruik te maken van een Breadboard. Dit is een component die bestaat uit vele rijen gaatjes die langs de onderkant in groepen zijn doorverbonden.

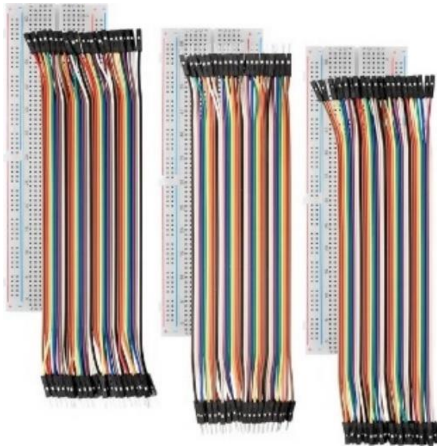


Figuur 32: Breadboard schematisch

Op de tekening hierboven zie je dat er horizontaal blauwe en rode lijnen zijn getrokken. Deze zijn over de hele lengte doorverbonden en kunnen bijvoorbeeld dienst doen om de positieve (bv. Blauw) en negatieve (bv. Rood) pool van een batterij op aan te sluiten. Op deze manier kan je alle elementen die van stroom voorzien moeten worden voeden met slechts 1 batterij. De verticale rijen zijn telkens per 5 doorverbonden. Merk op dat er in het midden geen doorverbinding is. De verticale rijen kan je gebruiken om sensoren, weerstanden, microcontrollers enzovoort in te pluggen. De vierkantjes komen overeen met de gaatjes waarin male pin headers van jumper wires of pinnen van sensoren gestoken kunnen worden.



## Jumper Wire Kit



*Figuur 33: Jumper Wire Kit - groot*

Om de nodige verbindingen te maken tussen de sensoren, actuatoren en de ESP32, maakte ik gebruik van “jumper wires”. Dit zijn kabeltjes die in verschillende, relatief korte lengten verkrijgbaar zijn, variërend van enkele centimeters tot 30 centimeter of meer. Van jumper wires zijn er 3 verschillende typen op de markt:

### 1) Male-to-Male

Deze soort bevat aan weerszijden een uitstekend pinnetje. Vergelijk het met een kabel waar aan elke kant een stekker hangt.

### 2) Male-to-Female

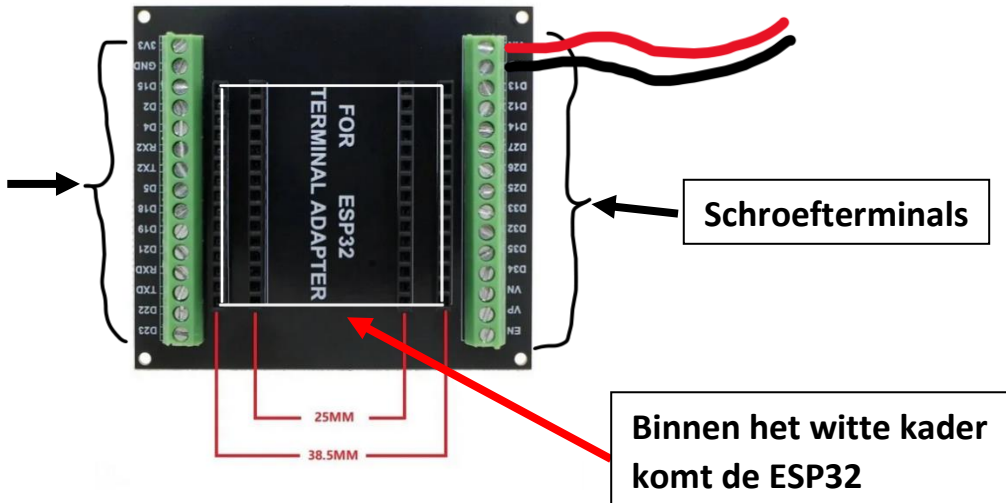
Deze soort bevat aan de ene zijde een uitstekend pinnetje, aan de andere zijde een opening waar een pinnetje ingeplugd kan worden. Denk aan een standaard verlengkabel met een stekker aan de ene en een stopcontact aan de andere kant.

### 3) Female-to-Female

Deze soort bevat aan weerszijden een opening waar een pinnetje ingeplugd kan worden. Analoog aan de 2 voorgaande typen zou dit een kabel zijn met aan beide kanten een stopcontact.

Deze kabeltjes kan je goedkoop in grote hoeveelheden en in alle mogelijke kleuren bestellen. De verschillende kleuren zijn vooral handig als je een onderscheid wilt bewaren tussen de kabeltjes die van A naar B lopen en om consistentie te hebben wat betreft zaken als bijvoorbeeld: Rood is positief (+ pool) en zwart is negatief (- pool). Daarnaast kan je ze ook veelvuldig hergebruiken gedurende 1 of zelfs meerdere projecten.

## Breakout Board ESP32



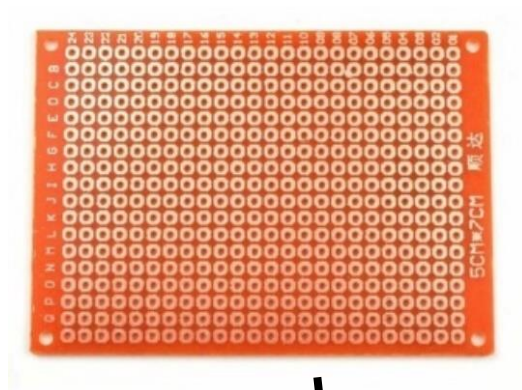
Figuur 34: Breakout board ESP32 - groot

Om zeker te zijn dat de verbindingen tussen de ESP32 en de andere componenten vast bleven zitten, maar toch ook wat flexibiliteit te behouden voor het geval er nog een foutje gebeurd was bij het bekabelen, koos ik ervoor om een breakout board te gebruiken. Voor ik aan dit project begon wist ik nog niet af van het bestaan van deze component. Een definitief systeem dat verbindingen bevatte die in een breadboard staken, was niet betrouwbaar genoeg. De kabeltjes komen hieruit namelijk nogal makkelijk los, waardoor dit enkel wordt aangeraden voor testdoeleinden.

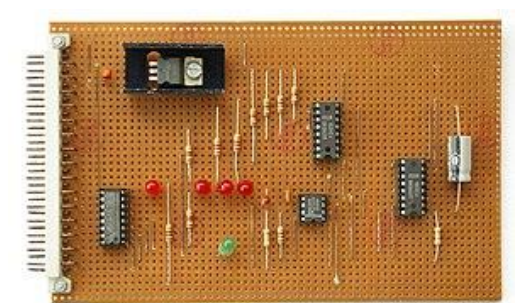
Gesoldeerde verbindingen waren ook een optie, al is het hier wel de bedoeling om achteraf niet te veel verbindingen meer los te solderen of te wijzigen. Dit kan ervoor zorgen dat je de sensor waaraan je een kabel soldeert beschadigt, of de volgende verbinding minder stevig is dan de oorspronkelijke. Wat solderen betreft, is het mogelijk om op de component zelf te solderen of – als je bijvoorbeeld nog weerstanden, diodes, condensatoren, ... tussen de ESP32 en de andere component dient te plaatsen – gebruik te maken van een geperforeerde PCB, korter gezegd een perfboard.

Een goede tussenoplossing voor het systeem dat ik ontwikkelde was een Breakout board, speciaal gemaakt voor de ESP32 Devkit. De bedoeling is dat je de ESP32 Devkit in de daarvoor voorziene Female headers plugt. Ik heb op de foto een witte kader getekend om duidelijk te maken waar de ESP32 ongeveer dient te komen. Deze headers zijn op hun beurt doorverbonden met schroefterminals waar je draadjes kan inschuiven en vastschroeven om voor een permanente verbinding te zorgen.

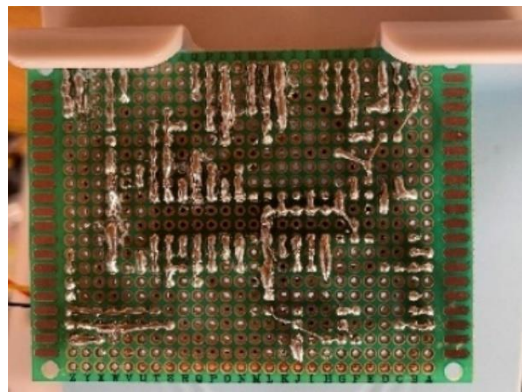
## Perfboard



Figuur 35: Perfboard leeg



Figuur 36: Perfboard met componenten



Figuur 37: Perfboard onderzijde

Dit lijkt qua ontwerp wat op een breadboard, met dat verschil dat de kabeltjes er niet gewoon ingeplugd worden, maar worden vastgesoldeerd met soldeertin. Dit zorgt ervoor dat het dienst kan doen als permanente oplossing, zeker als je een systeem aan het bouwen bent dat veel weerstanden, diodes, condensatoren, en andere 'kleinere' elektronica vereist. Je vertrekt van een leeg perfboard, waar ideaal gezien aan minstens één zijde zogenaamde 'soldeereilandjes' aangebracht zijn. Soldeereilandjes zijn de metalen ringetjes die elk gaatje omringen. Deze zorgen ervoor dat het soldeertin zich makkelijker hecht aan het gaatje waar je een component in wil vast solderen.

Hierop plaats je alle nodige componenten voor je opstelling. Je zorgt ervoor dat alle 'pootjes', of metalen uiteinden van de componenten zo veel mogelijk door de gaatjes heen komen en de componenten zo dicht mogelijk tegen het perfboard-oppervlak aanzitten. Je plooit de pootjes best opzij zodat de componenten stevig blijven zitten. Vervolgens draai je je perfboard om en kan je beginnen solderen.

Zorg ervoor dat elk gaatje mooi opgevuld wordt met soldeertin (geen luchtholtes) en je niet morst op andere gaatjes. Je wil namelijk geen kortsluiting maken door per ongeluk eilandjes met elkaar te verbinden. Eens een component volledig is vastgesoldeerd kan je het resterend, uitstekend stuk van het 'pootje' afknippen met een kniptang. Je kan ook doelbewust gaatjes met elkaar doorverbinden als dat nodig is natuurlijk. Een tip die ik van Ruben kreeg was om de afgeknipte pootjes te gebruiken om mooie doorverbindingen te krijgen als je meer dan 2 gaatjes met elkaar moet verbinden. Dit kan je goed zien op de uiterst rechtse foto. Kijk tot slot na of alles goed vastzit en er geen gaatjes met elkaar in contact komen met elkaar wanneer dat niet de bedoeling is. Plaats de componenten om deze reden best niet vlak tegen elkaar maar voorzie een beetje speling om dit probleem te vermijden.

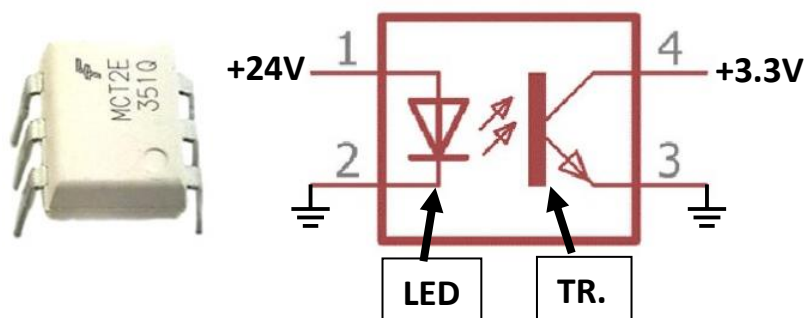
## Optocoupler (2-kanaals)



Figuur 38: Optocoupler - groot

Om te detecteren of één van de lasers van de productielijn actief was en dus met een bewerking bezig was, dienden we een signaal te gaan meten in de laserkast. Hier bevond zich een zogenaamd M4 controller. Aan dit bordje bevonden zich heel wat aansluitpinnen, waarvan er sommige reeds in gebruik waren ter aansturing van de laser. 1 van deze pinnen kon ook gebruikt worden om een spanningsniveau te meten. Wanneer de laser inactief was, bedroeg deze spanning 0 volt, wanneer deze actief was 24 volt. Probleem was wel dat we dit signaal niet zomaar rechtstreeks konden verbinden aan de ESP32. Deze microcontroller verdraagt slechts aansluitingen van maximaal 3.3 volt. Gelukkig bestaat er iets zoals een optocoupler.

Dit stukje elektronica kan 2 stroomkringen fysiek van elkaar scheiden en toch doorgeven wanneer er stroom vloeit door 1 van de twee stroomkringen. Hieronder leg ik uit hoe een optocoupler werkt. Ik koos ervoor een optocoupler met 2 kanalen te kiezen voor het geval er 1 van de 2 defect zou zijn of zou stukgaan na verloop van tijd.



Figuur 39: Werking Optocoupler

Aan de linkerkant zien we aan de bovenkant een spanningsniveau van 24 Volt. Dit is de kant van de laser, hier gaan we een signaal meten dat de rest van het systeem nodig heeft als trigger om de afzuiging op te zetten. Zoals je kan zien in de schematische voorstelling bevindt er zich een LED. Wanneer er spanning wordt aangelegd aan de linkerkant zal er stroom door de LED lopen en zal deze dus oplichten. Aan de rechterkant bevindt er zich een lichtgevoelige transistor (TR.) die enkel en alleen stroom zal geleiden als er licht op valt. Er kan geen stroom rechtstreeks van de ledzijde naar de transistorzijde vloeien omdat er een elektrisch isolerend doorzichtig materiaal tussen zit (kan ook gewoon lucht zijn maar meestal is dit een soort kunststof). Wanneer de lichtgevoelige transistor licht opvangt, begint er stroom te vloeien aan de rechterkant. Aangezien dit een circuit is dat een maximale spanning heeft van 3.3 volt, is dit geen probleem voor onze ESP32. Op deze manier kan het signaal van de laserzijde worden doorgegeven aan de ESP32 zonder fysiek contact te maken en weet de ESP32 dus wanneer de laser actief is.

## Relaisbord (4-kanaals & 1-kanaals)

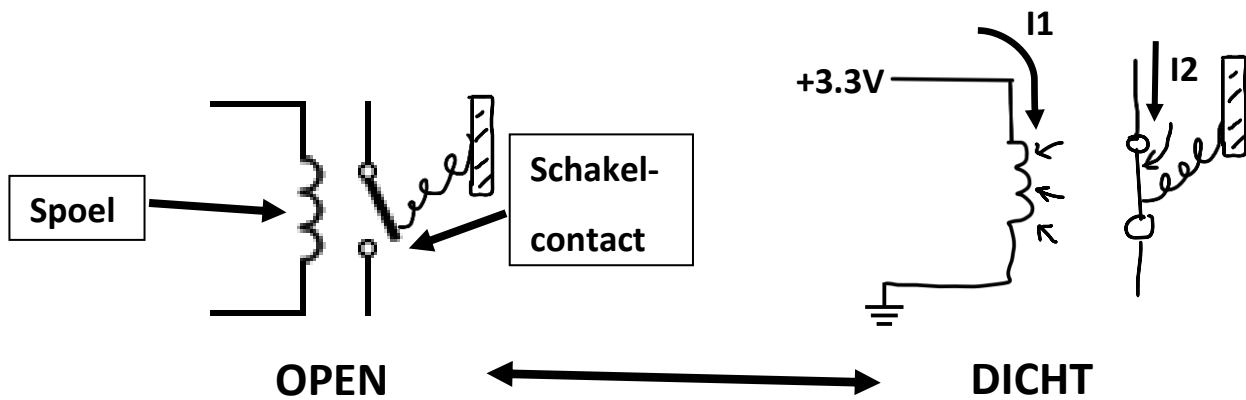


Figuur 40: Relaisbord 4-kanaals - groot

Om de lampen, de zoemer en de contactor aan- of uit te zetten had ik een soort schakelaar nodig. Niet het type schakelaar waarmee je de verlichting thuis aanzet, wel een schakelaar die een kleine stroom en een laag spanningsniveau vereist en die je dus met de ESP32-microcontroller kan bedienen. Hiervoor opteerde ik voor 2 relaisbordjes.

1 bordje met 4 kanalen om de rode, gele en groene lamp + de zoemer te schakelen.

1 bordje met 1 kanaal om de contactor te schakelen. Over de contactor kan u hieronder meer info verschaffen.



Figuur 41: Relais werking

Een relais werkt op deze manier:

In de blauwe kubusjes die je hierboven kon zien zit een spoel en een schakelcontact. Een spoel ziet er wat uit als een staafje waar heel veel koperdraad is rondgewikkeld. Het schakelcontact ziet er wat uit als een metalen deur die altijd openstaat. Vaak is deze verbonden aan een veer zodat ze altijd blijft openstaan. Wat gebeurt er nu als je spanning (+ 3.3V boven en 0V beneden) aanbrengt op de spoel? Zoals je kan zien begint er dan een stroom  $I_1$  van boven naar beneden door de spoel te lopen. Deze stroom zorgt ervoor dat er een magnetisch veld wordt opgewekt in de spoel (3 pijltjes naast de spoel). De schakelaar (de metalen 'deur') die open bleef staan door de veer, wordt nu zo hard aangetrokken door de spoel dat de veer de deur niet meer kan tegenhouden. Hierdoor sluit de schakelaar (slaat de deur dicht) en kan er een stroom  $I_2$  in de 2<sup>e</sup> stroomkring (2<sup>e</sup> kamer) van boven naar beneden stromen. Deze kan op haar beurt bijvoorbeeld één van de lampen aanzetten. Wanneer de spanning wegvalt, valt het magnetisch veld van de spoel weg en word de schakelaar (deur) niet meer dichtgetrokken. De veer trekt de schakelaar weer open en de lamp die aanstond wordt weer uitgezet omdat er geen stroom meer vloeit.

Deze relaisbordjes vereisen een voedingsspanning van 5V – die los staat van het schakelcircuit. Deze zal geleverd worden door een externe voeding (die we ook al nodig hebben ter voeding van de ESP32). Ze vereisen ook een kleine schakelstroom, die geleverd kan worden door de ESP32, om het schakelcontact te openen of te sluiten. De relaisbordjes zijn dus aan de ingang aangesloten op een schakelcircuit van 3.3V, samen met de ESP32. Aan de uitgang kan een stroomkring aangelegd worden die een hoger spanningsniveau heeft, tot maximaal 30 Volt. Dit is voldoende om zowel de lampen, de zoemer als de contactor te schakelen. Zij vereisen allen een spanningsniveau van 24 Volt. Op deze manier kan de ESP32 dus zaken aan- of uitzetten die een spanningsniveau nodig hebben dat hoger ligt dan 3.3 Volt.

## Schakelaar (3 inputs)



*Figuur 42: Schakelaar 3 inputs - groot*

Het door mij ontwikkelde systeem, AFCS, kan zich in 3 verschillende toestanden bevinden (4 als je de situatie waarbij het systeem niet is ingeplugd meerekent). Ik som ze hier nog kort even op ter opfrissing:

- Uit-modus (0 op de schakelaar): Zet de blower handmatig uit.
- Manuele modus (I op de schakelaar): Zet de blower handmatig aan.
- Auto-modus (II op de schakelaar): Laat het systeem de blower aan- of uitzetten in functie van de laseractiviteit.

Om deze 3 modi op een eenvoudige wijze te kunnen instellen, ging ik op zoek naar een schakelaar met 3 standen. Tijdens mijn zoektocht kwam ik ook tuimelschakelaars tegen, die soms nog meer mogelijke standen hadden, maar dat leek me minder duidelijk dan degene die ik gebruikt heb.



*Figuur 43: Tuimelschakelaar*



## Signaallamp + zoemer





Figuur 44: Signaallamp + zoemer - groot

Ter signalisatie koos ik voor een signaallamp met 3 verschillende kleuren, waar ook een zoemer ingebouwd zat. Reden hiervoor was om verschillende statussen van het systeem te kunnen weergeven en te kunnen waarschuwen wanneer er een toestandswijziging op til was. Dit zijn de opgenomen statussen en hun betekenis.

<ul style="list-style-type: none"> <li>• Rode lamp: uit</li> <li>• Gele lamp: uit</li> <li>• Groene lamp: uit</li> <li>• Zoemer: uit</li> </ul>		<p>Het systeem is uitgeschakeld, wordt niet voorzien van stroom.</p>
<ul style="list-style-type: none"> <li>• Rode lamp: aan</li> <li>• Gele lamp: uit</li> <li>• Groene lamp: uit</li> <li>• Zoemer: uit</li> </ul>		<p>Het systeem wordt voorzien van stroom en bevindt zich in status 0 (uit-modus). Rode lamp dient enkel ter info of het systeem ingeplugd is of niet.</p>
<ul style="list-style-type: none"> <li>• Rode lamp: aan</li> <li>• Gele lamp: uit</li> <li>• Groene lamp: aan</li> <li>• Zoemer: uit</li> </ul>		<p>Het systeem wordt voorzien van stroom en bevindt zich in Auto-modus (II). De blower staat nog niet aan maar zal automatisch opgezet worden bij activiteit van minstens 1 van de aangesloten lasers.</p>
<ul style="list-style-type: none"> <li>• Rode lamp: pinkt (3 seconden)</li> <li>• rGele lamp: uit</li> <li>• Groene lamp: uit</li> <li>• Zoemer: aan (3 seconden)</li> </ul>		<p>Statuswijziging blower. Wanneer de blower uitstaat en wordt aangezet of de blower aan staat en zal worden uitgezet, zie en hoor je deze indicatie als waarschuwing voor de werknemers in de buurt.</p>



<ul style="list-style-type: none"> <li>• Rode lamp: uit</li> <li>• Gele lamp: uit</li> <li>• Groene lamp: aan</li> <li>• Zoemer: uit</li> </ul> 	De blower is operationeel. Het systeem kan zich in zowel Manuele modus (I) als Auto-modus (II) bevinden.
<ul style="list-style-type: none"> <li>• Rode lamp: n.v.t.</li> <li>• Gele lamp: pinkt 3x per 10 sec</li> <li>• Groene lamp: n.v.t.</li> <li>• Zoemer: uit</li> </ul> 	Filterzak is vol en dient vervangen te worden. De gele lamp dient enkel ter indicatie voor de status van de filterzak en staat los van de status van de rest van het systeem. Ook al staat het systeem ingesteld op Uit-modus (0), de status van de filterzakken wordt nog steeds gemonitord.

## Contactor (Siemens 3RT1015-1BB41)



Figuur 45: Contactor - groot

Zoals ik hierboven al had uitgelegd, bevat het systeem 2 verschillende soorten relaisbordjes. Beiden dienen als schakelaars die door de ESP32 bediend kunnen worden. Voor de lampen en de zoemer was er niets extra nodig, zij werkten namelijk op een spanning van 24 Volt (Gelijkspanning). De blower daarentegen is een toestel dat werkt op 230 Volt (Wisselspanning). Om deze met de ESP32 te kunnen in- of uitschakelen, dienden we nog een component te zoeken die de overbrugging kon maken van 24 Volt naar 230 Volt.

Hiervoor ben ik eens in de opslagruimte van CERcuits gaan kijken. Hier stond allerlei materiaal, waaronder ook defecte elektronica, lasers en dergelijke meer. Hier vond ik wat ik zocht; een contactor van Siemens. Een contactor verschilt qua werkingsprincipe eigenlijk niet zo hard van de relais, met dat verschil dat het het “grotere broertje” ervan zou kunnen zijn. De contactor kan namelijk wel gebruikt worden voor stroomkringen met een spanning van 230 Volt. Een bijkomend voordeel van deze vondst was dat we ze niet meer hoefden aan te kopen en dus een ruwe 30 euro uitspaarden per systeem.

De contactor vereist een spanning van 24 Volt om in staat te zijn te kunnen schakelen. Deze spanning wordt aan 1 van de 4 kanalen aangelegd. De andere kanalen kunnen dienen om kringen tot maximaal 400V te schakelen. Alle aan te sluiten kabels dienden afgewerkt te worden met adereindhulzen en tot slot te worden vastgemaakt in de schroefterminals. Hieronder verduidelijk ik nog een beetje hoe de blower nu uiteindelijk kan worden opgezet met de ESP32.

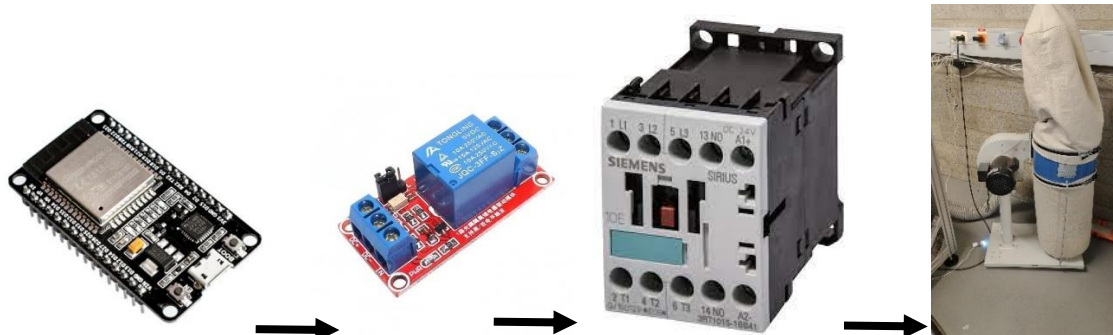


Fig. Aansluitvolgorde contactor

## Voeding 24V + 5V

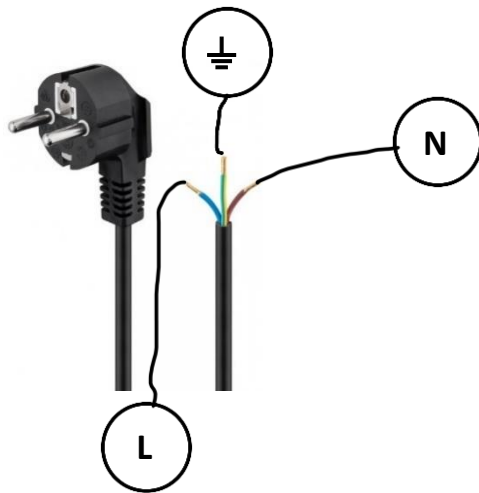


Figuur 46: Voeding - groot

Om het gehele systeem van stroom te voorzien had ik een bron nodig die zowel een spanning van 24 Volt (+V2 op de voeding) kon leveren, als 5 Volt (+V1 op de voeding). Om de contactor en de signaallamp met ingebouwde zoemer te laten functioneren moest ik deze aansluiten op 24 Volt. De 5V diende dan weer om de ESP32 plus de relaisbordjes (4-kanaals & 1-kanaals) te voorzien van voeding. Tijdens mijn zoektocht was ik eerst van plan om zelf een transformator te bouwen die 230 Volt wisselspanning kon omvormen naar 24 & 5 Volt gelijkspanning. Ik wou dit graag op deze manier oplossen omdat het een uitdaging vormde en het sowieso goedkoper zou zijn dan de voedingen die ik op dat moment al online had gevonden.

Na een gesprek met Giel, één van mijn collega's, had ik het idee om standaard adapters te gebruiken. De adapters die worden gebruikt om bijvoorbeeld smartphones op te laden leveren bijvoorbeeld 5V, de laptopladers dan weer vaak 24V. Ik dacht dat dit wel een goed idee kon zijn omdat ik hoogstwaarschijnlijk wel nog oude laptopladers en gsmladers zou vinden in de opslagruimte van CERcuits. Probleem was wel dat we dan 2 stopcontacten nodig hadden om het hele systeem te kunnen gebruiken. Het is pas nadat Ruben, mijn stagementor, me terug in de richting had gewezen van de industriële voedingen, dat ik na een tijdje zoeken een voeding vond van die beide spanningsniveau's leverde. Het enige wat ik nog nodig had was een voedingskabel om aan te sluiten op de aarding ( $\perp$ ), de lijningang (L) en de neuteringang (N) van de spanningsbron.

## Voedingskabel 230V



Figuur 47: Voedingskabel 230V - groot

Om de voeding te kunnen gebruiken diende ik enkel nog een drie-aderige voedingskabel te zoeken die dik genoeg was om deze van stroom te voorzien. Drie-aderig wil zeggen dat er in de zwarte plastic behuizing 3 afzonderlijke kabels zitten. In dit geval bevat deze kabel een blauwe kabel (lijningang of L), een bruine kabel (neuter of N) en een aarding ( $\perp$ ). Na overleg met mijn stagementor kwam ik erachter dat in de meeste gevallen een kabelkerndikte van 0.5 vierkante millimeter volstond voor de meeste toepassingen. Toen ik op de website van Tinytronics, waar ik al heel wat andere elektronica had besteld, een standaard voedingskabel zag met kerndikte van 0.75 vierkante millimeter, bestelde ik ze meteen. Ik had ook kunnen opteren voor kabels die hadden gediend om oude computerschermen of iets in die aard te voeden, maar daar had ik pas later aan gedacht. Qua veiligheid leek het me achteraf gezien ook wel het beste om nieuwe kabels te bestellen.

Om de 230V die via deze voedingskabel binnenkwam ook te kunnen gebruiken voor de contactor (en niet enkel voor de spanningsbron), diende ik deze af te takken en met behulp van zogenaamde Wago-klemmen aan elkaar te verbinden. Een aansluitschema hierover vindt u terug onder "Schema's" wat lager in dit document.

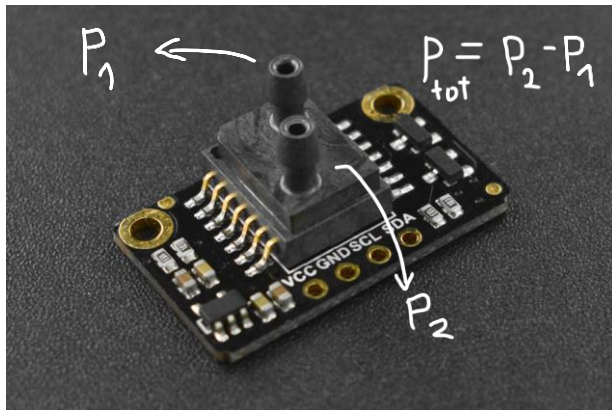
## WAGO Verbindingsklem



Toen ik op zoek was naar manieren om kabels aan elkaar te verbinden op een betrouwbare manier, zonder te moeten solderen, kwam ik op een gegeven moment uit op verbindingsklemmen van het merk WAGO. Het principe hier is heel simpel: de door mij gebruikte wago-klemmen hebben 3 aansluitpunten, die onderling allemaal doorverbonden zijn. Vooraleer je een kabel kan vastmaken in de klem moet je het oranje klepje omhoog klikken. Vervolgens schuif je de (gestripte) kabels in 1 van de 3 openingen waarvan het oranje klepje omhoog staat. Om deze kabel vast te maken klik je het oranje klepje weer naar beneden. Herhaal dit tot 3 keer toe, alle kabels die nu in de Wago-klem zitten zijn nu aan elkaar verbonden.

Oorspronkelijk was het plan om op deze manier heel wat zaken aan elkaar te verbinden. Zo was ik van plan om alle 5V-kabels door te verbinden, de Ground-kabels, de weerstanden, ... . Dit bleek al snel heel omslachtig en toen mijn stagebegeleider dit zag, adviseerde hij mij vriendelijk maar onder lichte dwang om toch eens te kijken naar perfboardjes of iets in die aard. Gelukkig waren de bestelde Wago-klemmen nog bruikbaar voor de doorverbinding van de voedingskabel naar de blower en de voeding zelf. Ook konden ze de losse contacten in de blower zelf (waar de noodstop had gezeten) aan elkaar verbinden. Door deze uit de blower te halen werd het makkelijker om de rest van het systeem aan te sluiten.

## Differentiële druksensor (LWLP5000-5XD)



Figuur 48: Druksensor - groot

Één van de systeemvereisten was het aangeven van de status van de filterzakken. Wanneer deze vol zitten met stofdeeltjes dienen deze namelijk vervangen en gewassen te worden. Al snel werd duidelijk dat de beste manier om dit te detecteren waarschijnlijk via een drukmeting zou zijn. Dan rees de vraag: meten we enkel de druk in de filterzak of meten we een differentiële druk? Een differentiële drukmeting is een meting waarbij de druk gemeten wordt op 2 verschillende meetpunten ( $p_1$  en  $p_2$ ), waarna de 'gemeten druk' ( $P_{tot}$ ) het verschil bedraagt tussen deze 2 punten. Dat leek mij ideaal aangezien we een consistentere drukmeting zouden hebben als we zowel de druk buiten de filterzak als binnen zouden meten.

Eens dat beslist was kon ik mijn zoekopdracht verfijnen. Het bleek niet zo eenvoudig om een sensor te vinden die niet waanzinnig duur was én dan ook nog eens eenvoudig aan te sluiten was op een ESP32. Daarnaast was er ook nog het volgende probleem: Hoe groot wordt de druk in een volle filterzak geschat? Hiervoor vond ik gelukkig ook een oplossing.

Hiervoor wachtte ik tot er 1 van de filterzakken volledig vol zat en de druk in de filterzak maximaal was. Ik schakelde de blower even uit. Vervolgens nam ik een rubberen slang die ik deels vulde met water. De fysica vertelt ons dat, als je deze buis dan in een U-vorm houdt en het water even de tijd geeft om tot stilstand te komen, het water aan beide zijden even hoog komt. Het volstaat om dan aan beide zijden te markeren met een stift tot waar het waterniveau komt. Vervolgens stak ik 1 van de zijden van de slang in de filterzak (zonder water te morsen) en schakelde ik de blower weer in. Ik wachtte tot de filterzak weer volledig opgeblazen was vooraleer ik ging kijken hoeveel het water, aan de zijde die niet in de filterzak stak, gestegen was. Opnieuw markeerde ik het waterniveau aan beide zijden met de stift. Ik schakelde de blower weer uit en haalde de slang eruit. Ik goot het water eruit en mat de afstand tussen beide markeringen. Om nu de druk in de filterzak te berekenen maakte ik gebruik van de Wet van Pascal.

Deze wet vertelt ons:

*“Een druk die wordt uitgeoefend op een vloeistof die zich in een geheel gevuld en gesloten vat bevindt, zal zich onverminderd in alle richtingen voortplanten.”*

In formulevorm wordt dat:

$$p = p_0 + \rho \cdot g \cdot h$$

- $p$  is de hydrostatische druk in een willekeurig punt in een vloeistof op een diepte  $h$  onder het vloeistofoppervlak.
- $p_0$  is de druk aan het vloeistofoppervlak ( $h = 0$ ). Voor een vloeistof in de lucht opgesteld, is  $p_0$  de luchtdruk (1013 hPa/1,013 bar).
- $\rho$  is de dichtheid van de beschouwde vloeistof. Voor water is dit ongeveer 1000 kg/m<sup>3</sup>.
- $g$  is de zwaartekrachtversnelling. Deze bedraagt 9.81 m/s<sup>2</sup>.

Aangezien we hier een differentiële drukmeting doen en dus aan beide zijden een druk  $p_0$  meten, mogen we deze bij onze berekening verwaarlozen en dus weglaten. Na invulling van mijn gegevens kwamen we hierop uit.

$$p = 1000 \text{ kg/m}^3 \cdot 9.81 \text{ m/s}^2 \cdot 0.02 \text{ m} = 196.2 \text{ Pascal}$$

Ik moest dus op zoek gaan naar een sensor die waarden accuraat kon weergeven van 0 tot 196 Pascal. Om wat speling te hebben nemen we dan best een sensor die een iets ruimer bereik heeft. Eens ik dit wist vond ik al vrij snel dat de sensor van DFRobot de goede keuze zou zijn. Deze sensor had namelijk een bereik van -500 tot 500 Pascal en was dus ideaal voor deze opstelling.

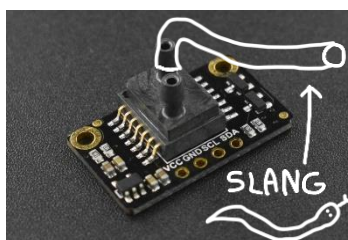
[https://nl.wikipedia.org/wiki/Wet\\_van\\_Pascal](https://nl.wikipedia.org/wiki/Wet_van_Pascal)

<https://nl.wikipedia.org/wiki/Luchtdruk>

<https://nl.wikipedia.org/wiki/Water>

<https://www.dfrobot.com/product-2096.html>

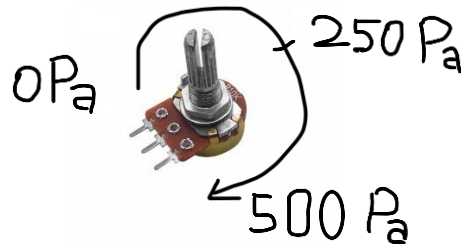
## Rubberen slang



Om de drukmeting in de filterzak zelf uit te voeren kon ik ervoor kiezen om de sensor aan de binnenkant van de houder voor de filterzak te plaatsen, aan de buitenkant van deze houder, of vlakbij de ESP32. De laatste optie is hier de beste, om verschillende redenen. Als ik de sensor aan de binnenkant van de filterzak zou plaatsen, bestaat er een kans dat de sensor na een tijdje niet meer degelijk werkt, doordat er zich kleine stofdeeltjes genesteld zouden hebben binnenin de sensor. Hierdoor zou het kunnen dat de sensor geen accurate metingen meer toont en dus vervangen dient te worden. Daarnaast is het altijd beter om de afstand tussen een sensor en de verwerker van de gegevens (in dit geval de ESP32) zo klein mogelijk te houden. Hoe langer de draden, hoe groter de kans dat er onderweg elektromagnetische storing optreedt, waardoor er fouten kunnen sluipen in de meetgegevens die worden ontvangen aan de zijde van de ESP32.

Tot slot had Ruben ook gevraagd om het systeem zo compact mogelijk te houden, en dus was het vanzelfsprekend dat ik de sensor zo dicht mogelijk bij de ESP32 wou houden. Omdat er daardoor een afstand van ongeveer een meter ontstond tussen de ESP32 en de blower, en we nog steeds een drukmeting dienden te doen aan de binnenzijde van de filterzak, hadden we een dunne rubberen slang nodig. Deze verbond dan de binnenkant van de filterzak met 1 van de pijpjes die op de druksensor stonden. Het andere pijpje liet ik onaangeroerd zodat het de luchtdruk buiten de filterzak kon meten.

## Potentiometer 10k



Een potentiometer is niets meer dan een regelbare weerstand. De weerstand van dit specifiek type is er eentje die varieert van 0 Ohm tot 10.000 Ohm. Regelen doe je door te draaien aan het staafje dat aan de potentiometer hangt. De potentiometer ken je waarschijnlijk niet als regelbare weerstand, maar wel als bijvoorbeeld een volumeknop van een radio. De geregelde weerstand kan dus gekoppeld worden aan zaken die er geen rechtstreeks verband mee hebben. In mijn opstelling maakte ik gebruik van een potentiometer om de grenswaarde van de druksensor handmatig in te stellen. Niet elke blower blaast exact even hard, misschien zit er ook een klein verschil op de metingen van de verschillende druksensoren, ... . Het was dus wenselijk om de grenswaarde (de waarde vanaf wanneer het systeem zou aangeven dat de filterzak vol was) voor elk systeem afzonderlijk te kunnen regelen. Door programmatorisch aan te geven dat 0 Ohm overeen kwam met 0 Pascal en 10.000 Ohm overeen kwam met 500 Pascal, werd dit gerealiseerd.

## Draaiknop



Om het geheel iets mooier af te werken, en in één oogopslag te zien of de potentiometer open- of dichtgedraaid was, besloot ik om een draaiknop te bestellen. Ik koos voor een zwart met rode draaiknop omdat rood mijn favoriete kleur is.





Om het geheel mooi te kunnen afwerken, alles compact bij elkaar te houden en tegelijkertijd af te schermen tegen externe factoren zoals water of statische elektriciteit, besloot ik om een aftakdoos te gebruiken. Ze bevatten een afneembaar deksel dat je zelfs als 'deurtje' kan openzwaaien als je in slechts 1 zijde van de doos de bijgeleverde plastic schroeven voor de helft inschroeft. Dit heb ik proberen verduidelijken met de pijltjes en de schroeven hierboven. Deze dozen zijn vervaardigd uit harde plastic van enkele millimeters dik, wat het wel nog steeds mogelijk maakt om vlot gaten te boren of te zagen in het paneel.

Ik zocht naar een aftakdoos die groot genoeg was om alle elektronische componenten er mooi in te laten passen en maakte gaten op de plekken waar er aansluitingen voor kabels of knoppen moesten komen. De signaallamp met zoemer plaatste ik uiteraard niet in deze doos, omdat deze goed zichtbaar moest zijn vanop afstand. Eens alle componenten verbonden waren en op hun plek zaten, boorde ik gaatjes door het paneel en bevestigde ik alles stevig met boutjes en moeren.

## DIN-5 Panel mount

Om een betrouwbare, eenvoudig vervangbare verbinding te hebben tussen AFCS – dat zich in de aftakdoos bevond – en de signaallamp koos ik ervoor om een DIN 5 kabel te gebruiken. Dit is een kabel die vijfaderig is. De 5 aders die door deze kabel lopen zijn de volgende:

- 1) Rode lamp
- 2) Gele lamp
- 3) Groene lamp
- 4) Zoemer
- 5) Ground

Het grote voordeel van dit type kabel is dat stevig vastzitten, maar niet permanent. Stel dat de kabel door welke reden dan ook beschadigd zou geraken, hoef je niet alles opnieuw vast te solderen. Blijkt dat de afstand tussen AFCS en de lamp toch groter moet worden? Het volstaat dan om in beide gevallen een nieuwe (langere) DIN-5 kabel te kopen. Ik zag tijdens mijn stage ook dat voor veel van de aansluitingen bij de lasers (voeding, X, Y & Z-as) ook dit type kabel gebruikt werd. Hierdoor was de DIN kabel een logische keuze.



## Wartels



*Figuur 49: Wartel - groot*

Ook dit onderdeel hoorde bij de afwerking van mijn systeem. Om te vermijden dat de voedingskabel zou kunnen worden losgetrokken uit het systeem, bestelde ik Wartels met zogenaamde 'trekontlasting'. Hierdoor kwam de voedingskabel voor zowel de voeding als de contactor. Eens deze erdoor zat restte mij enkel nog de Wartel aan te draaien zodat de kabel erin werd vastgeklemd.

## Dubbele jack 3.5mm (5m & 10m)



*Figuur 50: Dubbele Jack - groot*

Deze kabels gebruikte ik om een correcte signaaloverdracht tussen de optocoupler (die in de laser zat) en AFCS te maken. De hoofdreden hiervoor was het feit dat de draadjes in een Dubbele Jack-kabel twisted zijn, waardoor ze minder gevoelig zijn aan elektromagnetische interferentie. Reden waarom ik een vervangbare kabel gebruik is ook analoog aan de uitleg over de DIN 5 kabel.

## Jack 3.5mm Panel mount



Figuur 51: Jack Panel mount - groot

Deze heb ik vastgemaakt in de aftakdoos, zodat de Aux-kabels in AFCS geplugd konden worden en dus verbinding konden maken met de ESP32.

## Stekkerhulskit + krimptang



Figuur 52: Stekkerhulskit - groot

Ook wel stekkerschoentjes genoemd. Deze kleine onderdeeljes dien je te bevestigen aan het uiteinde van de kabels die je over een bepaalde stekker wil schuiven. Ik heb deze gebruikt om een stevige verbinding tussen de ESP32 en de schakelaar te garanderen.

## Adereindhulskit + krimptang

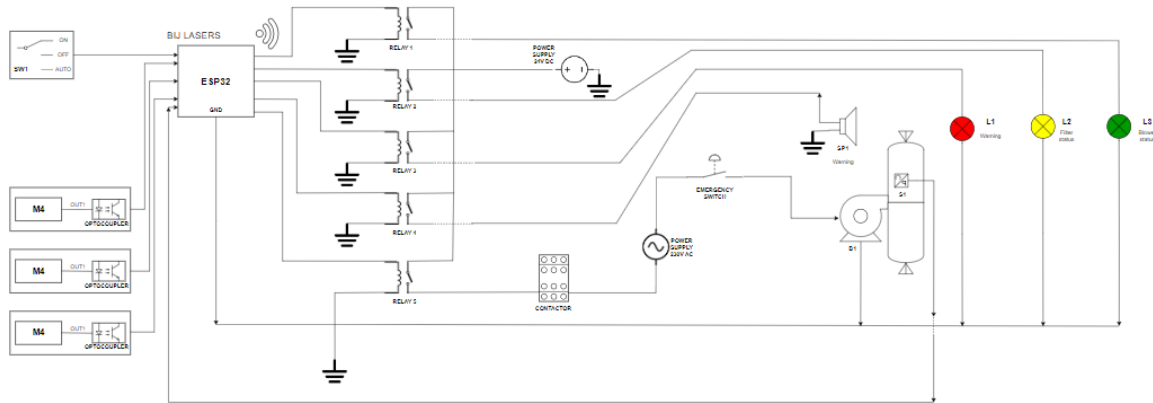


Figuur 53: Adereindhulskit - groot

Ook wel kabelschoentjes genoemd. Bevestig je op gelijkaardige wijze aan het uiteinde van een draadje om deze vervolgens in een schroefterminal te kunnen draaien zonder dat deze loskomt.

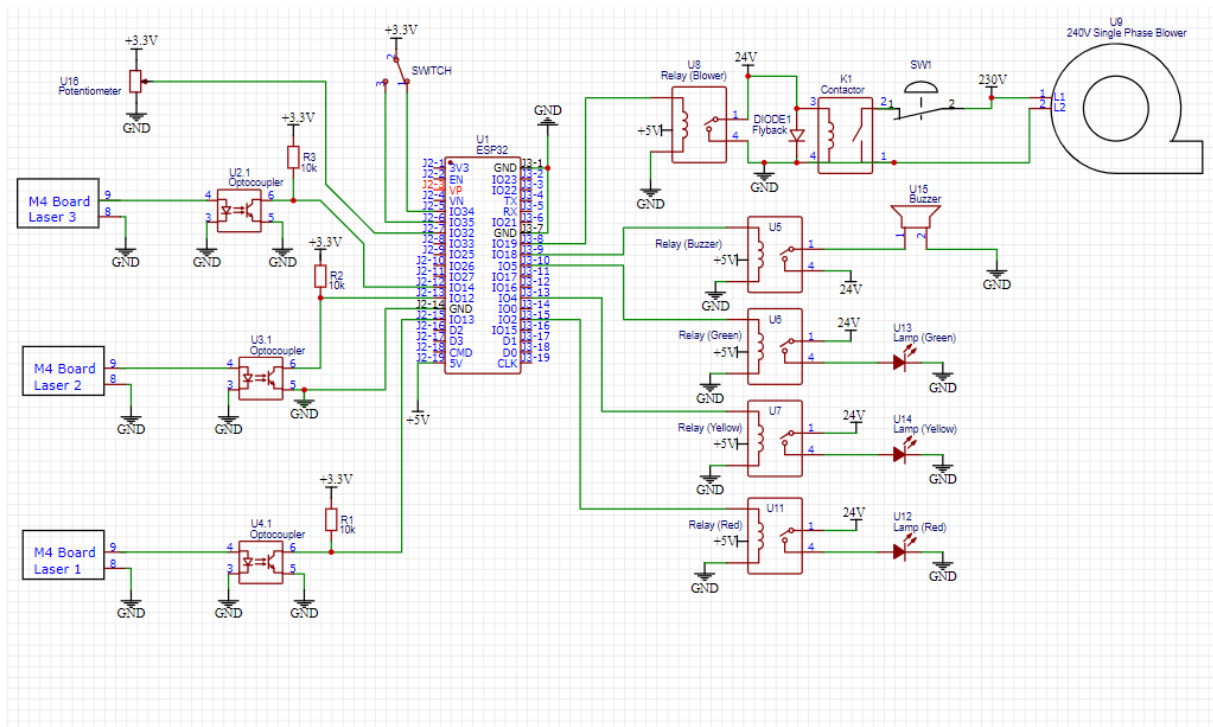
# Schema's

## Globaal schema



Figuur 54: AFCS Schematisch

## Elektrisch schema

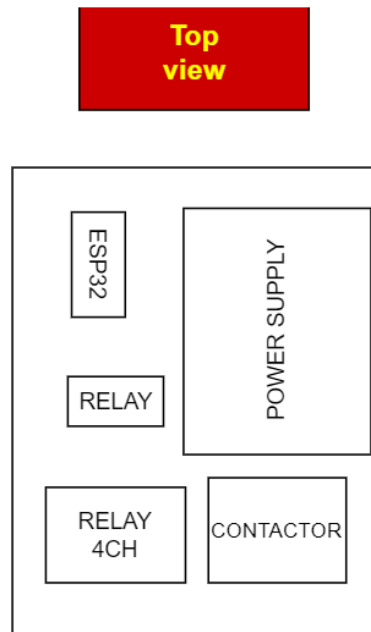


Figuur 55: Elektrisch Schema AFCS

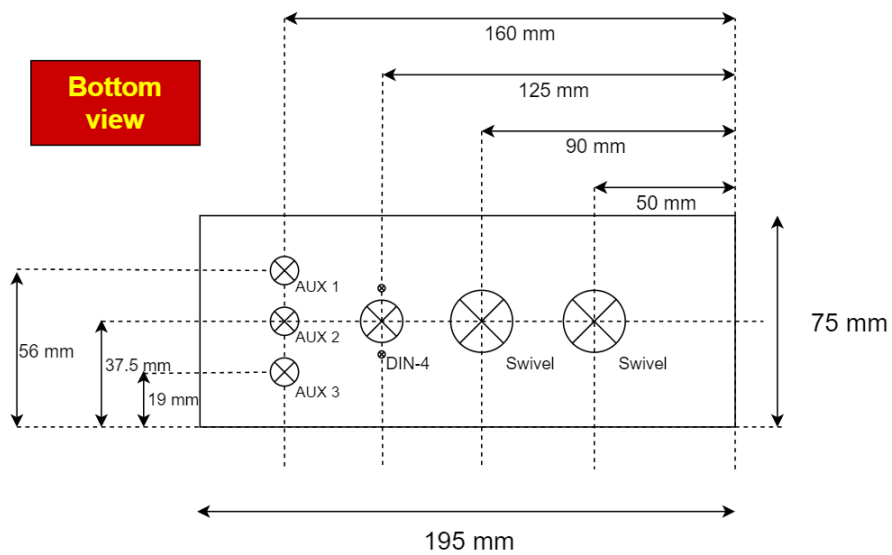
## Aansluittabel ESP32

Terminal op ESP32 (Breakout Board)	Aangesloten op
3V3	Rij 1 op het perfboard. Op deze rij worden alle zaken aangesloten die 3.3V vereisen. Dit zijn: <ul style="list-style-type: none"> <li>• Potentiometer (+Pool).</li> <li>• Middelste contact van schakelaar (0)</li> <li>• VCC van druksensor</li> </ul>
GND	Rij 2 op het perfboard. Op deze rij worden de GND-pinnen van deze componenten aangesloten.
D2	CH1 van 4-kanaals relaisbord. Rode lamp.
D4	CH2 van 4-kanaals relaisbord. Gele lamp.
D5	CH3 van 4-kanaals relaisbord. Groene lamp.
D18	CH4 van 4-kanaals relaisbord. Zoemer.
D19	CH1 van 1-kanaals relaisbord. Blower.
D21	SDA-pin Druksensor.
D22	SCL-pin Druksensor.
D13	Plek X op perfboard. Verbinden met V1 van optocoupler horende bij Laser 1 op perfboard (gebruik hiervoor de dubbele Jack). Verbinden met pulldown weerstand op perfboard.
D27	Plek Y op perfboard. Verbinden met V1 van optocoupler horende bij Laser 2 op perfboard (gebruik hiervoor de dubbele Jack).. Verbinden met pulldown weerstand op perfboard.
D14	Plek Z op perfboard. Verbinden met V1 van optocoupler horende bij Laser 3 op perfboard (gebruik hiervoor de dubbele Jack). Verbinden met pulldown weerstand op perfboard.
D32	Middelste pin Potentiometer.
D35	Pin Manuele modus. Aansluiten aan de (I) zijde van de schakelaar.
D34	Pin Auto-modus. Aansluiten aan de (II) zijde van de schakelaar.
Vin	5V. Rechtstreeks op de voeding of aan een 5V ingang van 1 van de relaisbordjes op voorwaarde dat deze op hun beurt verbonden zijn aan 5V van de voeding.

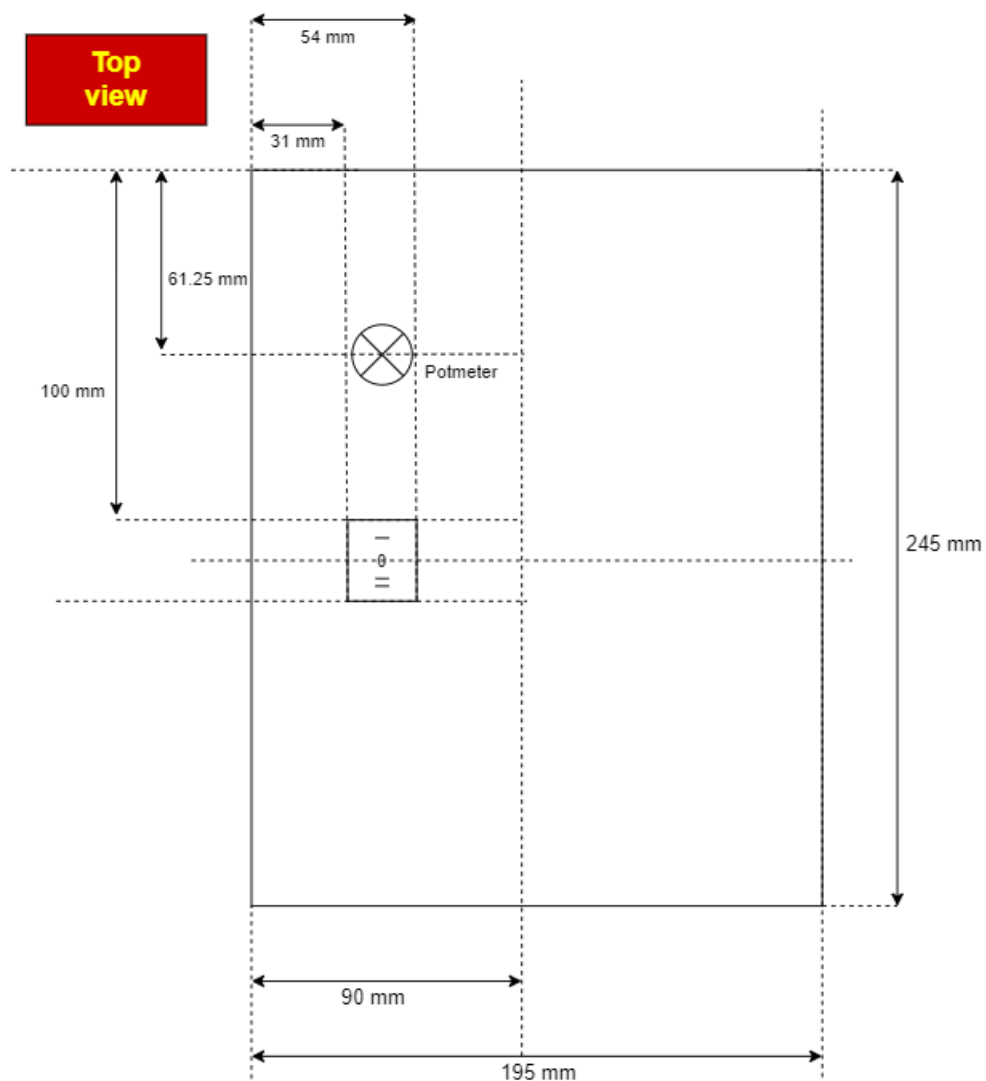
## Schema plaatsing componenten



## Schema onderkant aftakdoos



## Schema voorzijde aftakdoos



## ***Software***

Hieronder vind je een uiteenzetting van de gebruikte software in het project. Dit is software gaande van externe software om schema's te maken tot het programma zelf dat ik schreef om mijn hele systeem te laten functioneren.

### **Confluence**

Platform waar ze bij CERcuits reeds gebruik van maakten. Het is een verzamelplaats voor documentatie, schema's, instructies voor bepaalde processen enzovoort. Je kan het gebruiken om (sprint)planningen en to-do lists te maken, en hoogstwaarschijnlijk nog veel meer. Ik gebruikte het vooral om mijn vooruitgang wat betreft de stage beknopt bij te houden, nuttige links op te slaan en bestellijsten te maken. Ook schema's maken was eenvoudig via een ingebouwd tekenprogramma.

<https://tettra.com/article/confluence-vs-sharepoint/#:~:text=and%20sharing%20information.-,Confluence%20is%20primarily%20used%20for%20knowledge%20management%2C%20while%20Sharepoint%20offers,on%20G2%20is%204.1%2F5.0>

### **EasyEDA**

Dit is gratis software waarmee je eenvoudig elektrische schema's kan ontwerpen. Ik had dit reeds gebruikt tijdens mijn lessen Embedded Devices van mijn opleiding en kon hier reeds mee werken. Het grote voordeel hiervan is dat er een ingebouwde library is van zowat alle soorten elektronica die je kunt bedenken. Aan de hand van de schema's die je hier tekent wordt het eenvoudig om nadien het beoogde systeem te bouwen of zelfs een op maat gemaakte printplaat te bestellen. Een op maat gemaakte printplaat is het niet geworden, al heb ik mijn schema wel veelvuldig gebruikt als hulp bij het solderen. Ook was het handig om dit te hebben als ondersteuning wanneer ik met technische vragen zat en hulp ging vragen bij mijn collega's.



## Het programma zelf

Het programma dat ik geschreven heb voor AFCS is volledig geschreven in de programmeertaal van Arduino. Ze is echter niet geschreven via de Arduino IDE, maar via Visual Studio Code gecombineerd met PlatformIO. Hoofdreden hiervoor is dat het gebruiksvriendelijker is dan de Arduino IDE, en er vaak ook meer libraries te vinden zijn binnen PlatformIO dan binnen de Arduino-omgeving. Gedurende dit project heb ik anders leren programmeren, beter leren programmeren door de sturing van Ruben. Ik heb 'propere' code leren schrijven die aan het einde van de rit veel leesbaarder en schaalbaarder is. Ik heb me laten inspireren op de principes die men toepast bij Object-georiënteerd coderen.

Zo is het voor iemand die later met mijn code aan de slag zou willen gaan veel aangenamer en duidelijker om te beginnen. Ik was gewend vanuit mijn opleiding om gewoon zo lang te sleutelen aan mijn code tot het deed wat het moest doen. Daar stopte het meestal. Nu heb ik geleerd dat het ook belangrijk is om na te denken over wat er nadien met mijn code zou gebeuren.

Dit zijn de voornaamste zaken die ik vroeger niet deed, en nu wel heb geleerd.

- Zorg ervoor dat je je programma in zo klein mogelijke stukjes hakt; maak van alle functionaliteiten die je systeem moet hebben een aparte functie. Op deze manier kan je bij het ontwikkelen ervan
  - Zorg ervoor dat een functie slechts 1 verantwoordelijkheid heeft.
  - Zorg ervoor dat functies niet afhankelijk zijn van andere functies.
  - Zet geen lees- en schrijfacties in je functies, maar hou in plaats daarvan statussen bij. Je kan deze statussen ook gebruiken om code te testen zonder er fysieke componenten aan te koppelen.
  - Maak 1 functie om alle inputs van het systeem te lezen aan het begin van een cyclus.
  - Maak 1 functie om alle outputs weg te schrijven aan het einde van een cyclus.
  - Gebruik geen 'blocking functions'. Dit zijn stukjes code die de rest van de code beletten om uitgevoerd te worden, vaak zijn dit zaken als:
    - Wachten tot een bepaalde tijd verstreken is
    - Wachten tot een variabele een bepaalde grenswaarde heeft bereikt
    - ➔ Vermijd While lussen.
- ➔ Door dit te doen vermijd je dat functies met elkaar gaan conflicteren.

Verder in dit verslag heb ik screenshots bijgevoegd van de AFCS-code, telkens voorzien van extra uitleg over dat specifieke stukje code. Ik heb getracht om alles zo duidelijk mogelijk te voorzien van commentaar opdat je goed snapt hoe de code in elkaar zit.

## Imports en initialisatie (1)

```
1  #include <Arduino.h>
2  #include <WiFi.h>
3  #include "time.h"
4  #include <DFRobot_LWLP.h>
5
6  // Define the relay pins
7  #define RED_PIN 2
8  #define YELLOW_PIN 4 // for all the other systems
9  #define GREEN_PIN 5
10 #define BUZZER_PIN 18
11 #define BLOWER_PIN 19
12 #define LASER_PIN_1 13
13 #define LASER_PIN_2 27
14 #define LASER_PIN_3 14
15 #define MANUAL_PIN 35 //switch position I
16 #define AUTO_PIN 34 //switch position II
17 #define POTENTIO_PIN 32
```

De eerste 4 regels van de code bevatten allemaal 'include' statements. Deze statements voegen bepaalde libraries toe die het mogelijk maken om specifieke functies en/of syntax te kunnen gebruiken.

Regel 1 importeert de Arduino library, hierdoor kunnen we in deze specifieke programmeertaal coderen. Regel 2 voegt een Wi-Fi-bibliotheek toe. Deze had ik toegevoegd om de mogelijkheid tot internetverbinding op de ESP32 open te houden. Als derde heb ik de 'time' library toegevoegd om timers en dergelijke te kunnen gebruiken. Tot slot voegde ik een library toe die hoorde bij de druksensor. Hierdoor kon ik functies aanroepen die de meetdata van de sensor konden verwerken en presenteren.

Regel 7 tot 17 definiëren de nummers van de gebruikte GPIO pinnen van de ESP32 en hun bijhorende naam.

## Imports en initialisatie (2)

```
25
26 //initializing variables
27 int redStatus = 0;
28 int yellowStatus = 0;
29 int greenStatus = 0;
30 int buzzerStatus = 0;
31 int blowerStatus = 0;
32 int laserStatus[3] = {0, 0, 0};
33 //int laser_1_Status[3] = {0, 0, 0};
34 int manualStatus = 0;
35 int autoStatus = 0;
36 const char *ntpServer = "pool.ntp.org"; //
37 DFRobot_LWLP lwlp; // initialize pressure sensor
38 DFRobot_LWLP::sLwlp_t data; // initialize data from pressure sensor
39 float pressure = 0.0;
40 int potentiometerValue = 0;
41 float maxPressure = 0.0;
42 unsigned long previousMillis;
43 unsigned long currentMillis;
44 const long blinkInterval = 500;
45 const long blinkDuration = 3100;
46 const long blowerOffDelay = 900000;
47 const long pressureDuration = 10000;
48 const long pressureInterval = 600;
49 int count = 0;
50 int pressureCount = 0;
51 int warningFlag = 0;
52 float pressureArray[256];
53 uint8_t pressureIndex = 0;
54 int pressureLength = 256;
```

Hier worden variabelen die we later in het programma zullen gebruiken geïnitieerd. Met initialisatie bedoel ik dat er wordt aangegeven welk datatype er bij deze variabele hoort (bv. **Integer**, **character**, ...) en welke standaardwaarde ze eventueel dienen te hebben wanneer het programma voor de eerste keer zal worden gerund. Zo zie je dat de statussen van alle lampen, zoemer, blower en lasers aanvankelijk op 0 worden gezet. 0 betekent hier hetzelfde als 'uitgeschakeld, inactief'. Lijn 34 en 35 beschrijven de variabelen die de modus van AFCS zullen aangeven.

De variabele voor de druksensor en de variabele waarin de meetdata van deze terecht zal komen worden ook gecreëerd. Daarna worden de waarden voor de gemeten differentieële druk, maximale druk die gemeten mag worden vooraleer de filterzak als vol beschouwd wordt en de variabele waarin de data van de potentiometer (hoeveel deze opengedraaid is) zal worden opgeslagen. De variabelen `previousMillis` en `currentMillis` zullen timestamps, een tijd in milliseconden stockeren. Het gebruik hiervan licht ik toe wat lager in dit document aan de hand van de toepassing ervan in mijn programma.

Daarna worden er nog enkele constanten aangemaakt. Dit zijn – zoals de naam het zegt – zaken die constant of onveranderd blijven in heel het programma. Ze worden gebruikt als waarden ter vergelijking, meer specifiek om te checken of het aantal verstreken milliseconden gelijk is aan een bepaalde vaste waarde.

Lijn 49 en 50 zijn counters, deze zaken worden gebruikt om bij te houden hoeveel keer iets al gebeurd is in het programma. Regel 52 tot 54 beschrijven de 'opslagplaats' waarin de geregistreerde drukmetingen terecht zullen komen. De grootte ervan en de index wordt meegegeven, om te weten hoeveel waarden er kunnen worden opgeslagen en om te kunnen zoeken naar een meting met specifieke index (nummer in het rijtje) uit de hoop metingen.

## Wi-fi-verbinding

```
57 //Functions
58
59 // Initialize WiFi
60 void initWiFi() {
61     WiFi.mode(WIFI_STA);
62     WiFi.begin(ssid, password);
63     Serial.print("Connecting to WiFi ..");
64     while (WiFi.status() != WL_CONNECTED) {
65         Serial.print('.');
66         delay(1000);
67     }
68     Serial.println(WiFi.localIP());
69 }
```

Deze functie heeft als doel verbinding op te zetten tussen de ESP32 en een bestaand Wi-Fi-netwerk. Ik had dit opgenomen in mijn project om, indien er voldoende tijd was, de gemeten data op te sturen en op te slaan in de Cloud ter analyse en voor eventueel secundair gebruik. Zie Verbetervoorstellen onderaan dit document voor de mogelijkheden hiervan.

## Mapping potentiometer

```
71 // Potentiometer mapping
72 float floatMap(float x, float in_min, float in_max, float out_min, float out_max) {
73     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
74 }
75
```

De potentiometer in het systeem diende om de grenswaarde met betrekking tot de differentiële drukmeting handmatig af te stellen. Zo konden potentiële verschillen tussen de verschillende systemen worden opgevangen en kon na een periode van tweaken een accurate maximum drukwaarde worden ingesteld die kon worden gebruikt als trigger voor het monitoringsysteem van de filterzakken. Eens die ingestelde drukwaarde overschreden werd voor het systeem in kwestie, zou de gele lamp aangeven dat de filterzak vol zat. Voorwaarde om dit te doen werken was dat de gemeten spanning over de potentiometer, die recht evenredig is aan de overeenkomstige weerstand op dat moment, zou worden gekoppeld aan een waarde tussen 0 en 500. Deze functie 'mapt' de potentiometerwaarden programmatorisch aan waarden die kunnen worden geïnterpreteerd en gebruikt verder in het programma.

## Lezen van inputs

```
76 //Read all inputs
77 void Digital_IN(){
78     laserStatus[0] = !digitalRead(LASER_PIN_1); // Read the status of Laser 1
79     laserStatus[1] = !digitalRead(LASER_PIN_2); // Read the status of Laser 2
80     laserStatus[2] = !digitalRead(LASER_PIN_3); // Read the status of Laser 3
81     manualStatus = digitalRead(MANUAL_PIN);
82     autoStatus = digitalRead(AUTO_PIN);
83     data = lwlp.getData();
84     pressure = abs(data.presure);
85     potentioValue = analogRead(POTENTIO_PIN);
86 }
```

Deze functie heeft als doel alle inputs van de ESP32 te lezen aan het begin van de cyclus van het programma. Zo wordt er eerst gecheckt bij de pinnen horende bij de lasers of deze hoog zijn, wat wil zeggen dat de desbetreffende laser actief is. Deze waarden worden bijgehouden in een array genaamd laserStatus. Deze array kan dan verder in het programma worden gebruikt om gekoppeld te worden aan bijhorende acties (lampen inschakelen, relais inschakelen, ...). De status van de schakelaar wordt uitgelezen om te weten in welke modus het systeem zich bevindt. Wat lager wordt de data van de druksensor (lwlp) geanalyseerd om te weten wat de huidige druk is binnenin de filterzak. Tot slot wordt de 'rauwe' data van de potentiometer binnengehaald (een spanningsniveau tussen 0 en 3.3V vertaald naar een integer dat daarmee correspondeert).

## Schrijven van outputs

```
88 //Control all outputs
89 void Digital_OUT(){
90     digitalWrite(YELLOW_PIN, yellowStatus);
91     digitalWrite(RED_PIN, redStatus);
92     digitalWrite(GREEN_PIN, greenStatus);
93     digitalWrite(BUZZER_PIN, buzzerStatus);
94     digitalWrite(BLOWER_PIN, blowerStatus);
95 }
96
```

Als er een functie bestaat om alle inputs te lezen, hebben we natuurlijk ook een functie nodig die alle outputs wegschrijft. Deze functie wordt bovenaan de functie geplaatst om conflicten m.b.t. variabelen die lager worden aangeroepen te vermijden. Logischerwijs wordt ze wel pas helemaal onderaan uitgevoerd, vlak voor de cyclus is afgerond. Vervolgens kunnen alle inputs weer worden gelezen en begint alles opnieuw. Zoals je kan zien heeft AFCS slechts 5 outputs (de gele, rode & groene lamp, de zoemer en de relais die de contactor voor de blower schakelt).

## Lampfunctie

```
97 // Control lamp with different possible cases
98 void lamp(int lampOption) {
99     switch (lampOption) {
100
101         // Turn everything off (at startup)
102         case 1:
103             redStatus = 0;
104             yellowStatus = 0;
105             greenStatus = 0;
106             break;
107
108         // Warning signal before turning blower on
109         case 2:
110             static unsigned long blinkTimerStart = 0;
111             blinkTimerStart = millis();
112             while (millis() - blinkTimerStart <= blinkDuration) {
113                 buzzerStatus = 1;
114                 currentMillis = millis();
115                 if (currentMillis - previousMillis >= blinkInterval && count <= 6){
116                     previousMillis = currentMillis;
117                     count += 1;
118                     redStatus = !redStatus;
119                     Digital_OUT();
120                 }
121             }
122             warningFlag = 0;
123             buzzerStatus = 0;
124             count = 0;
125             break;
126
```

```
127 // Indicate that filter bag is full
128 case 3:
129     static unsigned long pressureTimerStart = 0;
130     pressureTimerStart = millis();
131     greenStatus = blowerStatus;
132     while (millis() - pressureTimerStart <= pressureDuration) {
133         currentMillis = millis();
134         Serial.print("pressure count = ");
135         Serial.println(pressureCount);
136         if (currentMillis - previousMillis >= pressureInterval && pressureCount <= 5){
137             previousMillis = currentMillis;
138             pressureCount += 1;
139             yellowStatus = !yellowStatus;
140             Digital_OUT();
141         }
142     }
143     pressureCount = 0;
144     break;
145
146 // Indicate the status of the blower
147 case 4:
148     if (blowerStatus == 1) {
149         greenStatus = 1;
150     }
151     if (blowerStatus == 0 && autoStatus == 0) {
152         greenStatus = 0;
153     }
154     break;
155
156 case 5:
157     redStatus = 0;
158     break;
```

```

159
160     case 6:
161         yellowStatus = 0;
162         break;
163
164     case 7:
165         greenStatus = 0;
166         break;
167
168     case 8: // Off mode
169         redStatus = 1;
170         break;
171
172     case 9: // Auto mode
173         redStatus = 1;
174         greenStatus = 1;
175         break;
176     }
177 }

```

## Lasercheck

```

179 // Check if at least one of the lasers is active
180 bool isAnyElementHigh(int laserStatus[])
181 {
182     //to rework add array
183     for (int i = 0; i < 3; ++i)
184     {
185         if (laserStatus[i] == HIGH)
186         {
187             return true; // Return true if at least one element is HIGH
188         }
189     }
190     return false; // Return false if none of the elements are HIGH
191 }
192

```

## Blowerfunctie

```
193 // Control the blower
194 void blower(int laserStatus[]) {
195     static unsigned long blowerTimerStart = 0; // Variable to store the start time for warning blink
196
197     if (isAnyElementHigh(laserStatus)) { // If one of the LASER_PINS is high
198         blowerStatus = 1;
199         blowerTimerStart = 0; // Reset the timer
200     } else { // If all LASER_PINS are low
201         if (blowerTimerStart == 0) { // If timer is not running
202             blowerTimerStart = millis(); // Start the timer
203             Serial.print("blowerTimerStart = ");
204             Serial.println(blowerTimerStart);
205         }
206         else {
207             if ((millis() - blowerTimerStart) >= (blowerOffDelay - blinkDuration) && blowerStatus == 1) { // If blower is about to turn off
208                 warningFlag = 1;
209             }
210             if (millis() - blowerTimerStart >= blowerOffDelay) { // If blowerOffDelay time has passed
211                 blowerStatus = 0;
212                 warningFlag = 0;
213                 blowerTimerStart = -1; // Reset the timer
214             }
215         }
216     }
217 }
```

```
219 int sort_desc(const void *cmp1, const void *cmp2)
220 {
221     // Need to cast the void * to int *
222     int a = *((int *)cmp1);
223     int b = *((int *)cmp2);
224     // The comparison
225     return a > b ? -1 : (a < b ? 1 : 0);
226     // A simpler, probably faster way:
227     //return b - a;
228 }
```



## Monitor druk in filterzak

```
230 void monitorPressure(float pressure, float maxPressure, int yellowStatus) {
231
232     pressureArray[pressureIndex] = abs(data.presure);
233     Serial.print(data.presure);
234
235     pressureIndex++;
236     Serial.println("Starting median calculation");
237     qsort(pressureArray, pressureLength, sizeof(pressureArray[0]), sort_desc); Serial.print("Pressure = ");
238     float median = pressureArray[int(pressureLength/2)];
239     Serial.print("Median value = ");
240     Serial.println(median);
241     Serial.println(pressureIndex);
242
243     if (median > maxPressure ) {
244
245         lamp(3); // Turn on the yellow LED
246     }
247     if (median < maxPressure && yellowStatus == 1) {
248         lamp(1); // Turn off the yellow LED
249     }
250 }
```

```
254 void setup() {
255
256     // Initialize relay pins as OUTPUT
257     pinMode(YELLOW_PIN, OUTPUT);
258     pinMode(RED_PIN, OUTPUT);
259     pinMode(GREEN_PIN, OUTPUT);
260     pinMode(BUZZER_PIN, OUTPUT);
261     pinMode(BLOWER_PIN, OUTPUT);
262     pinMode(LASER_PIN_1, INPUT);
263     pinMode(LASER_PIN_2, INPUT);
264     pinMode(LASER_PIN_3, INPUT);
265     pinMode(MANUAL_PIN, INPUT);
266     pinMode(AUTO_PIN, INPUT);
267     delay(500);
268     // Initially turn off all relays
269     digitalWrite(YELLOW_PIN, HIGH);
270     digitalWrite(RED_PIN, HIGH);
271     digitalWrite(GREEN_PIN, HIGH);
272     digitalWrite(BUZZER_PIN, HIGH);
273     digitalWrite(BLOWER_PIN, LOW);
274     delay(500);
275
276     digitalWrite(YELLOW_PIN, LOW);
277     digitalWrite(RED_PIN, LOW);
278     digitalWrite(GREEN_PIN, LOW);
279     digitalWrite(BUZZER_PIN, LOW);
```

```

285 //Start wifi connection
286 //initWiFi();
287 //configTime(0, 0, ntpServer);
288
289 // Start Serial communication
290 Serial.begin(9600);
291 int N = 0;
292 while (lwlp.begin() != 0 or N<=10) {
293     digitalWrite(BUZZER_PIN, !digitalRead(BUZZER_PIN));
294     N++;
295     delay(500);
296 }
297
298 }

```

```

302 void loop() {
303
304     currentMillis = millis();
305     Digital_IN();
306     float maxPressure = floatMap(potentialValue, 0, 4095, 0, 500); // map the measured potentiometer
307
308     // Things to do when the switch is set to auto mode (II)
309     if (autoStatus == 1) {
310         if (blowerStatus != 1) {
311             lamp(9);
312         }
313         Digital_OUT();
314         if (blowerStatus == 0 && isAnyElementHigh(laserStatus)) {
315             lamp(2);
316         }
317         if (warningFlag == 1 ) {
318             lamp(2);
319             warningFlag = 0;
320         }
321         // Continuously run the blower function
322         blower(laserStatus);
323         monitorPressure(pressure, maxPressure, yellowStatus);
324         lamp(4);
325     }
326 }

```

```

327 // Things to do when the switch is off
328 if (manualStatus == 0 && autoStatus == 0) {
329     lamp(8);
330     Digital_OUT();
331     if (blowerStatus == 1 ) {
332         lamp(2);
333     }
334     blowerStatus = 0;
335     lamp(4);
336     monitorPressure(pressure, maxPressure, yellowStatus);
337 }
338
339 //Things to do when switch is set to manual mode (I)
340 if (manualStatus == 1) {
341     if (blowerStatus == 0) {
342         lamp(2);
343     }
344     lamp(1);
345     blowerStatus = 1;
346     monitorPressure(pressure, maxPressure, yellowStatus);
347     lamp(4);
348
349 }
350 Digital_OUT();
351 }

```

## Troubleshooting

Hier vindt u problemen – en hun oplossingen (!) – terug die ik tegenkwam tijdens mijn project. Hopelijk vergewissen ze u van het maken van dezelfde fout, zij het om zelf een nieuw systeem te bouwen of een bestaand systeem aan te passen naar uw noden.

Probleem	Oplossing
Afbreken van koperdraadjes	Gebruik geen harde draden (die in een bepaalde vorm blijven staan) maar gebruik zachte draden. Zachte draden zijn minder onderhevig aan metaalmoeheid.
Loskomen van solderingen	Gebruik zoveel mogelijk aansluitingen via schroefterminals.
Loskomen van draadjes uit schroefterminals	Gebruik adereindhulzen die je vastknijpt op het uiteinde van een kabeltje met de bijgeleverde krimptang. Draai de schroefterminal helemaal open met behulp van een kleine schroevendraaier, schuif

	de adereindhuls erin en draai de schroefterminal goed vast. Door deze hulzen te gebruiken, wordt het uiteinde van een kabeltje wat dikker en kan je deze beter vastklemmen in de terminal.
Resetten van ESP32	Elektromagnetische storing. Plaats contactor en ESP32 verder uit elkaar.
Foute spanningsniveaus	<p>Aansluitingen nakijken</p> <ul style="list-style-type: none"> <li>- Zijn schroefterminals goed aangedraaid?</li> <li>- Zijn solderingen proper gebeurd, zonder barsten, zijn er geen onopgevulde gaatjes?</li> <li>- Zijn draden op de juiste plek aangesloten (geen 3.3V aan 5V en vice versa) ?</li> </ul> <p>Onderdelen nakijken</p> <ul style="list-style-type: none"> <li>- Geven ze meetwaarden die liggen binnen de verwachtingen? Indien niet is dit onderdeel mogelijk defect.</li> </ul>

## Verbetervoorstellen

### ***Noodstop***

In de huidige opstelling werd de noodstop uit het circuit gehaald om de aansluiting van de contactor aan de blower te vergemakkelijken. Om het systeem veiliger te maken zou het goed zijn om terug een noodstop te introduceren. Voorwaarde is wel dat deze noodstop dan de mogelijkheid verschaft om deze uit te lezen via een van de ingangspinnen van de ESP32.

Zonder de mogelijkheid om de laatste status op te vragen weet AFCS niet of de noodstop ingeschakeld is of niet na een reset.

## ***Internetverbinding AFCS***

Aangezien de ESP32 Wi-Fi aan boord heeft, is het mogelijk om de verzamelde data door te sturen en op te slaan in de cloud. Eens dit gelukt is volgen er nog andere uitbreidingsmogelijkheden.

### **Online dashboard**

- Predictive maintenance

### ***Extra***

- Controlesysteem in bureau om te zien of er problemen zijn met 1 van de AFCS of filterzak = vol
- Temperatuur in ruimte slim wijzigen door rekening te houden met extra warmte
- Winter: extra warmte gebruiken om slim verwarming te regelen
- Teveel aan warmte afgeven aan trainingshal naast kamer met lasers
- Ruben had dit al gezegd, lucht moet wel gefilterd worden door stofdeeltjes en die kost dient dan betaald te worden door eigenaar hal (die altijd te koud is)
- Zachte kabeltjes met adereindhulzen
- Eigen PCB ontwerpen met enkel de nodige sensoren/chips op

- Extra indicaties met signaallamp om bepaalde errors aan te geven

Vb. alle 3 lampen laten pinken in interval

1x = error druksensor

2x = ...

Oplossing voorzien zodat het systeem niet uitschakelt als je overschakelt van Auto-modus naar Aan-modus en vice versa

## **Besluit**

Niet meer toe gekomen.

## **Bibliografie**

Niet meer toe gekomen.