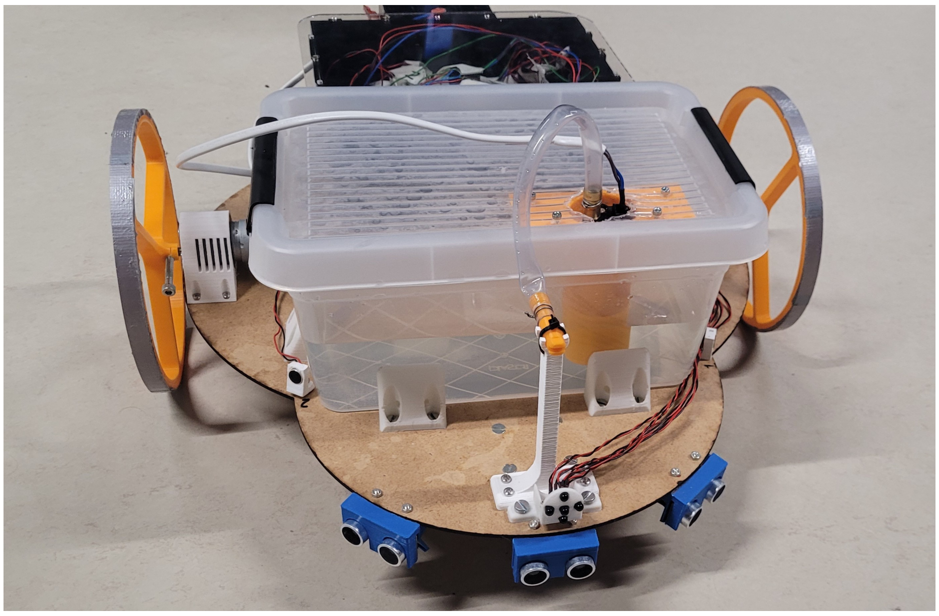
Autonomous Fire Supression Vehicle

University of Southern Denmark, 3rd Semester project, Group 4



Project participants:

Frank Dahl Andersen, Jonas Studsgaard Nielsen,

Max Gallardo Boluda, Rabea Schwarm, Virgil Djonny Muxoll

Project supervisors:

Kasper Mayntz Paasch, Henrik Andersen

Date of hand-in: 2nd January, 2023

University: University of Southern Denmark (SDU) - Alsion 2, 6400 Sønderborg
Faculty: The technical faculty (TEK)
Education: BSc./BEng. in Mechatronics
Project: MC-SPRO3, 3rd Semester project
Project group: Group 4
Project period: 1st September (2022) to 2nd January (2023)
Project title: Autonomous Intelligent Vehicle
Project participants:

Frank Dahl Andersen
Jonas Studsgaard Nielsen
Max Gallardo Boluda
Rabea Schwarm
Virgil Djonny Muxoll

Project supervisors:

Kasper Mayntz Paasch
Henrik Andersen

No. of pages: 53
No. of appendixes: 9

Contents

List of Figures

1. Introduction

This report revolves around the development, design and test of an autonomous intelligent vehicle, as part of the 3rd semester on the mechatronics bachelor at the University of Southern Denmark - Sønderborg campus.
The project supervisors have set a budget of 1000DKK for new components through the university and anything already on hand or otherwise obtainable will be unaccounted for, in the total budget.
As a result of the previous semester project being significantly focused on the mechanical aspects, this project has a deeper focus on hard- and software.
According to the set parameters, the following report documents the entirety of the project starting from the 1st of September 2022 until the 2nd of January 2023. The chosen project being an autonomous fire supression vehicle, as will be elaborated upon in the following chapters.
The project serves as a learning exercise for students.

2. Project formulation

2.1 Background

Fire suppression in todays modern buildings is mostly solved by the use of fire sprinkler systems [1]. The cost to install one of these sprinkler systems costs between 70 - 270 DKK pr m^2 [2]. The sprinkler systems rise significantly in price when retro fitted into buildings, the price is between 135-680 DKK per m^2 [2]. When upscaled to larger structures the cost of sprinkler systems adds up and could have potentially unwanted side effects, such as water damage to inventory in warehouses or damage to electronics in datacenters [3]. Therefore alternate solutions could prove useful in warehouses or similar. In the past there has been attempts to make a robot that aids humans in fire suppression [4]. The robots way of aiding human fire fighters is by keeping them out of harm, because the robot is being remote controlled by the fire fighter at a safe distance.

2.2 Problem statement

The aim of this project is to investigate the feasibility of constructing a small-scale autonomous firefighting vehicle. To explore this, a "proof of concept" will be developed and built. This prototype should have the ability to autonomously locate and suppress minor fires, to prove that the concept can be applied in the real-world environment. Preferably it should be achieved in a way that, provided sufficient money and time, could be upscaled to a size where it would be practical for a real-world application.

2.3 Requirements & Research question

Is development of a fully autonomous fire suppression vehicle possible, while also being a viable alternative to potentially expensive modern day sprinkler systems?

2.3.1 Requirements

- Include at least two analog sensors.
- Include at least one analog filter.

- Stay within a 1000 kr. development budget.
- Able to locate fire.
- Have a PCB for electronics.
- Able to extinguish/supress fire.
  - Fire should at maximum be located 30cm above the floor.
- Move to fire autonomously, without moving into fire

2.3.2 Secondary Requirements

- Wireless connection with smoke detectors.
- Call 112.
- Charging and refilling station.

2.4 Project delimitation

In the development and construction of the autonomous fire suppression vehicle, certain objectives will be outside the scope of this project.
In detection of fires, the sensors will not have the ability to distinguish different types of fires, due to the complexity of this. Furthermore, the coverage area of the vehicle will be less than $20m^2$.
It will only extinguish/suppress fires consisting of small dry combustibles.
The vehicle will not have heat shielding, due to the cost and complexity of implementation.

3. Research and development

3.1 Idea generation

To be able to generate as many ideas as possible each group member made their own mindmap, with anything which could be considered an autonomous vehicle. All the mindmaps were then combined into one (3.1).

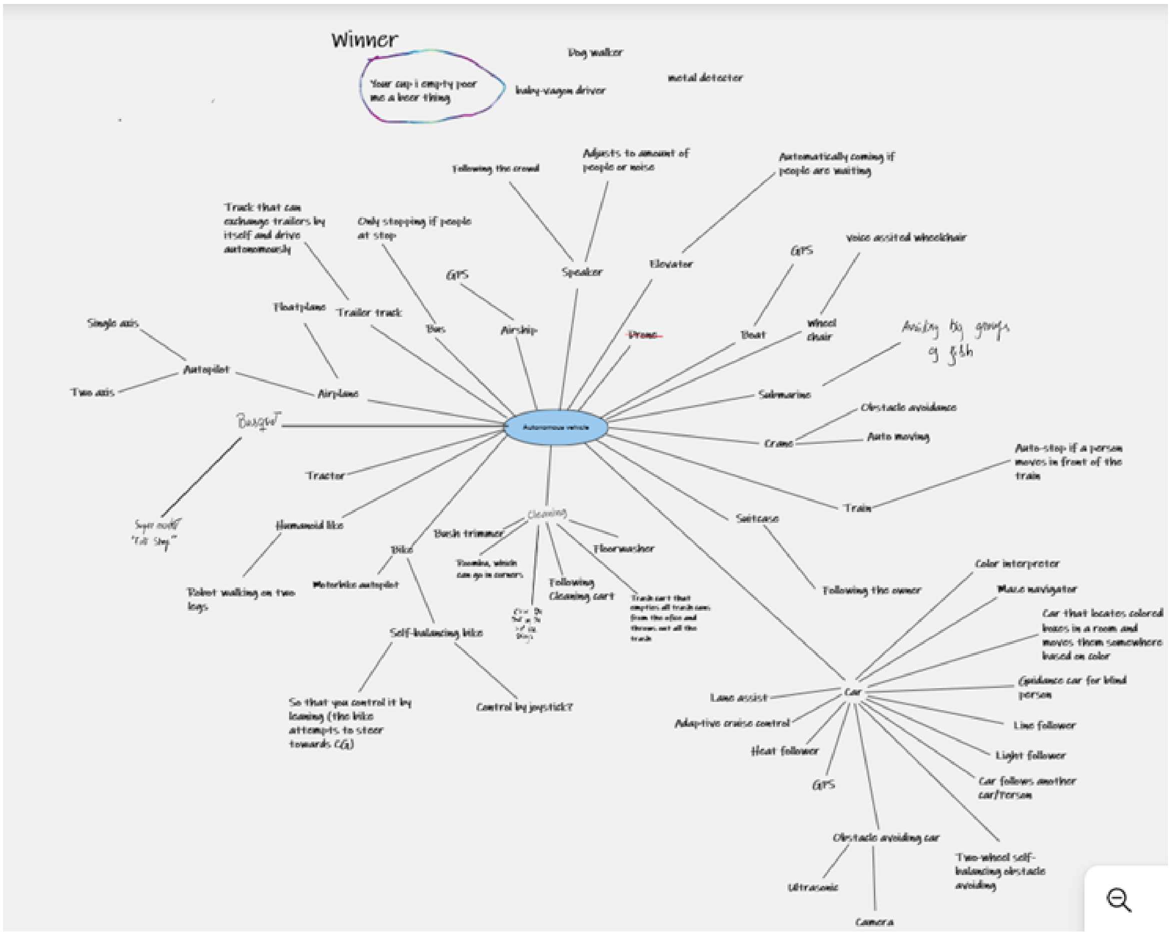

Winner
Your cup i empty poor me a beer thing
baby-vagon driver
Dog walker
metal detector
Following the crowd
Adjusts to amount of people or noise
Automatically coming if people are waiting
Truck that can exchange trailers by itself and drive autonomously
Only stopping if people at stop
GPS
Speaker
Elevator
GPS
Voice assited wheelchair
Single axis
Floatplane
Trailer truck
Bus
Airship
~~Drone~~
Boat
Wheel chair
Autopilot
Two axis
Airplane
Submarine
Obstacle avoidance
Autonomous vehicle
Crane
Auto moving
Tractor
Auto-stop if a person moves in front of the train
Train
Suitcase
Humanoid like
Bush trimmer
Cleaning
Floorwasher
Bike
Roomba, which can go in corners
Robot walking on two legs
Motorbike autopilot
Following Cleaning cart
Trash cart that empties all trash cans from the office and throws out all the trash
Following the owner
Color interpreter
Maze navigator
Car that locates colored boxes in a room and moves them somewhere based on color
Self-balancing bike
So that you control it by leaning (the bike attempts to steer towards CG)
Control by joystick?
Lane assist
Car
Guidance car for blind person
Adaptive cruise control
Line follower
Heat follower
Light follower
GPS
Car follows another car/person
Obstacle avoiding car
Two-wheel self-balancing obstacle avoiding
Ultrasonic
Camera

Figure 3.1: Initial mindmap

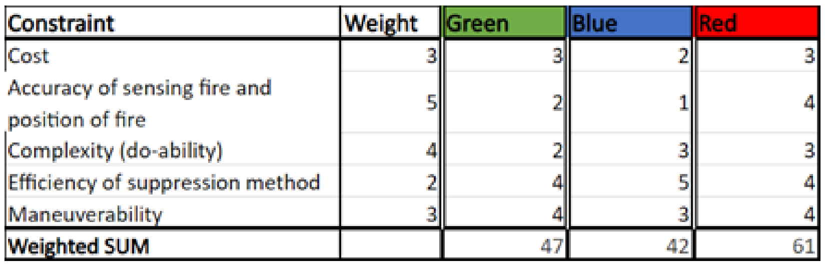
From that mindmap every member choose a top five. At the next meeting the group then voted for their favorite idea. The top idea was a robot that would detect if your glass was empty and would then proceed to fill up the glass with beer. After some consideration, it was decided that the idea might be too complex. Hence the beer refilling robot got changed to a fire supression/extinguishing robot.

3.2 Morphology

To define the project in full details, every group member made their own morphology chart. Each member got the same categories given, which were: Chassis construction, Drivetrain construction, obstacle avoidance method, extinguishing method and fire detecting method. All the morphology charts were then combined into a master morphology chart. See attached appendix 1.
In the master morphology chart three different paths were made to decide how the project would be approached in the end. After drawing the paths, a decision matrix (3.2) was made. The paths were then judged on a scale from 1-5, in the following categories:

- Cost, would it be able to stay within the 1000dkk budget
- Accuracy of sensing fire and position of the fire
- Complexity, how easy is it to build, where 5 is very complex
- Effiency of surpression method
- Maneuverability

The group decided together which score each category for each path would get.



Constraint	Weight	Green	Blue	Red
Cost	3	3	2	3
Accuracy of sensing fire and position of fire	5	2	1	4
Complexity (do-ability)	4	2	3	3
Efficiency of suppression method	2	4	5	4
Maneuverability	3	4	3	4
Weighted SUM		47	42	61

Figure 3.2: Selection matrix

The weighted sum was calculated by multiplying the score given with the weight of the category, and those multiplications were then added together to give the weighted score.
That means the red path was chosen, based on it getting the highest weighted sum.

3.3 User interaction

Since the project revolved around fire suppression, reaching out to the fire department seemed logical. Therefore a meeting with the head of the fire department was set up, to be able to explain the project and ask some relevant questions.

3.3.1 The best way to extinguish a fire

During the meeting, it was found out that depending on the type of fire and material on fire, different ways of diminishing a fire are most optimal. Initial research pointed the project in the direction of using water mist, as it makes the surroundings more humid, and therefore cooling it down. After consulting the head of the fire department, using bigger droplets, with a drop size of around 2mm, would prove to be more effective. As smaller droplets, in the range of 0.1-0.3mm, will evaporate before hitting the material initially on fire. The smaller droplets are therefore not cooling the material on fire down but resulting in an atmospheric decrease in temperature around the fire, making it difficult for the fire to spread. Larger droplets are therefore better used for putting out a fire, and misting is a better alternative to get rid of smoke particles or toxins in the air, as they will mix with the small water particles, and get dragged to the ground. Furthermore, to avoid carrying a huge tank of water, adding foaming liquid was suggested, as it is going to help diminishing the fire quicker. Water combined with foaming liquid penetrates the burning material more deeply, as the foaming agent acts to destroy the surface tension of the water.

3.3.2 Future testing

Talking to the head of the fire apartment was a huge success, as they offered to help with testing in the future. For future testing, the opportunity to test the project with the fire department on their premisses was a nice addition to the project. This also leads the testing to be as safe and as controlled as possible, since professionals are involved.

3.3.3 Relevance

After the talk with the head of the fire department, it was clear that the project, would be very relevant for potential future use, in the real world. He mentioned, when he started working in 1989 compared to now, nothing has changed in the procedure of diminishing fire. Which means when they get a call, they have a very limited amount of time to prepare their vehicle, drive to the desired destination, open a technical closet which states which alarm was triggered, find the paper plan of the building and then find the triggered alarm on the map. This method leads to lost time, as the firemen often end up being at the opposite part of the building, of where the fire is located. This means that important time is wasted on locating the fire, and afterwards moving to the destination of the fire. This project has potential to aid them in their job, as it suppresses the fire and therefore gives the fire department more time to arrive on site. Furthermore, the robot would pick up on the frequency of the fire alarm. This feature could in the future be developed, so that the robot initially sends the destination of the triggered fire alarm to the fire department. This way the fire department saves precious time, on locating the fire, as the robot has already done that.

The head of the fire department also mentioned potential places where our project would be especially relevant. This includes retirements homes and other places where people have difficulty moving themselves out of the building, in case of a fire occurring. This way the robot provides the people in the building with more time to evacuate, as the robot will detect and try to extinguish the fire as soon as it picks up on the infrared rays from it, or the sound of the fire alarm. This also diminishes the chances of the fire, growing out of control, before the fire department arrives.

It was also mentioned that to make the robot even more efficient for real world use, the head of the fire department suggested to use smoke detectors on top of infrared detectors. The reason is that most fires create smoke before flames, and hence it would be very useful, as this would lead the robot to stop the fire in its earliest stages, so that no flames will erupt.

4. Risk analysis

This process was focused on identifying the likelihood of a risk to the project; working on determining the probability of occurrence, the resulting impact, and the additional safeguards that mitigate this impact.

First a structure tree was made, to identify all the components and sub-components that makes up the robot. A D-FMEA (Design Failure Mode Effect Analysis) was made on the water tank (see D-FMEA in appendix 9). The FMEA was made on the water tank, as that was considered a rather simple component and our project timeframe, did not allow us to do a complete FMEA analysis of every component (nor would we have the time to act on the actions the analysis would identify).

The analysis of the individual part consists of the following:

Item Function

- Defining the function of the part

Potential failure mode

- State the different ways the component could fail.

Potential Effect(s) of Failure

- Evaluate the effect of each failure.

Potential cause(s)/Mechanism(s) of failure

- Evaluate the cause of failure.
- For lowest level components, the cause is either a noise- or control factor. For sub-assemblies, the cause is a failure of the components.

Current Design Controls Prevention

- Describing what is being done at the current iteration of the design, to attempt to prevent the failure mode through design.

Current Design Controls Detection

- Elaborates on the measures used to detect and find potential defects and/or discover design weaknesses.

Recommended Action(s)

- What actions are to be done, in order to counter, the risk of potential failure, if any action is taken at all.

In order to identify which failures is the most critical and thus is the most urgent to prevent; each failure mode is evaluated in three categories:

- Severity of the failure modes effect (SEV): A value applied on a scale of 1 (low) to 10 (high).
- Frequency of failure occurring (OCCUR): A value applied on a scale of 1 (infrequent) to 10 (frequent).
- Detectability/preventability (DETEC): A value assigned on a scale of 1 (very detectable) to 10 (not detectable).

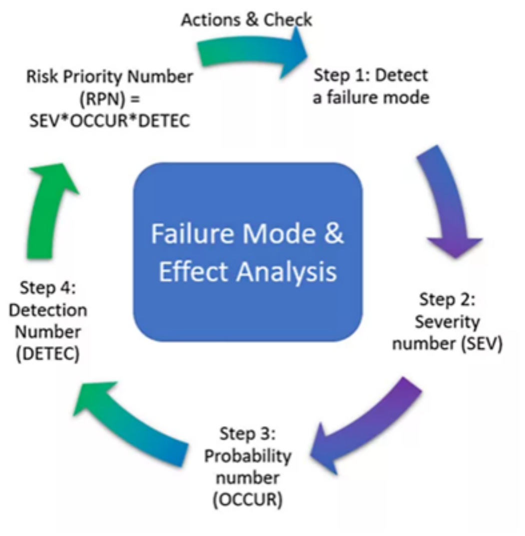

Actions & Check
Step 1: Detect a failure mode
Step 2: Severity number (SEV)
Step 3: Probability number (OCCUR)
Step 4: Detection Number (DETEC)
Risk Priority Number (RPN) = SEV*OCCUR*DETEC
Failure Mode & Effect Analysis

Figure 4.1: Faliure mode & effect analysis [5]

The RPN is then determined by multiplying SEV, OCCUR, and DETEC. Calculating the RPN gives a number ranging all the way anywhere from 1 (low risk) to 1,000 (high risk). makes it possible to define what is acceptable and unacceptable for the failure being analyzed. See appendix 9 for in depth analysis of water tank.

5. Mechanical design

5.1 Motor dimensioning

In order to calculate the requirements for the drive motors the following calculations were made.
Firstly, a free body diagram of the vehicle is made (5.1), where the gravity and normal forces cancel out. The motor applies a torque at the centre of the wheel, which has a resultant force on the perimeter due to friction.

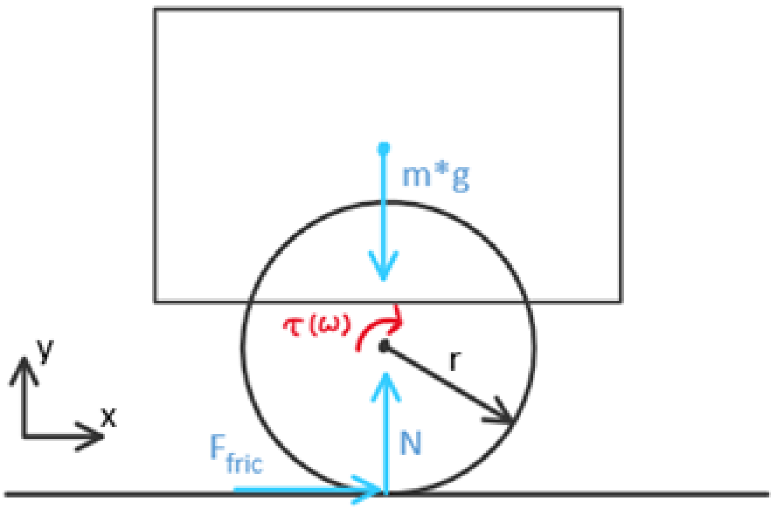


Figure 5.1: Free body diagram of the vehicle

The friction force F_{fric} can be quantified as:

$$F_{fric} = \frac{\tau}{r} \tag{5.1}$$

In line with Newton's second law, the acceleration of the vehicle is:

$$\sum F_x = F_{fric} = m \cdot a \Leftrightarrow a_x = \frac{F_{fric}}{m} \tag{5.2}$$

A brushed DC motor can be characterized as the torque being inversely proportional to the angular velocity. With the maximum torque, being the stall torque τ_s and the maximum angular velocity, being the no load velocity ω_n

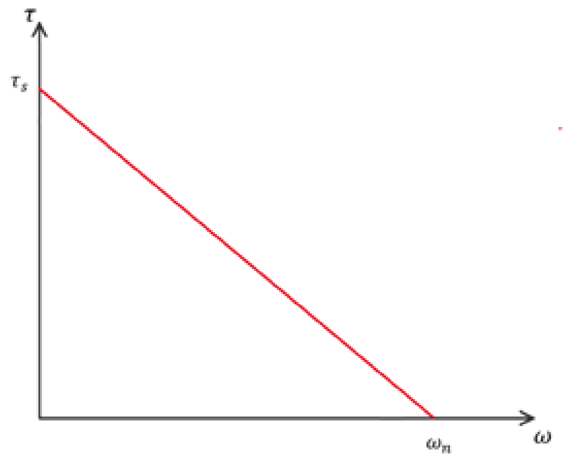


Figure 5.2: Torque/velocity relation of a brushed DC motor.

$\tau(\omega)$ can be characterized as:

$$\tau(\omega) = \tau_s - \frac{\tau_s}{\omega_n} \cdot \omega \tag{5.3}$$

The relation between the angular velocity of the wheel and the velocity of the vehicle is:

$$v = \omega \cdot r \Leftrightarrow \omega = \frac{v}{r} \tag{5.4}$$

Substituting (5.4) into (5.3) the relation between the output torque and the vehicle velocity is defined:

$$\tau(v) = \tau_s - \frac{\tau_s}{\omega_n \cdot r} \cdot v \tag{5.5}$$

Deviding (5.5) by r, the relation can be set equal to (5.1):

$$F_{fric}(v) = \frac{\tau(v)}{r} = \frac{\tau_s}{r} - \frac{\tau_s}{\omega_n \cdot r^2} \cdot v \tag{5.6}$$

Then deviding (5.6) by m, it can be set equal to (5.2):

$$a_x(v) = \frac{F_{fric}(v)}{m} = \frac{\tau_s}{mr} - \frac{\tau_s}{\omega_n \cdot mr^2} \cdot v \tag{5.7}$$

Since the velocity is the derivative of the acceleration ($\frac{dv}{dt} = a$), a differential equation can now be created:

$$\frac{dv}{dt} = \frac{\tau_s}{mr} - \frac{\tau_s}{\omega_n \cdot mr^2} \cdot v \tag{5.8}$$

Solving the differential equation (5.8) with MATLAB gives:

$$v(t) = \omega_n \cdot r(1 - e^{-\frac{t \cdot \tau_s}{m \cdot \omega_n * r^2}}) \tag{5.9}$$

Since there were two identical salvageable motors found at the university, the calculations were now specified towards those motors. The size of the caster wheels and that the motors would be mounted on the bottom plate concluded in a wheel diameter of 190mm. For the mass, it was concluded that the water for extinguishing would be the biggest contribution, since the size of the water container was specified to be around 5L. The total mass of the construction was estimated not to exceed 10kg, which meant the mass that was calculated with is 10kg. The stall torque (τ_s) was found on the datasheet of the salvaged motors, 3000 g/cm [6]. The found value was then converted to Nm. To convert from g/cm to Nm, one needs to multiply the g/cm with $9.80665 \cdot 10^{-5}$.

$$3000 \cdot 9.8065 \cdot 10^{-5} = 0.2941995Nm \tag{5.10}$$

Since the prototype will be driven by two motors, the torque needs to be doubled.

$$0.2941995 \cdot 2 = 0.588399 \tag{5.11}$$

Max output speed(ω_n)was also found on the datasheet of the motor, there it was stated to be 144 rpm, for the calculations the value was needed in rad/s, to convert from rpm to rad/s, one needs to multiply the value with 0.10472.

$$144 \cdot 0.10472 = 15.07968\frac{rad}{s} \tag{5.12}$$

To find the maximum velocity and acceleration of the prototype, the two found functions were graphed (5.3), with the correct r, τ_s and ω_n.

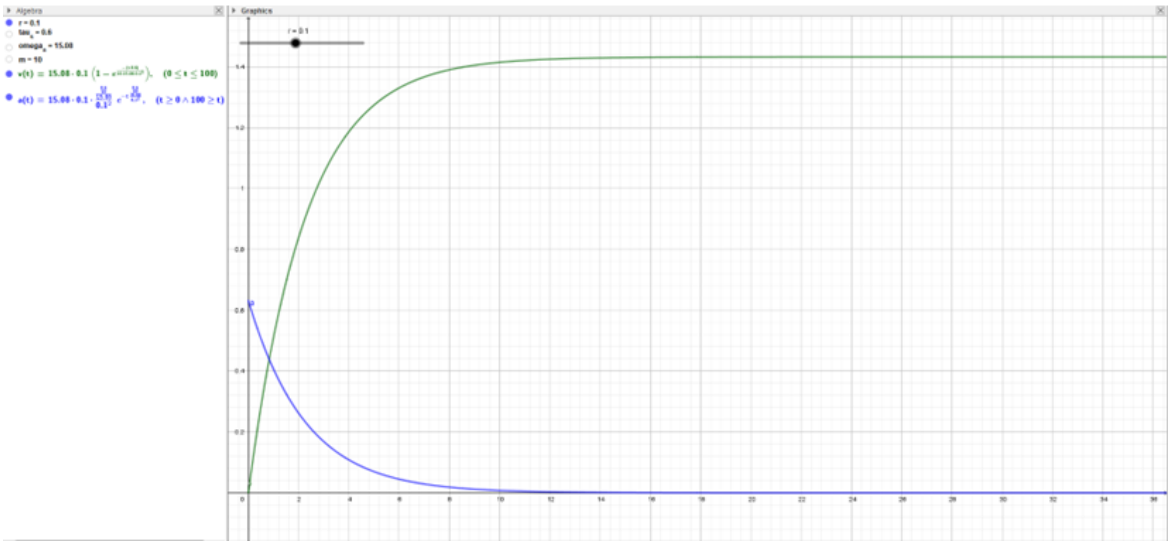


Figure 5.3: Graph of the velocity and acceleration

To find the max acceleration, one needs a(0). To calculate a(0), one needs to find a(t) first, a(t) can be found by differentiating v(t) with respect to t and then t=0:

$$v(t) = 15.08 \cdot 0.095(1 - e^{-\frac{t \cdot 0.6}{10 \cdot 15.08 \cdot 0.095^2}}) \tag{5.13}$$

$$v'(t) = \frac{0.6315789}{2.718281828459^{0.440862 \cdot t}} \tag{5.14}$$

$$v'(0) = \frac{0.6315789}{2.718281828459^{0.440862 \cdot 0}} = 0.6315789 \tag{5.15}$$

That means the max acceleration of the motor is $0.63\frac{m}{s^2}$.
Since the velocity has a max value and nears a horizontal asymptote, the max velocity can be found by setting t equal to a large number.

$$v(100) = 1.4326 \tag{5.16}$$

That means the maximum velocity is $1.43\frac{m}{s}$.
To find the time when the prototype will reach its maximum velocity, one can look at the graphs, and see at which point in time they both flatten out. One can then see that both graphs have flattened out at t=12.

$$v(12) = 1.42538 \tag{5.17}$$

By calculating v(12) one can see that the velocity at t=12, is also $1.43\frac{m}{s}$.

5.2 Water system

Researching the different ideas of a possible designs, limited to our morphology analysis, still left a lot of possible options regarding the mechanical construction, and its overall functionality. Firstly, the most urgent task at hand, was to design the water-spraying system, including the tank, pump, hose, and nozzle. To get a better idea of the different possibilities, further calculations were needed, to find the optimal design for the task.

5.2.1 Selection of waterspraying system

It was originally thought that a misting nozzle with droplets from 0.1 to 0.3mm were best suited for extinguishing fires. For further reliable information, consulting the head of the fire department seemed like a great opportunity. Through research, it was proven a fan nozzle would a better choice, as the small droplets from a misting nozzle would vaporize, before hitting the material on fire. The initial idea was to suppress the fire, by increasing the humidity, and lowering the surrounding temperature, which would have been optimal with a misting nozzle, as the droplets are easily vaporized. After consulting the fire department, they endorsed the idea, but recommended using lager droplet sizes. This was suggested, as using larger water droplets seemed more ideal for extinguishing fires, as smaller droplets otherwise would evaporate before hitting the initial material needed to be cooled down, in order to stop the fire. Therefore, working with larger droplets in the 2-3mm spectrum is better suited. The droplet size recommended, is of that equal to a fan nozzle. As the hole at the end of a fan nozzle is significantly bigger than for a misting nozzle, this also means that the amount of water able to move through the nozzle increases, hence increasing the flowrate. As a result, less pressure is needed, in order to move a larger scale of water; making it easier and more affordable to design a pump circuit.

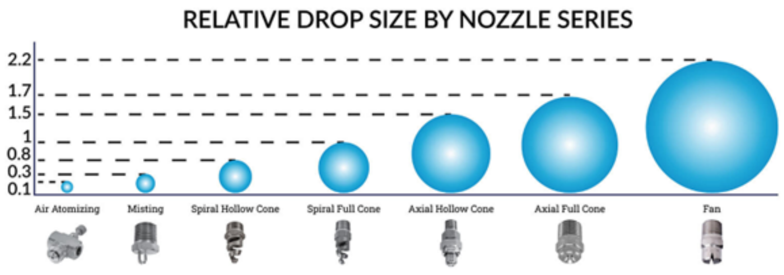


Figure 5.4: Dropsize of the different nozzles[7]

Aiming for around 3 minutes of working time, the right combination of tank capacity, nozzle, and pump must be chosen. Looking at the datasheets given for a series of fan nozzles [8] working with any nozzle from the range of NF01 to NF08 requires a relative low pressure, resulting in a more affordable pump. Researching into pumps, acquiring a water pressure pump, proved too expensive for the budget, as a higher pressure resulted in an increasing prize. Therefore, looking into fuel pumps proved to be a better option, as they are cheaper and capable of delivering a higher pressure at a lower cost. Looking into 458F0072 RIDEX fuel pump, capable of moving 90 litre of water an hour, with a max pressure of 4 bar at a cost of merely 129 DKK, designing the water system from this pump, would allow a great range of diversity in nozzles, capable of working with all the nozzles from the selected list, at a low price. As the pump can deliver 1.5 litre per minute, opting for a 5-litre tank would result in more than 3 minutes of total working time, while also reducing the total weight of the system as much as possible, therefore putting less strain on the motors.

5.2.2 Water tank and pump

To reduce weight and complexity of the water containing vessel, a plastic storage box would fit perfectly onto the bottom plate. The fuel pump must be mostly submerged into the liquid container, as one is not supposed to attach a hose to the inlet of the pump. That fact meant that the pump had to be mounted inside the water tank, without the electronics, that are mounted near the outlet of the pump, being near water to eliminate the possibility to damage the electronics by them coming in contact with water. A holder therefor needed to be designed to hold the pump upright on the underside of the lid of the box. The box lid would then later be modified by adding some holes to accompany for the hose and electronics. The holder was designed to be a pressfit around the pump to make it as water resistant as possible. Since the part was manufactured by 3D-printing, the wanted pressfit was not quite achieved, since the pump ended up being loose inside the holder. The problem was then solved by adding some caulk to the pump, and then inside of the holder, which made it water resistant.
To get the best weight distribution the water tank was mounted between the two motors.

5.2.3 Testing

Testing the first prototype of the water spraying system, water was leaking from both ends of the hose. In order to account for the water leaking, hose clamps were added in both ends. After another test it was found out that the hose clamps weren't clamping evenly around the hose, which resulted in leaking again. The hose clamps were then exchanged for twisting wire. The wire significantly improved the problem, but water was still spilling from the connection point between the pump and the hose. This seemed to be caused by the increasing pressure inside the hose, leading to the leak. In order to decrease the pressure, increasing the hole of the nozzle lead to less water would building up inside the hose, fixing the leaking problem; at the cost of a spray spread of 90 degrees.

5.2.4 Nozzle holder

The nozzle needed to be mounted in the front of the vehicle to ensure spraying at the wanted angle. A mount therefore had to be designed. Since the nozzle that was used in the end had a hexagonal shape in the middle, the holder was designed to clamp to the nozzle in front of and behind the hexagonal part. The holder is needed to ensure that the nozzle is held at the same height and angle while operating. Refer to appendix 8 for NX files.

6. Electronics design

6.1 Microphone bandpass filter

In the planning phase of the project it was decided to utilize microphones in order to decrease a potential search area and hone in on the fire, by listening for fire alarms. To achieve this it was neccesary to design an amplifier and filter circuit for the microphones. A bandpass filter was found to be suitable for this application, since it allowed for the isolation of a specific frequency.
The following sections goes through the designing of filter and amplifier circuitry for 1 microphone. However the physical design has 4 microphones to allow for directional listening, therefore the designs in the following sections are duplicated in the final PCB design.

6.1.1 Filter design

Firstly a mid frequency for the filter was choosen at 3.5kHz, since most residential smoke detectors were found to be in the range of 3000 to 4000Hz [9, pg.5].
Because it was preferable to isolate the 3.5kHz signal as much as possible, it was found that a high Q-factor and a higher order filter would be preferable. For this reason, the design methods described in "Op Amps For Everyone" [10, pg.230] seemed applicable. The filter will therefore be a fourth-order bandpass filter using the staggered tuning method. Firstly a bessel filter type was choosen, although in retrospective a Tschebyscheff filter type would have been ideal, due to a higher gain rolloff. Using the values from table 16.4 in "Op Amps For Everyone" [10, pg.231] the following values can be stated:

$$a_1 = 1.3617 \tag{6.1}$$

$$b_1 = 0.6180 \tag{6.2}$$

$$\alpha = 1.0032 \tag{6.3}$$

$$f_m = 3.5kHz \tag{6.4}$$

With a Q-factor of 10, which equates to a bandwidth of: $Q = \frac{f_m}{B} \Rightarrow B = 350Hz$ which is found adequate.
Filter gain (A_m) is also set at 1, due to difficulties during initial testing.

6.1.2 Operational amplifier requirements

In order to determine if the OPA340 operational amplifier already on hand is suitable for this filter circuit, it is neccesary to look at the unity-gain bandwidth, open-loop gain and slew rate requirements.

The unity-gain bandwidth requirement is found by use of the following formula[10]:

$$f_T = 100 \cdot Gain \cdot \frac{f_c}{a_i} \sqrt{\frac{Q_i^2 - 0.5}{Q_i^2 - 0.25}} \tag{6.5}$$

However this first requires the individual pole quality, which is the same for both partial filters:

$$Q_i = Q \cdot \frac{(1 + \alpha^2)\, b_1}{\alpha \cdot a_1} \Rightarrow 10 \cdot \frac{(1 + 1.0032^2) \cdot 0.6180}{1.0032 \cdot 1.3617} \approx 9.0769 \tag{6.6}$$

Using equation 6.5 the unity-gain bandwidth requirement is then found:

$$f_T = 100 \cdot 1 \cdot \frac{3.5kHz}{1.3617} \sqrt{\frac{9.0769^2 - 0.5}{9.0769^2 - 0.25}} \approx 0.25664MHz \tag{6.7}$$

With a bandwidth requirement of only 0.26MHz, the filter is well within the specifications of the OPA340 at 5.5MHz supplied by its datasheet[11].

The slew rate is found by use of formula 6.8[10, pg.251]:

$$SR = \pi \cdot V_{pp} \cdot f_C \tag{6.8}$$

The expected peak to peak input voltage is 5V:

$$SR = \pi \cdot 5V \cdot 3.5kHz \approx 0.05498 \frac{V}{\mu s} \tag{6.9}$$

This value also falls well within the max slew rate of $6\frac{V}{\mu s}$ specified in the OPA340 datasheet[11]

Lastly the open-loop gain should be 40dB above the peak gain[10, pg.250]. Knowing the filter produces a max gain of 0dB and looking at figure 6.1 from the OPA340 datasheet, it is clearly visible that the open-loop gain of the operational amplifier is more than 60dB above the peak gain of the filter.

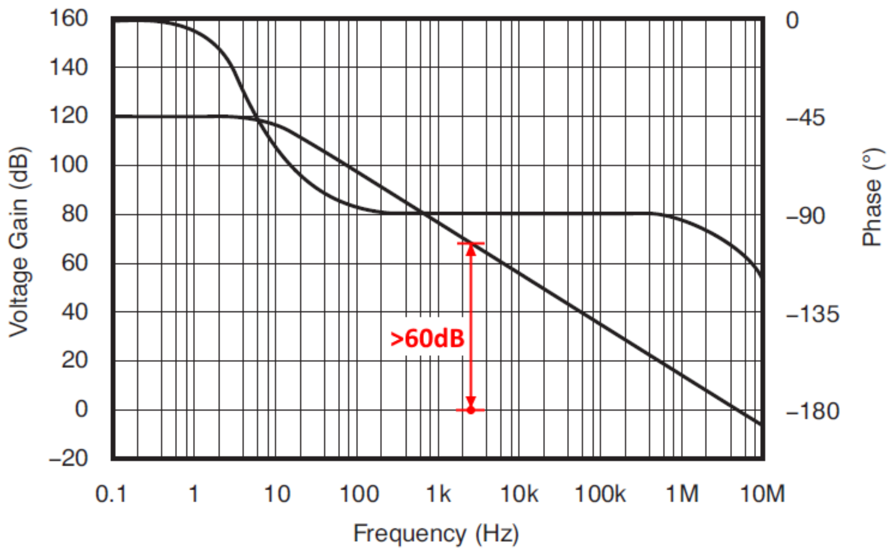


Figure 6.1: OPA340 Open-loop gain/phase vs frequency [11]

6.1.3 Filter dimensioning

With preliminary design and requirements handled, it was now possible to dimension the filter using the multiple feedback bandpass (MFB) filter typology, refer to figure 6.2. The MFB design utilizes two capacitors of same value, which are arbitrarily specified at: $C = 10nF$.

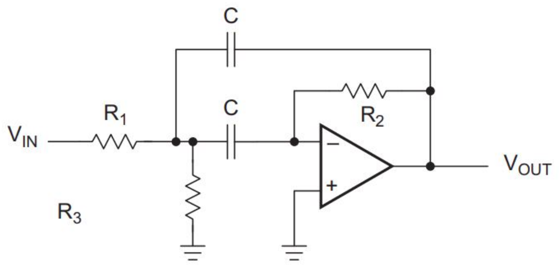


Figure 6.2: MFB typology [10, pg.229]

To obtain resistor values it was neccesary to calculate the mid frequency of the partial filters by use of formulas 6.10 and 6.11 [10, pg.231].

$$f_{m1} = \frac{f_m}{\alpha} \Rightarrow \frac{3.5kHz}{1.0032} = 3488.8Hz \tag{6.10}$$

$$f_{m2} = f_m \cdot \alpha \Rightarrow 3.5kHz \cdot 1.0032 = 3511.2Hz \tag{6.11}$$

As well as the individual gain at the partial mid frequencies using formula 6.12 [10, pg.231], which is equal for both partial filters

$$A_{mi} = \frac{Q_i}{Q} \cdot \sqrt{\frac{A_m}{b_1}} \Rightarrow \frac{9.0769}{10} \cdot \sqrt{\frac{1}{0.618}} \approx 1.155 \tag{6.12}$$

The resistor values for both filter stages are then determined by use of formulas 6.13, 6.14 and 6.15 [10, pg.232].

Filter 1:

$$R_{21} = \frac{Q_i}{\pi f_{m1} C} \Leftrightarrow R_{21} = 85268.349\Omega = 82k\Omega \tag{6.13}$$

$$R_{11} = \frac{R_{21}}{-2A_{mi}} \Leftrightarrow R_{11} = 36905.774\Omega = 36k\Omega \tag{6.14}$$

$$R_{31} = \frac{-A_{mi}R_{11}}{2Q_i^2 + A_{mi}} \Leftrightarrow R_{31} = 256.674\Omega = 270\Omega \tag{6.15}$$

Filter 2:

$$R_{22} = \frac{Q_i}{\pi f_{m2} C} \Leftrightarrow R_{22} = 80000.346\Omega = 82k\Omega \tag{6.16}$$

$$R_{12} = \frac{R_{22}}{-2A_{mi}} \Leftrightarrow R_{12} = 34625.682\Omega = 36k\Omega \tag{6.17}$$

$$R_{32} = \frac{-A_{mi}R_{12}}{2Q_i^2 + A_{mi}} \Leftrightarrow R_{32} = 240.816\Omega = 240\Omega \tag{6.18}$$

Resistor values were assumed to closest possible value available in university laboratories.

6.1.4 Filter simulation

To ensure viability of the bandpass filter design, it was simulated in LTspice, refer to figure 6.3 and 6.4. Each partial filter stage uses 10nF capacitors and the resistor values calculated previously. additionally a 2.5V bias was added in order to avoid clipping, since the operational amplifier will have a 0 to 5V supply and the input signal centered around 0V. Subsequently it was tested on a breadboard to finalize it for the PCB design schematic. The LTspice file can be found in Appendix 4.

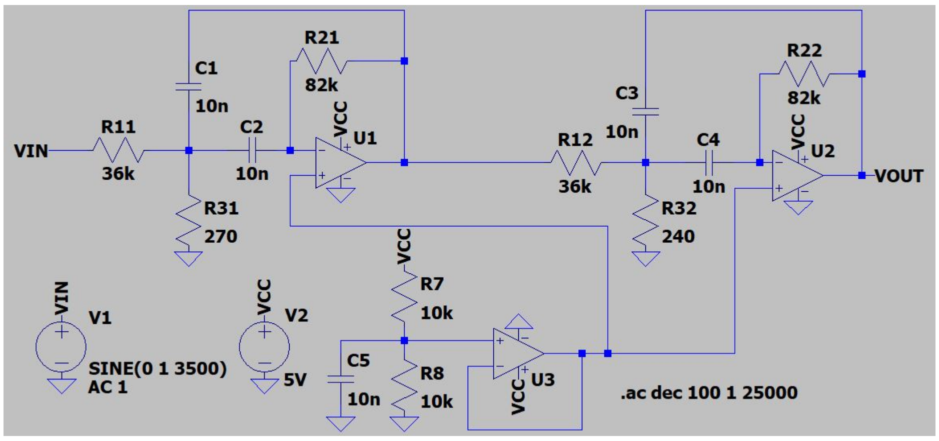


Figure 6.3: Bandpass filter circuit in LTspice

The frequency respone of the simulated bandpass filter is seen on figure 6.4, where it is clearly visible that its centered around 3.5kHz and has high attenuation on either side of it.

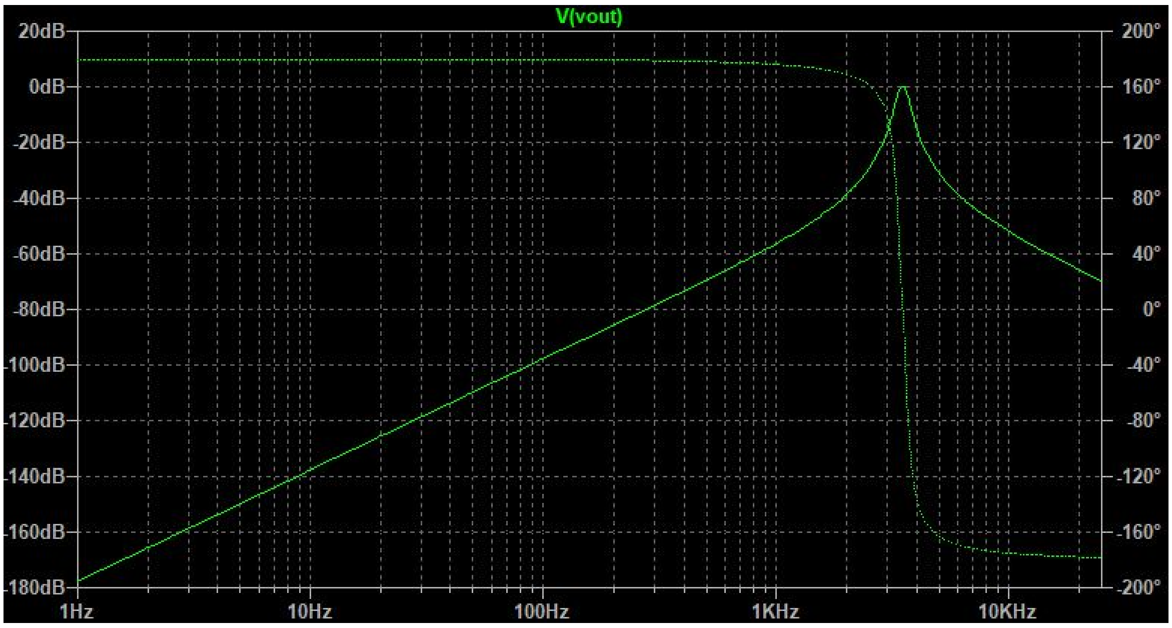


Figure 6.4: Frequency response of bandpass filter simulation in LTspice

6.2 Microphone amplifier

The bandpass filter designed in the previous section only attenuates signals and therefore an amplifier stage is neccesary to provide amplification of the microphones signals.

6.2.1 Amplifier design and simulation

The design of this amplifier stage was not the main focus and was therefore heavily inspired [12] by another design. Referring to figure 6.5 the R2 resistor determines the gain of the amplifier and was in this case experimentally selected during subsequent testing on breadboard. The amplifier is also DC biased with 2.5V, to avoid clipping since the supply voltage is 0 to 5V. See appendix 3 for LTspice file.

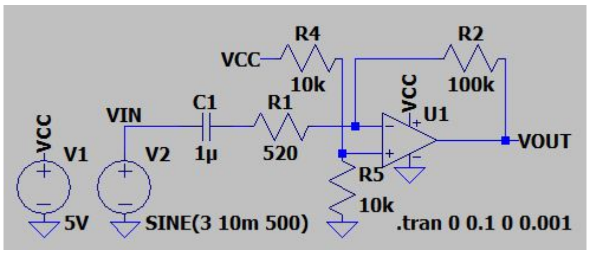


Figure 6.5: Microphone amplifier circuit in LTspice

Figure 6.6 shows the amplification of a 10mV signal with 3V bias in LTspice, with the V2 voltage source representing the microphone in figure 6.5.

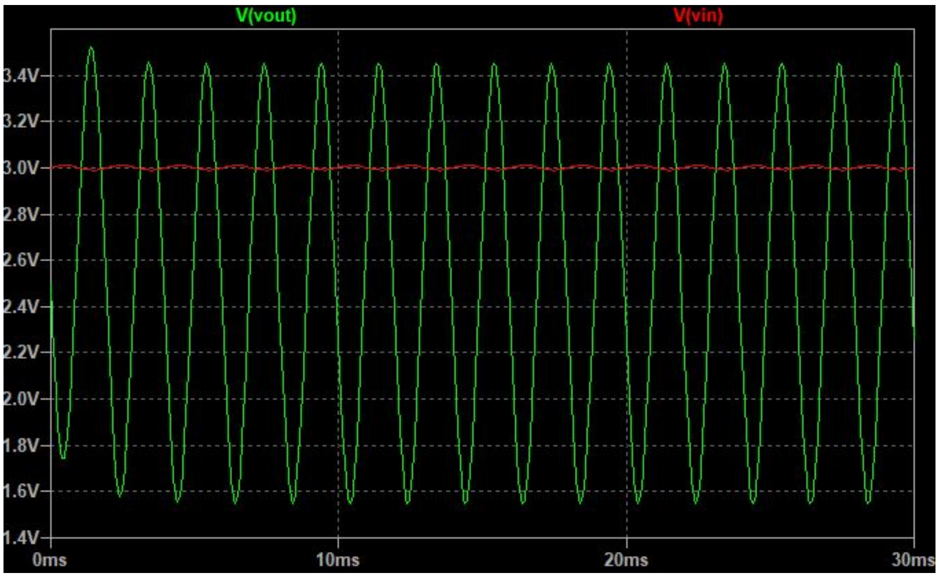


Figure 6.6: Microphone amplifier simulation

6.3 Microphone circuit characteristics

With both amplifier and filter circuit designed, simulated and verified on breadboard, it was possible to finalize the microphone circuitry for PCB design. 1 of the 4 microphone circuits is seen on figure 6.7. A 5.1k resistor is added before the amplifier stage to enable use of the condensor microphone. Furthermore all 3 operational amplifiers make use of the same buffered 2.5V reference. See appendix 5 for full schematics.

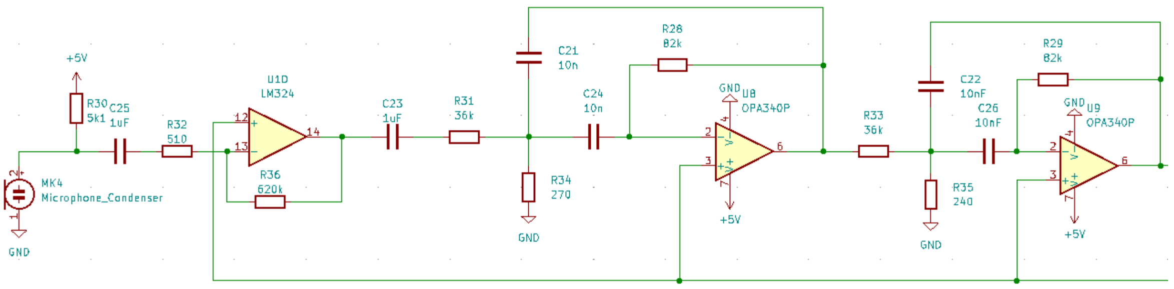


Figure 6.7: Finalized schematic of one microphone circuit

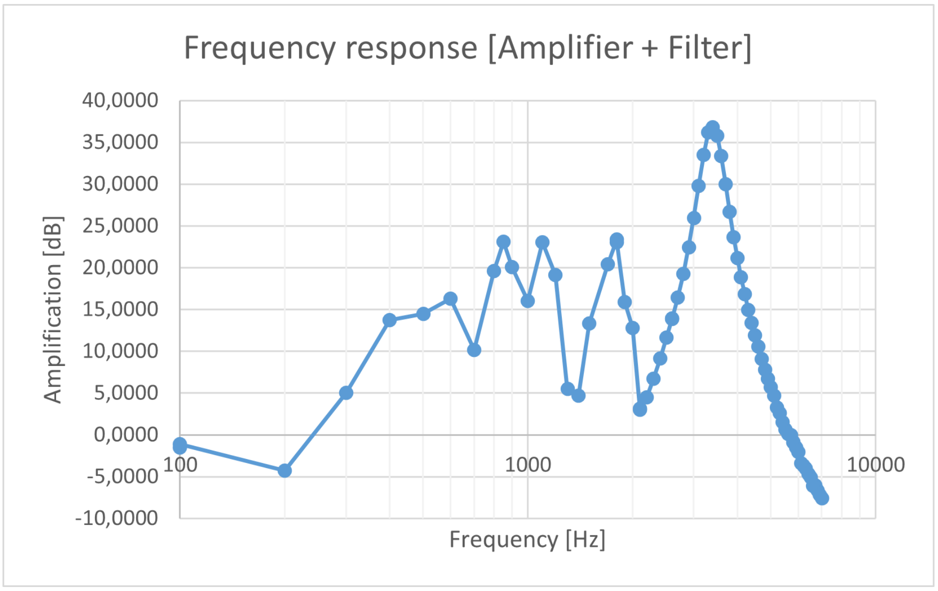


Figure 6.8: Frequency response of amplifier and filter on PCB

Testing the frequency response of the microphone filter and amplifier resulted in the bode plot seen above in figure 6.8. In the frequencies lower than the pass-band, some interesting spikes can be seen, on further inspection these spikes peak at whole fractions (1/2, 1/3, 1/4 etc.) of the mid frequency. The hypothesis as to the cause, is that the multiple feedback bandpass filter, have some kind of feedback at these frequencies as well, causing resonance. This hypothesis is further supported by the fact that the output of the filter, at these fractional input frequencies, is oscillating at the mid frequency of 3500 Hz. Since the feedback at the lower frequencies is amplified less than 10 dB than the mid frequency of the filter, it is not expected to cause any issues.

6.4 IR-Array and filter

While doing the morphology analysis IR phototransistors was considered for fire detection, primarily because they are very cheap, and rather easy to obtain a signal from. After discovering a master thesis on the subject [13], they were deemed an acceptable solution considering our limited budget.
These cheap phototransistors have a limited detection angle (commonly between 10-30 degrees) and therefore multiple phototransistors will be used to obtain a field of view of approximately 60 degrees in the horizontal and vertical directions. The field of view will however be a little bit limited at the borders of the field of view, due to the directional sensitivity of the phototransistors.

6.4.1 IR-Array

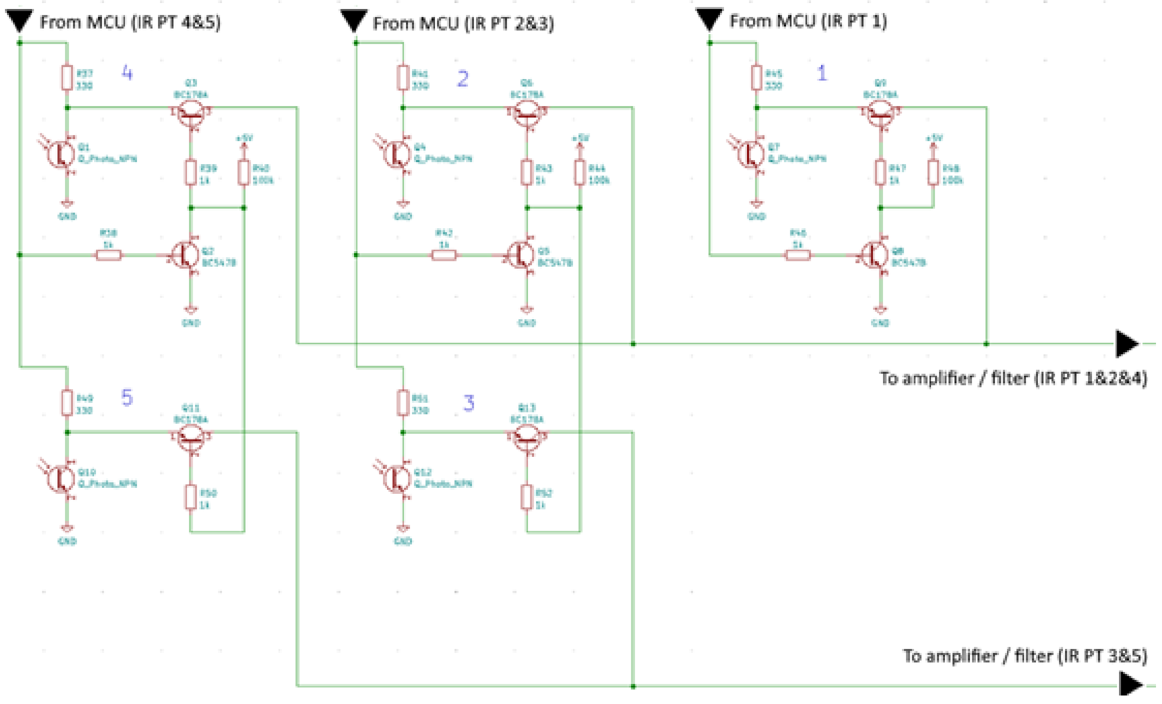

From MCU (IR PT 4&5)
From MCU (IR PT 2&3)
From MCU (IR PT 1)
4
2
1
5
3
To amplifier / filter (IR PT 1&2&4)
To amplifier / filter (IR PT 3&5)

Figure 6.9: Phototransistor array

As the Arduino Nano only have 8 ADC channels, it was not possible to connect all 5 IR phototransistors directly to the microcontroller (As the microphones use 4 channels). To circumvent this the phototransistors was arranged in a matrix/array and a transistor circuit was made to switch between them, see the circuit in figure 6.9. See appendix 5 for full schematics.

The circuit consists first of all of the phototransistors, the phototransistors used are OSRAM SFH 313 FA-2/3, which is a phototransistor that is sensitive in the near infrared spectrum from 740-1080nm, with sensitivity peaking at 870nm [14].
The collector on the phototransistor is connected to a current limiting resistor (e.g. R37 in the schematic), limiting the current through the phototransistor to roughly 15mA:

$$\frac{5V}{15mA} = 333\Omega \approx 330\Omega \tag{6.19}$$

The collector of the phototransistor is also connected to the emitter of a PNP-transistor, that is used as a switch. The reason for this added "switch" is to avoid the phototransistors of a "row" being connected in parallel. Since the PNP-transistor needs a current flowing from the emitter to the base in order to saturate the junction, a NPN-transistor (along with a pull up resistor) is used to create a simple inverter, so the

PNP transistor can be switched using the same output from the MCU that is used to power the phototransistors.

6.4.2 Phototransistor amplifier

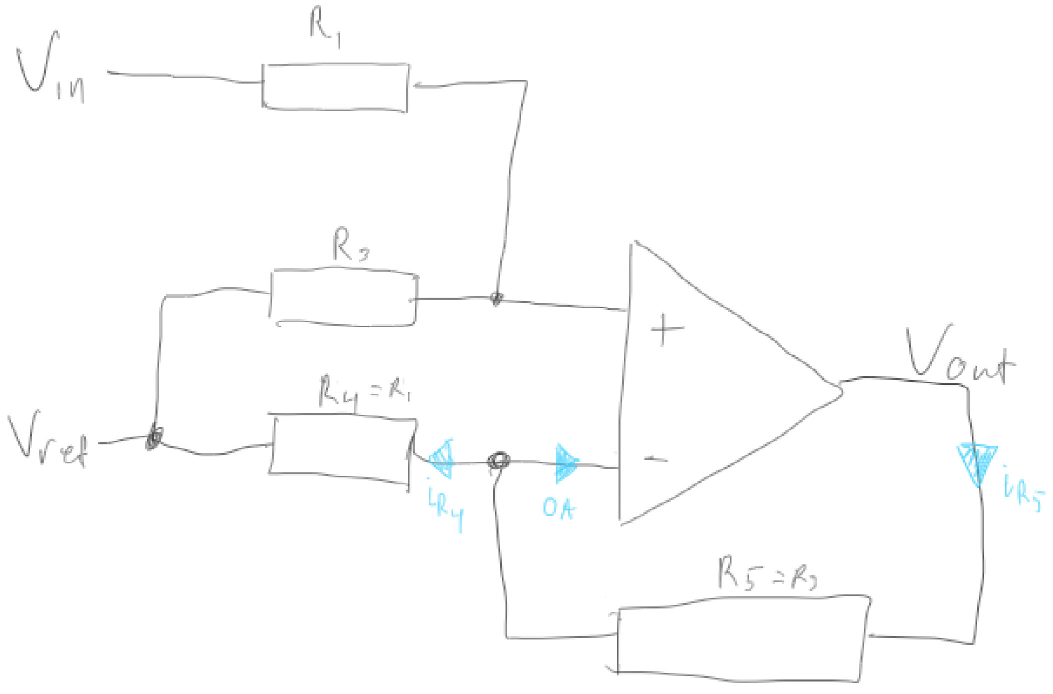


Figure 6.10: Sketch of phototransistor amplifier circuit

To find the output voltage of the non-inverting amplifier, it was first stated, since it is non-inverting $V_+ = V_-$. There was then found a formula for the two V's.
V_-:

$$V_{R_5} = R_5 \cdot i_{R_5} \tag{6.20}$$

$$V_{R_4} = R_4 \cdot i_{R_4} \tag{6.21}$$

Since everything is in series it can be said that:

$$i = i_{R_5} = i_{R_4} \tag{6.22}$$

Because of Kirchhoff's voltage law the conclusion is:

$$V_{R_5} + V_{R_4} = V_{out} - V_{ref} \tag{6.23}$$

Formulas (6.20) and (6.21) are now substituted into formula (6.23):

$$V_{out} - V_{ref} = i(R_4 + R_5) \tag{6.24}$$

i is now isolated:

$$i = \frac{V_{out} - V_{ref}}{R_4 + R_5} \tag{6.25}$$

Since it is known that $V_{R_4} + V_{ref} = V_{out}$, V_{R_4} can be changed with the help of formula (6.20) and (6.22):

$$V_- = R_4 \cdot i + V_{ref} \tag{6.26}$$

i is now isolated in the found formula:

$$i = \frac{V_- - V_{ref}}{R_4} \tag{6.27}$$

The two formulas for i are now set equal to one another:

$$\frac{V_{out} - V_{ref}}{R_4 + R_5} = \frac{V_- - V_{ref}}{R_4} \tag{6.28}$$

V_- is now isolated:

$$V_- = \frac{R_4}{R_4 + R_5} \cdot (V_{out} - V_{ref}) - V_{ref} \tag{6.29}$$

V_+ is now found by using superposition:

$$V_+(V_{in}) = V_{in} \cdot \frac{R_3}{R_1 + R_3} \tag{6.30}$$

$$V_+(V_{ref}) = V_{ref} \cdot \frac{R_1}{R_1 + R_3} \tag{6.31}$$

To find V_+ the two found formulas for V_+ are combined by addition:

$$V_+ = V_{ref} \cdot \frac{R_1}{R_1 + R_3} + V_{in} \cdot \frac{R_3}{R_1 + R_3} \tag{6.32}$$

The two found formulas for V were then set the two equal to another and V_{out} was isolated:

$$\frac{R_4}{R_4 + R_5} \cdot (V_{out} - V_{ref}) - V_{ref} = V_{ref} \cdot \frac{R_1}{R_1 + R_3} + V_{in} \cdot \frac{R_3}{R_1 + R_3} \tag{6.33}$$

To make it easier in the calculations it was decided that:

$$R_1 = R_4 \tag{6.34}$$

$$R_3 = R_5 \tag{6.35}$$

V_{out} can be isolated to be:

$$V_{out} = V_{in} \cdot \frac{R_3}{R_1} + V_{ref} \tag{6.36}$$

Since the amplifier has to amplify the voltage by 100 times, it can be said that:

$$\frac{R_3}{R_1} = 100 \Leftrightarrow R_3 = 100R_1 \tag{6.37}$$

6.4.3 Issues with phototransistors

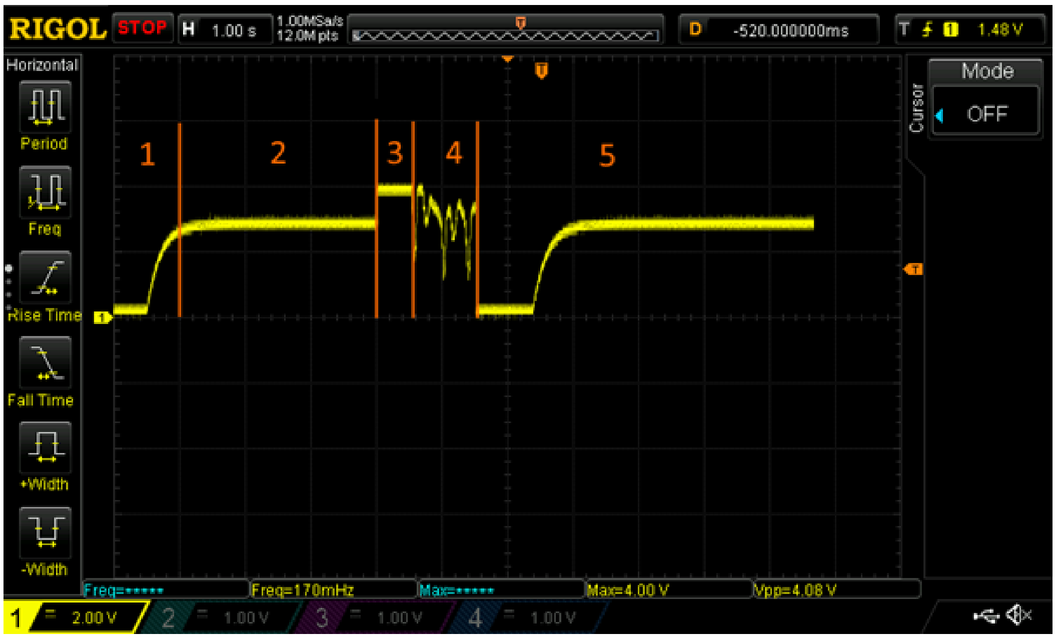


Figure 6.11: Screenshot of oscilloscope

While the phototransistor array was being tested some issues arose, as the values that were displayed on the oscilloscope did not make sense and seemed constant. To investigate these issues a single read cycle on a single phototransistor was measured on the scope. The issues immediately were identifiable as the high pass filter not being able to stabilize when the bias voltage of the array changed (inputs being shut on and off). Further delays were then introduced in the code to let the circuit stabilize before any measurements were made, which resulted in the oscilloscope picture in (6.11). The

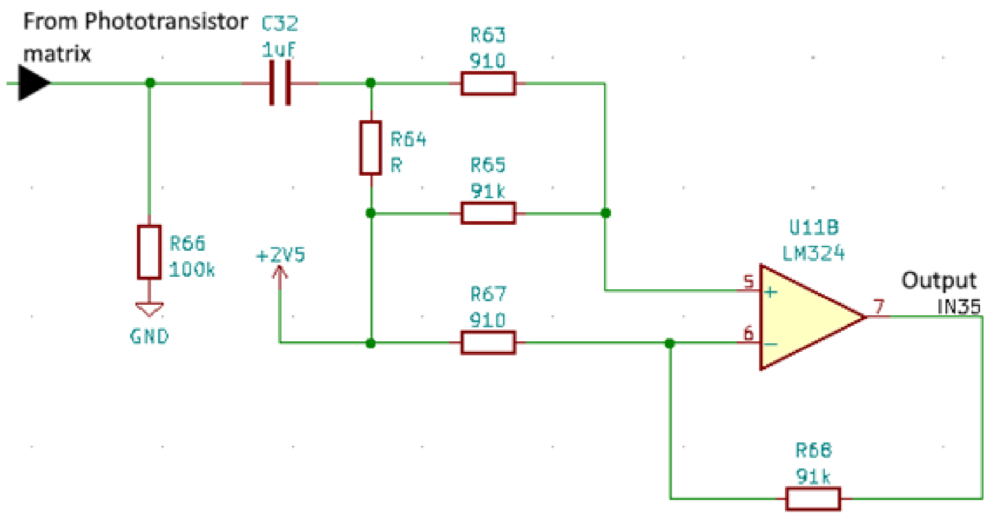


Figure 6.12: Diagram of the high pass filter along with amplifier

following paragraphs will dive into what is happening at different points in the cycle.

Point 1:

The microcontroller has just been reset, which means the 2.5V reference voltage have been held at ground level, before rising again (Almost instantaneously). This results in charging capacitor 'C32' through the resistors 'R63' and 'R65' (R64 is NC), which is a RC circuit with time constant:

$$\tau = RC = (91k\Omega \cdot 910\Omega) \cdot 1\mu\Omega = 0.09191s \tag{6.38}$$

The capacitor will be fully charged after 5 time constants have passed:

$$5\tau = 0.4596s \approx 0.5s \tag{6.39}$$

Which fits well with the oscilloscope readings.

Point 2:

Now the capacitor has been charged to 2.5V, which is "buffered" by the op-amp for the three second delay in code.

Point 3:

At this point the transistors in the phototransistor matrix is switched to connect one of the phototransistors to the capacitor 'C32'. It is known from experience from testing the phototransistors, that the voltage drop across them is usually around ≈ 4.5 volts. This changes the voltage drop across the capacitor and the circuit once again needs around .5 seconds to stabilize. During this period the output voltage of the circuit is around 4 volts. This is because of the limited output span of the LM324 op-amp (TODO insert reference to datasheet).

Point 4:
After the capacitor (or high pass filter) stabilizes, is starts to let the signal of a fire source being held in front of the phototransistor through (While filtering out the DC-bias, from e.g., sunlight). The signal is passed through for a total of approximately 1 second (Delay in code minus the .5 second charge time of the capacitor).

Point 5:
After a measurement have been made, the input to the array is shut off, and the array side of the capacitor is discharged to ground through the 100k resistor R66, while the amplifier side is being pulled to 2.5 volts. Since the ac-signal (and dc) is gone on the array side, the amplifier output stabilizes at 2.5 volts.

6.5 Ultrasonic sensors

In order to read values from three ultrasonic sensors all using timer/counter 1 of the ATmega328P, the following circuit was designed, see figure 6.13.

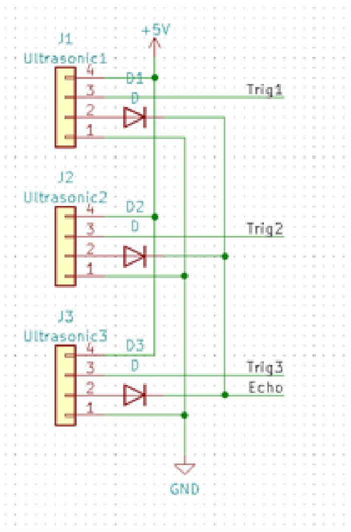


Figure 6.13: Schematic of ultrasonic sensor connections

In order to capture the signals on the echo pins using only one counter, the three echo pins are connected together (Through diodes to keep the sensors with interfering with each other) and connected to the input capture pin for timer/counter 1, PB0 [15].

The distance is represented by the length of a pulse on the echo pin [16], therefore the microcontroller needs to measure this. The required program flow for doing that has been drawn below in figure 6.14.

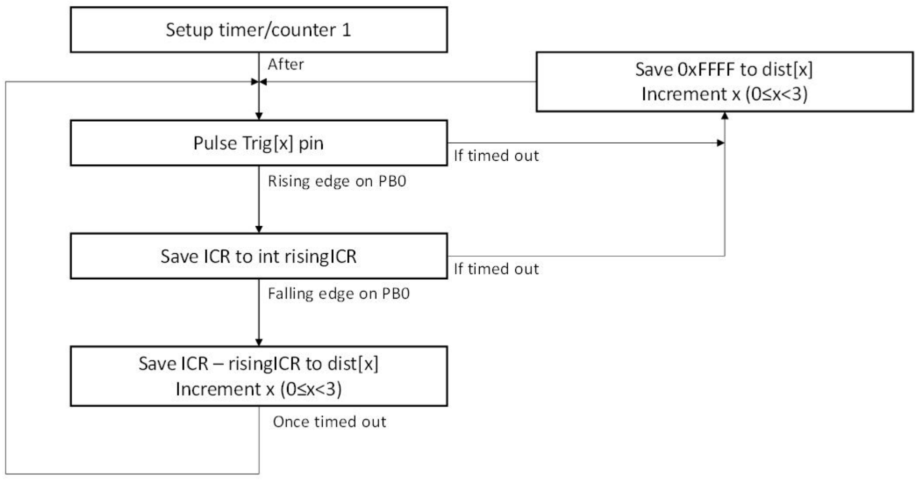


Figure 6.14: Basic flow of ultrasonic sensor measurements

The plan is to make the measurement only run in interrupts, so that it is set-and-forget once initialized.

6.6 PCB design

With the complexity of the circuit, soldering it up on perf- or stripboard was not an (time efficient) option, therefore a PCB was designed instead. Since the circuit is designed completely with through hole components, which are rather large, the PCB will be made as a single layer to save costs. The PCB was then laid out. A copper pour (connected to ground) was added as the last step.

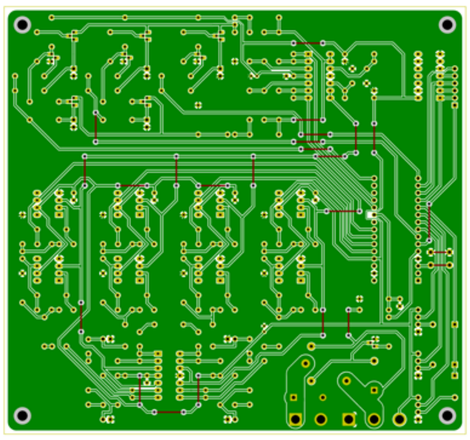


Figure 6.15: PCB layout for the robot (The red connections are jumper wires)

After running a DRC, the PCB was ordered and assembled. The finished result can be seen below. There was a bit of diagnosing while assembling the PCB, because the lack of a solder mask combined with small pads and traces, created some shorts while soldering.

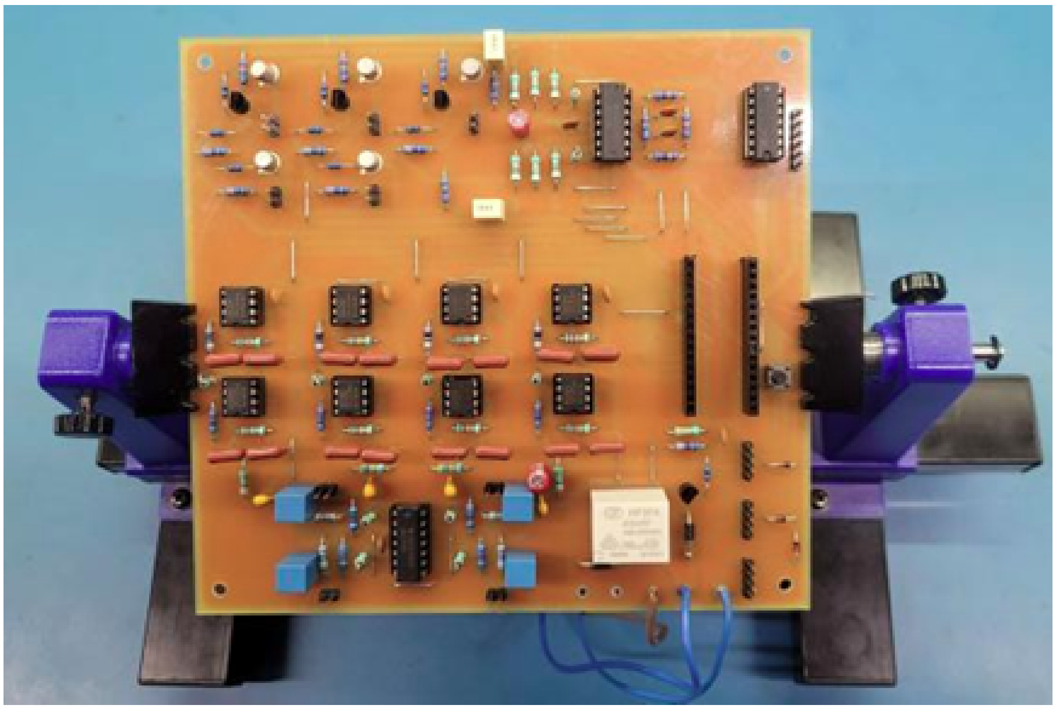


Figure 6.16: Finalized PCB with components

7. Software design

The code for this project is written in C, and it includes 4 files. Main.c is the main file, from where the other files are called, containing the state machine.

Idle.c is the file that activates the required pins and checks for distance with the ultrasonic sensor, and controls the motors. It also has the code for obstacle avoidance, depending on the input from the ultrasonic sensor, changes the speed and direction of the motors.

Pathfind_fire.c is the file that activates the required pins and checks for infrared light with the phototransistors. Once the fire is found, it also changes the speed of the motors to redirect the robot towards the fire and make it stop before it hits the fire. Once the robot is at a safe distance, it will activate the pump to spray water into the fire.

Pathfind_sound.c is the file that activates the required pins and checks for sound through the microphones. When the microphones detect sound, they give a voltage that then is compared in code, to check where the fire alarm is and drive towards it. This part of the code was not implemented in the final code, since some problems when combining all the codes made this code stop working and lack of time made it not possible to implement microphones in the last version of the project.

Moreover, a file named global.h is also included in this project, to be able to use and interchange values from variables through all the other files. This file defines constants and global variables to be used through all the project files. It defines different states for a state machine, variables to keep track of the current state, counters, and variables and ports for phototransistors, microphones, motors, ultrasonic sensors, and the pump. It also defines different ranges and timeouts for the ultrasonic sensors, and cycles for the motors.

```c
#ifndef GLOBAL_H
#define GLOBAL_H

//states for the state machine
typedef enum
{
    state_idle,
    state_pathfind_sound,
    state_pathfind_fire,
    state_extinguish,
    state_evacuate,
    state_shutdown,
}state;
//variabe to keep current state
int current_state;
```

../Code/global.h

The #ifndef and #define preprocessor directives at the top are used to prevent the contents of this header file from being included multiple times in a single source file. This can happen if the header file is included by multiple source files that are compiled together. The #endif at the end of the file marks the end of the preprocessor conditional block.

7.1 State machine (main.c)

```c
int main(void){
    //Call initialization functions in other files so all pins are
        settup
    idle_start();
    //pathfind_sound_start();
    pathfind_fire_start();
    //Start cycle counters
    run_idle_counter = 0;
    run_path_fire_counter = 0;
    //Set the initial state
    current_state = state_idle;
```

../Code/main.c

First the “idle_start()“ and “pathfind_sound_start()“ functions are called, that initialize all required pins and set registers for motors, ultrasonic sensors, photo transistors and the pump.
The cycle counters are also restarted to 0 and the initial state is set.

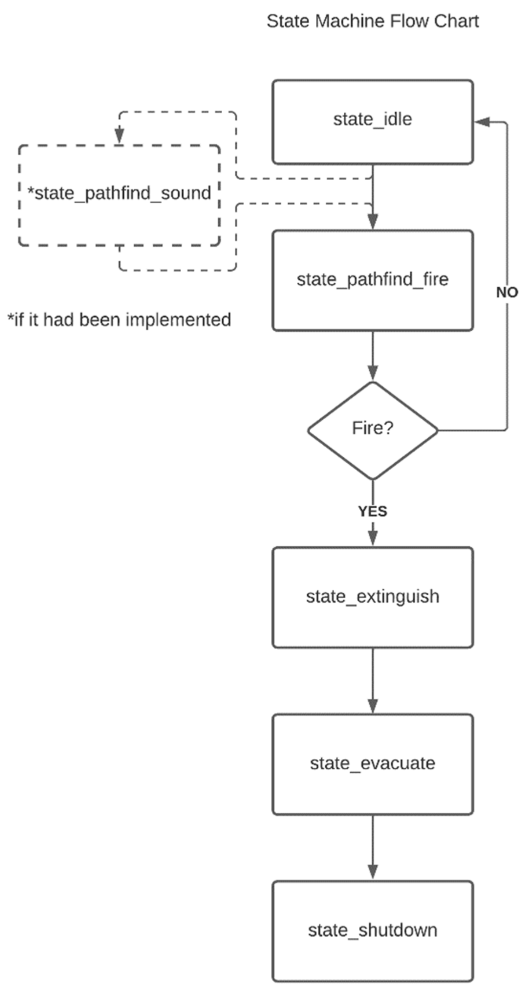


Figure 7.1: State machine flowchart

```
    while (1){
        //Define the state machine
        switch(current_state){
            case state_idle:
                idle();
                break;
            case state_pathfind_fire:
                pathfind_fire();
                break;
            case state_extinguish:
                extinguish();
                break;
            case state_evacuate:
                evacuate();
                break;
            case state_shutdown:
                break;
        }
    }
```

../Code/main.c

The main() function then enters a finite loop, where a switch case can be found. The switch is set up to change the current state of the program, current_state, and depending on the value of, current_state, the corresponding case will be called. When a case is changed, the program calls a function in another file. These functions will do their corresponding task and then when the task is finished, will change the value of current_state and break the loop, so the code goes back to the main file and changes the state, calling the corresponding case.

7.2 Obstacle avoidance & autonomous driving (idle.c)

This file is the responsible for doing the autonomous driving and obstacle avoidance. After the variables and arrays are defined, idle_start is started.

```
void idle_start(void){
    //Setting up pins for the motors
    DDRB |= (1 << DIRECTION_1 | 1 << DIRECTION_2 | 1 << PWM_2);
    DDRD |= (1 << PWM_1);

    //Setting up pins for ultrasonic sensors
    TRIG_DDR |= (1 << TRIG1_pin | 1 << TRIG2_pin | 1 << TRIG3_pin);

    //Initial motor directions
    PORTB &= ~(1<<DIRECTION_1 | 1<<DIRECTION_2);

    //Motor timer
    TCCR2A = 0xA3; //0b10100011
    TCCR2B = 0x05; //0b00000101

    //Call the ultrasonic funtion
    startUltrasonic();
}
```

../Code/idle.c

This function initializes the ultrasonic sensor, motor, and communication with the microcontroller. It sets the corresponding pins and registers to control the ultrasonic sensors and motors and initializes the timer that is used to control the motor's speed. Lastly it calls startUltrasonic function.

```c
    //Setting up pins for the motors
    DDRB |= (1 << DIRECTION_1 | 1 << DIRECTION_2 | 1 << PWM_2);
    DDRD |= (1 << PWM_1);
```

../Code/idle.c

These two lines, configure the pins on the Arduino that are used to control the motors. The "DDRB" and "DDRD" registers are the data direction registers for ports B and D, respectively. Setting the appropriate bits in these registers to 1 configures the corresponding pins as outputs. This allows the microcontroller to control the direction and speed of the motors. The "1 « DIRECTION_1 | 1 « DIRECTION_2 | 1 « PWM_2" part of the expression sets the bits corresponding to the "DIRECTION_1, DIRECTION_2, and PWM_2" macros to 1. The "1 « PWM_1" part of the expression sets the bit corresponding to the "PWM_1" macro to 1. These bits are then set in the "DDRB" and "DDRD" registers using the |= operator, which sets the corresponding bits to 1 without affecting the other bits.

```c
    TRIG_DDR |= (1 << TRIG1_pin | 1 << TRIG2_pin | 1 << TRIG3_pin);
```

../Code/idle.c

This line of code does the same thing as the two previous ones, but instead of setting the pins for the motors, sets the pins for the ultrasonic sensors.

```c
    PORTB &= ~(1<<DIRECTION_1 | 1<<DIRECTION_2);
```

../Code/idle.c

For the motors to go forward, the bit that controls the direction must be a 0, so this line of code sets a 0 to the port for the direction of the left and right motors.

```c
    //Motor timer
    TCCR2A = 0xA3; //0b10100011
    TCCR2B = 0x05; //0b00000101
```

../Code/idle.c

The "TCCR2A" and "TCCR2B" registers are control registers for timer/counter 2 on the AVR microcontroller. Setting the "TCCR2A" register to 0xA3 and the "TCCR2B" register to 0x05 configures the timer to use a prescaler of 64, and sets the timer mode

to fast PWM with a top value of 0xFF. This allows the timer to generate a PWM signal on the output compare pins (OC2A and OC2B) that can be used to control the speed of the motor. The PWM signal will have a frequency of 16 MHz / $(64 \cdot 256) = 976.5625$ Hz.
Lastly, the idle_start, calls startUltrasonic, this is done to initialize the interrupt used for the ultrasonic sensor.

```c
//Function for setting up the timer for distance measurement operation
void startUltrasonic(void){
    //Setting up waveform generation mode to CTC mode, with OCR1A as top
    //Enabling the ICNC1, input capture noise canceler
    //Setting a clk/64 prescaler, for a span of 262 ms and a resolution of 4 us
    //Setting up so the input capture interrupt is called on a rising edge
    TCCR1B |= (1 << WGM12 | 1 << ICNC1 | 1 << CS11 | 1 << CS10 | 1 << ICES1);

    //Setting OCR1AH for a 12 us cycle
    OCR1A = ULTRASONIC_pulse;

    //Enabling the Output compare A match and interrupt capture interrupt (Global interrupts still disabled)
    TIMSK1 |= (1 << OCIE1A | 1 << ICIE1);

    //Setting the trigger to high
    TRIG_port |= (1 << trigPins[ultrasonicIndex]);

    //Resetting the counter to 0
    TCNT1 = 0;

    //Enabling interrupts
    sei();
}
```

../Code/idle.c

This code sets up a timer to measure the distance using ultrasonic sensors. The WGM12 bit of TCCR1B register is set to enable the CTC mode, ICNC1 bit enables the input capture noise canceler, CS11 and CS10 bits enable the timer to be clocked by the F_CPU/64 frequency, ICES1 bit enables the input capture edge select (rising edge), OCR1A register is set to ULTRASONIC_pulse which sets the compare match value to 12us cycle, OCIE1A bit enables the Output compare A match interrupt and ICIE1 bit enables the input capture interrupt. The trigPins[ultrasonicIndex] is set to high to trigger the ultrasonic sensor and the timer counter is reset to 0. Finally, global interrupts are enabled.

```c
//This interrupt is called when the trigger pulse has been turned
    on for 12us
ISR(TIMER1_COMPA_vect){
    if (OCR1A == ULTRASONIC_pulse)
    {
        //Turning off trigger pins
        TRIG_port &= ~(1 << TRIG1_pin | 1 << TRIG2_pin | 1 <<
            TRIG3_pin);

        //Setting OCR1A for timout
        OCR1A = ULTRASONIC_timout;

        //Update pulseindex
        pulseIndex = ultrasonicIndex;
    }
    else if (OCR1A == ULTRASONIC_timout) //Moving on to next dist
        measurement
    {
        //If no measurement have been made in cycle
        if (pulseIndex == ultrasonicIndex)
        {
            //Saving an erroneous result in dist
            dist[ultrasonicIndex] = 0xFFFF;
```

../Code/idle.c

Once the interrupt is called, if the OCR1A register is set to the ULTRASONIC _pulse variable, the trigger pins are turned off, the OCR1A register is set to the ULTRASONIC_timout variable, and the pulseIndex is set to the ultrasonicIndex variable. If the OCR1A register is set to the ULTRASONIC_timout variable, the distance measurement will move onto the next measurement. If no measurement has been made in the cycle, an erroneous result will be saved in the dist array and the ultrasonicIndex variable will be incremented. The TCCR1B register is then set to trigger the interrupt on a rising edge, the OCR1A register is set to the ULTRASONIC_pulse variable, the trigger pin is set to high, and the counter is reset to 0.

```c
//Depending on the DistanceState, this interrupt either reacts on
   rising or falling edge
ISR(TIMER1_CAPT_vect){
    //If interrupt is triggered on rising edge and not generating
       pulse
    if (TCCR1B & (1 << ICES1) && OCR1A == ULTRASONIC_timout)
    {
        //Change so interrupt is triggered on falling edge
        TCCR1B &= ~(1 << ICES1);

        risingICR = ICR1;

    } else if (!(TCCR1B & (1 << ICES1)) && OCR1A ==
       ULTRASONIC_timout){
        //Moving the distance measurement to the global var
        dist[ultrasonicIndex] = (ICR1 - risingICR);

        //Incrementing the ultrasonicIndex var
        ultrasonicIndex++;
        if(ultrasonicIndex > 2){ultrasonicIndex = 0;}
    }
}
```

../Code/idle.c

The second ISR is called depending on the DistanceState. If the interrupt is triggered on a rising edge and the timer is set to ULTRASONIC_timout, the interrupt is changed to a falling edge. The risingICR is set to the ICR1 value. If the timer is set to ULTRASONIC_timout and the interrupt is triggered on a falling edge, the distance measurement is moved to the global var and the ultrasonicIndex is incremented.

```c
void idle(void){
    //Set the next state
    current_state = state_pathfind_fire;
    //Add 1 to the cycle counter
    run_idle_counter++;
    //Start of the driving algorithm
    //Check if distance is too close to an object, if it is set
       correponding ultrasonic varibale, "flag", to 1
    //Variable for right ultrasonic sensor (number 2).
    if (dist[2] < TOO_CLOSE_RANGE){
        dist_2 = 1;
    }
    //Variable for left ultrasonic sensor (number 0).
    else if (dist[0] < TOO_CLOSE_RANGE ){
        dist_0 = 1;
    }
```

../Code/idle.c

At the start of the idle function, current_state is set to state_pathfind_fire, so after doing a cycle of this code, it jumps to state_pathfind_fire and looks for fire. Then the algorithm for obstacle avoidance is implemented. First 1 is added to run_idle_counter, so it is registered a new cycle has started.

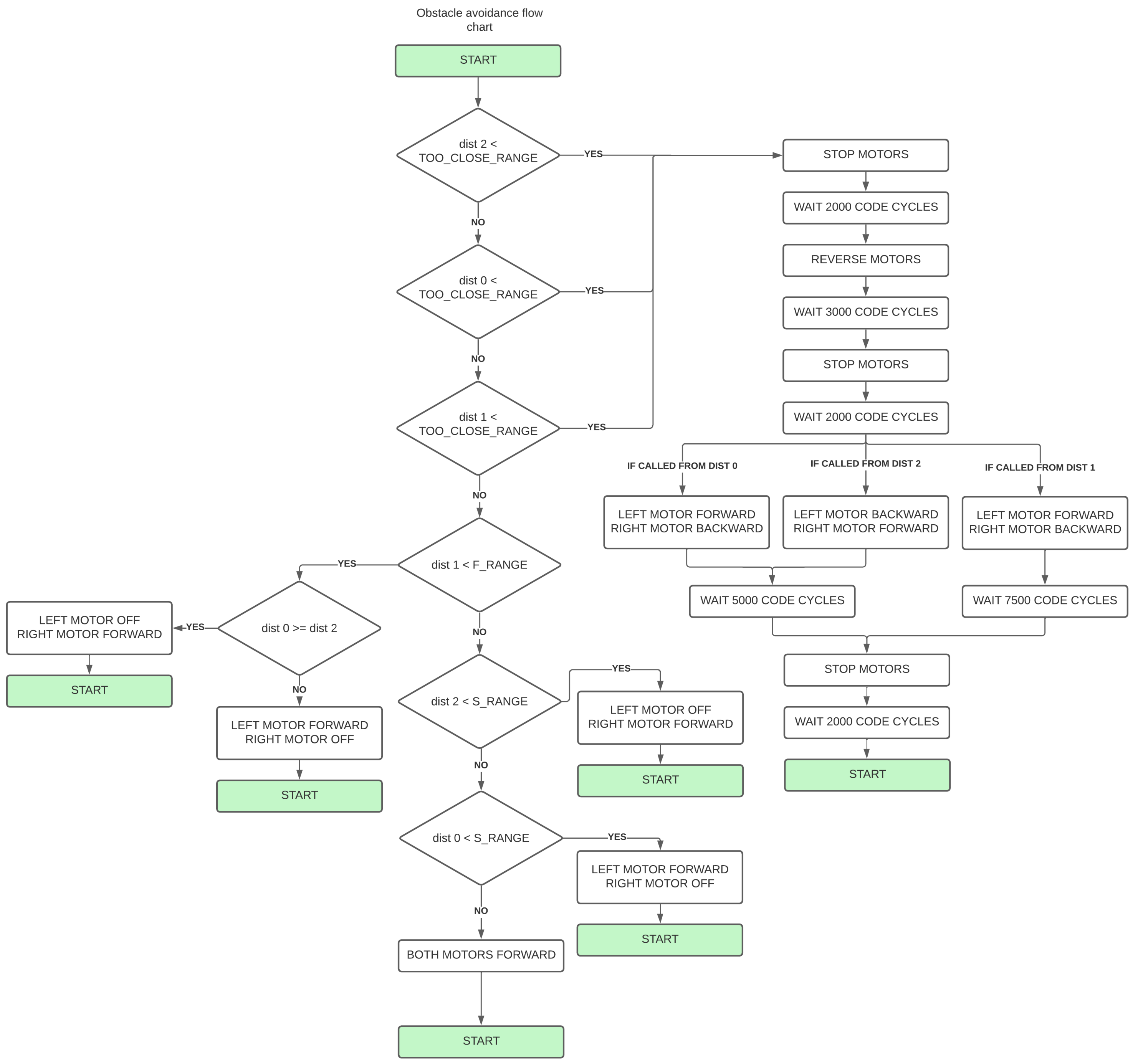


Figure 7.2: Driving algorithm flowchart

The obstacle avoidance algorithm starts by checking if either of the 3 ultrasonic sensors has an object in “TOO_CLOSE_RANGE”. If an object is closer than this range, this means a sharp turn needs to take place, so first the motors are stopped, then the motors are reversed, stopped again, then depending on the ultrasonic sensor that sensed an object in this range, it turns one way or another. The turn is made by making the motor spin opposite directions, so the robot can turn in the spot, avoiding hitting any object. If the left ultrasonic sensor (dist 0) detected the object, the left motor spins forward and the right one backward. If the right ultrasonic sensor (dist 2) detected the object, the right motor spins forward and the left one backward. If it was the center sensor (dist 1) that detected the object, a 180 degree is made by turning the left motor forward and the right one backward. To achieve this turn, the code lets the motor spin for 7500 code cycles instead of 5000 as it does if one of the side sensors detects the object. If none of the sensors are in this "TOO_CLOSE_RANGE", then the code checks if the center ultrasonic sensor detects something in the "F_RANGE". If it does, then it checks if the left sensor detects an object further away or at equal distance than the right one. If the object is further or equal from the left sensor, then the robot turns left by turning off the left motor and on the right one in the forward direction. Consequently, if an object is closer to the left sensor, a right turn is made by turning on the left motor and off the left one in the forward direction. If no ranges have been true yet, then the code checks if the right ultrasonic sensor is closer than "S_RANGE". If it is, turns on the right motor and off the left one. Then checks if the left ultrasonic sensor is closer than "S_RANGE". If it is, turns on the left motor and off the right one. By having 3 ranges, "TOO_CLOSE_RANGE", "F_RANGE" and "S_RANGE", adequate turns can be done. When the range is "TOO_CLOSE_RANGE", a sharp turn is done to avoid hitting the object. When the range is "F_RANGE", means that an object is in front of the object, but the sides are clear, so it checks which side is further away to go that way. If both sensors detect an object at the same distance, or neither detect an object, then the robot turns left. When range is "S_RANGE", the code checks if an object is close to the robot but can be avoided by doing a mild turn. That is why only one motor turns to make this turn. Finally, if no objects have been close enough to the robot to make it change direction, the two motors drive forward.

7.3 Fire detection & extinguishing (pathfind_fire.c)

As same as with the other files, first header files, voids, variables and arrays are included and defined.
After pathfind_sound_start function makes all the initialization for all ports and registers needed.

```c
void pathfind_fire_start(void)
{
    //Setting up pins for phototransistors
    TRIG_DDR |= (1 << TRIG1_pin | 1 << TRIG2_pin | 1 << TRIG3_pin);
    DDRB |= (1<<Out1 | 1<<Out45);
    DDRD |= (1<<Out23);
    //Setting up pin for the pump
    DDRD |= ( 1<<PUMP);
    //turn pump off
    PORTD &= ~(1<<PUMP);
    //Select Vref = AVcc
    ADMUX = (1<<REFS0);
    //Set prescaler to 128 and turn on the ADC module
    ADCSRA |= (1<<ADPS2 | 1<<ADPS1|1<<ADPS0 | 1<<ADEN);
    //set pin for the 1st Out on
    PORTB |= (1<<Out1);
}
```

../Code/pathfind_fire.c

First pins for the phototransistors and the pump are set up. It sets the direction data register, for the trigger pins to be an output and sets the DDR for the output pins to be an output as well. It then sets the pin for the pump to be an output and turns the pump off. It sets the Voltage reference to AVcc and sets the prescaler to 128 and enables the ADC module. Finally, it sets the pin for the first phototransistor output to be on.

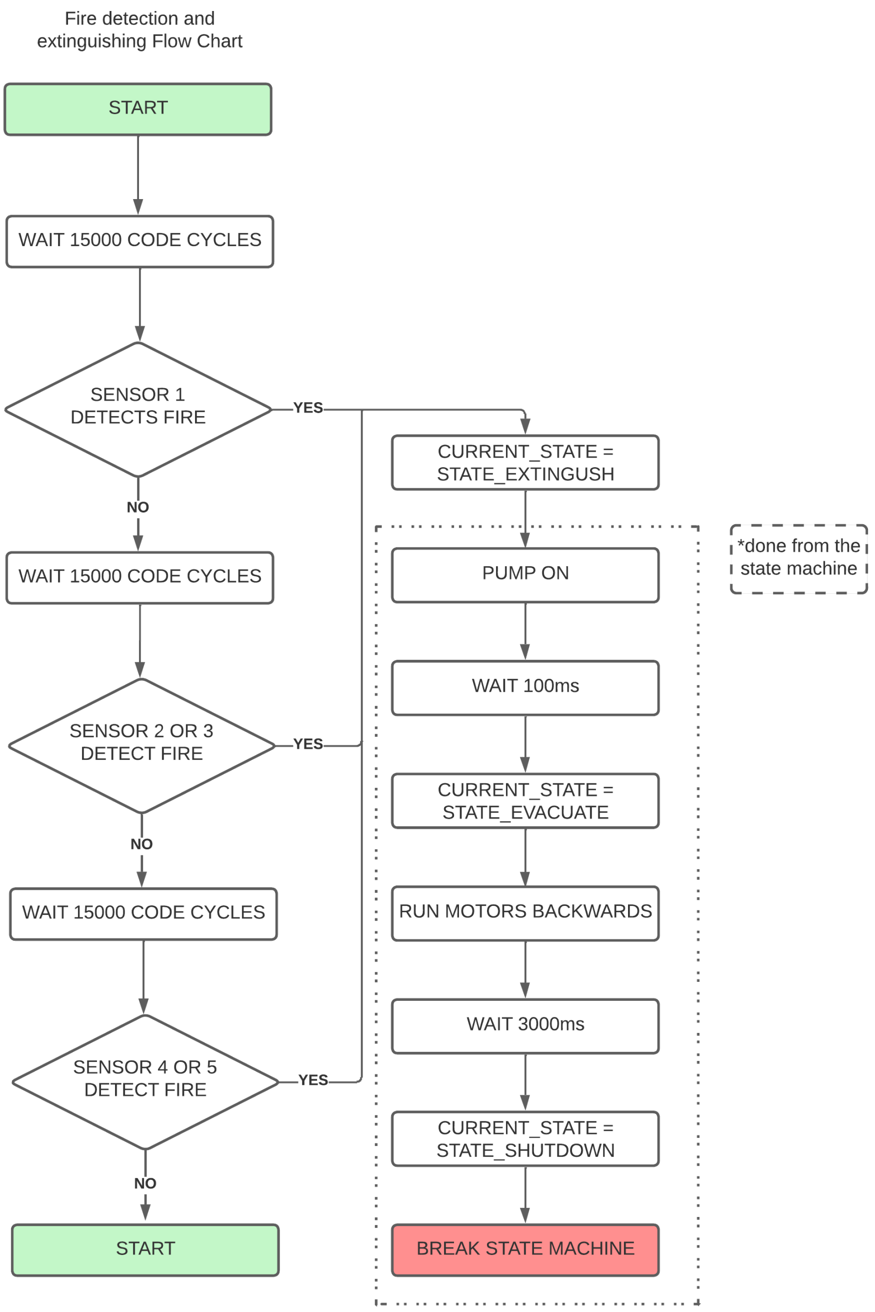

Fire detection and extinguishing Flow Chart
START
WAIT 15000 CODE CYCLES
SENSOR 1 DETECTS FIRE
YES
NO
WAIT 15000 CODE CYCLES
SENSOR 2 OR 3 DETECT FIRE
YES
NO
WAIT 15000 CODE CYCLES
SENSOR 4 OR 5 DETECT FIRE
YES
NO
START
CURRENT_STATE = STATE_EXTINGUSH
*done from the state machine
PUMP ON
WAIT 100ms
CURRENT_STATE = STATE_EVACUATE
RUN MOTORS BACKWARDS
WAIT 3000ms
CURRENT_STATE = STATE_SHUTDOWN
BREAK STATE MACHINE

Figure 7.3: Fire detection flowchart

The fire detection and extinguishing algorithm starts by checking if any sensor detects fire and then if fire is detected, gets back to the state machine to proceed with the fire extinguishing. Even though everything for fire detection and extinguishing is in the same file, is controlled through the state machine from the main.c file.

```
void pathfind_fire(void){
    //add 1 to the cycle counter
    run_path_fire_counter++;
    //start of the ir detetion
    //every 15000 cycles of code it jumps from 1 pair of IR
       sensors to the next one(the 1st sensor it is by its own)
    //1st IR sensor
    //jump into this if, if the run_path_fire_counter has done
       15000 cycles and run_pt_counter is equal to 1 do the
       following:
    if (run_path_fire_counter%15000 == 0 && run_pt_counter == 1){
        //1st ADC result equals the read from ADC in port in124
        adcResult[1] = adcRead(in124);
        //if the value is higher or lower than the threshold set
           current_state to state_extinguish
        if (adcResult[1] > FIRE_U || adcResult[1] < FIRE_D){
          current_state = state_extinguish;
        }
        //turn off pin for this phototransistor
        PORTB &= ~(1<<Out1);
        //set pins for the next phototransistors
        PORTD |= (1<<Out23);
        run_pt_counter ++;
        run_path_fire_counter++; //so it doesn't get into the next
           one
    }
```

../Code/pathfind_fire.c

The algorithm for detecting fire is in this pathfind_fire void. The first thing it does is add one to run_path_fire_counter to state a new cycle has started.
After the code waits for 15000 cycles when that has passed, if run_pt_counter is 1, the code gets the reading for the 1st phototransistor and stores it in the corresponding position of the adcResult array. Next the code checks if the value received was smaller or bigger than the threshold, FIRE_D and FIRE_U respectively. If the value was out of the threshold, then current_state is set to state_extinguish, because fire has been found. After, the Out pin for this phototransistor is turned off and the one for the next 2 are turned on.
Lastly, one is added to run_pt_counter, so code can get into the next phototransistors, and one is also added to run_path_fire_counter, so code does not immediately jump into reading the next phototransistors.
For the next phototransistors, the logic is the same, get the reading, compare it to the threshold and then turn off the pins used and turn on the ones for the next

phototransistors.
When all 5 phototransistors have been read, run_pt_counter is set to 1 to start again this process.

```c
void extinguish(){
    //turn on pump
    PORTD |= (1<<PUMP);
    //100ms delay
    _delay_ms(100);
    //turn pump off
    PORTD &= ~(1<<PUMP);
    //100ms delay
    _delay_ms(100);
    //set current_state to state_evacuate
    current_state = state_evacuate;
}
```

../Code/pathfind_fire.c

If fire is found, the state machine calls this function.
The first thing done is turning on the pump, then waiting 100 milliseconds and after turning it off. Lastly current_state is set to state_evacuate.

```c
void evacuate(void){
    //Set motors to oposite direction
    PORTB |= (1<<DIRECTION_1 | 1<<DIRECTION_2);
    //Set both motors to fast speed
    OCR2A = FAST;
    OCR2B = FAST;
    //Do a delay so the motors go on reverse for 3s
    _delay_ms(3000);
    //Stop the motors
    OCR2A = SLOW;
    OCR2B = SLOW;
    //Set next state to state_shutdown
    current_state = state_shutdown;
}
```

../Code/pathfind_fire.c

From the state machine when this function is called, the first thing done is reversing the motors to get away from the fire, then a 3000-millisecond delay is implemented, and then the motors are stopped. Lastly, current_state is set to state_shutdown.

State_shutdown will break the state machine and the robot will do nothing until it is restarted.

7.4 Proposed microphone implementation

For easier fire detection, microphones were installed to the robot to guide it towards fire alarms. But when trying to implement it, many problems appeared, so they were not implemented.
If they had been implemented, the logic of the code would have been the following:
For the state machine, after getting out of state_idle, would have jumped to detect for sound before trying to find fire. The following algorithm was thought to be implemented:

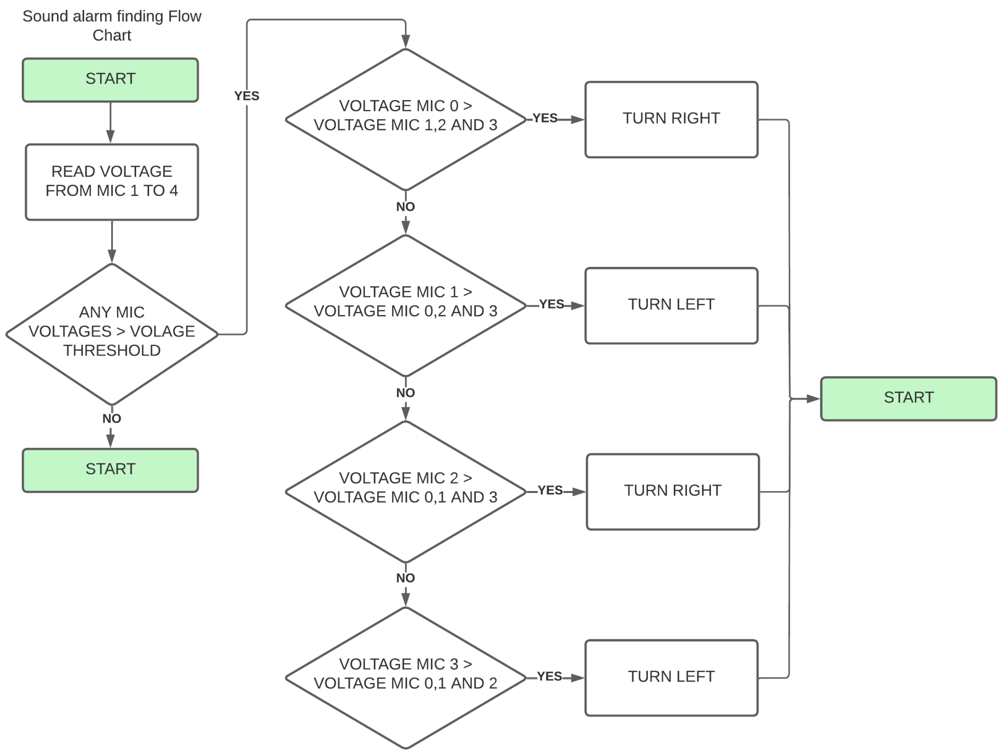

Sound alarm finding Flow Chart
START
READ VOLTAGE FROM MIC 1 TO 4
ANY MIC VOLTAGES > VOLAGE THRESHOLD
YES
NO
START
VOLTAGE MIC 0 > VOLTAGE MIC 1,2 AND 3
YES
TURN RIGHT
NO
VOLTAGE MIC 1 > VOLTAGE MIC 0,2 AND 3
YES
TURN LEFT
NO
VOLTAGE MIC 2 > VOLTAGE MIC 0,1 AND 3
YES
TURN RIGHT
NO
VOLTAGE MIC 3 > VOLTAGE MIC 0,1 AND 2
YES
TURN LEFT
START

Figure 7.4: Sound detection flowchart

The main concept was to check which microphone got a higher voltage and from that make the robot turn left or right until it reached the fire alarm. Due to lack of time and not being implemented in the project, no better logic was created, but with more time, a more intelligent algorithm would have been made to reach the fire alarm faster and more effectively.

```c
void pathfind_sound(void){
    //read voltage from the 4 michropones, one at a time
    for(int channel = 4; channel < 8; channel++){
        adc_result = adc_read(channel);
        //store voltage value from reading in voltage array in the
           chosen order
        voltage[channel] = adc_result*(V_REF/1024.0);
        //check if voltage is higher than threshold
        if (voltage[channel] > VOLTAGE){
            //call mic_ON void
            mic_ON();
        }
    }
    //set current_state to state_pathfind_fire
    current_state = state_pathfind_fire;
}
```

../Code/pathfind_sound.c

The microcontroller reads the voltage from the four microphones, one at a time. For each microphone, it reads the voltage from the ADC and stores it in an array in the chosen order. Then, it checks if the voltage is higher than a certain threshold. If it is, it calls the mic_ON void.

7.5 Smarter fire finding logic

The first thought for the fire finding logic was to only use the phototransistor on the bottom to call the extinguish state, and the other 4 to position the robot in front and close enough to the fire.
A simple state chart to understand what the concept thought to be implemented, would be the following:

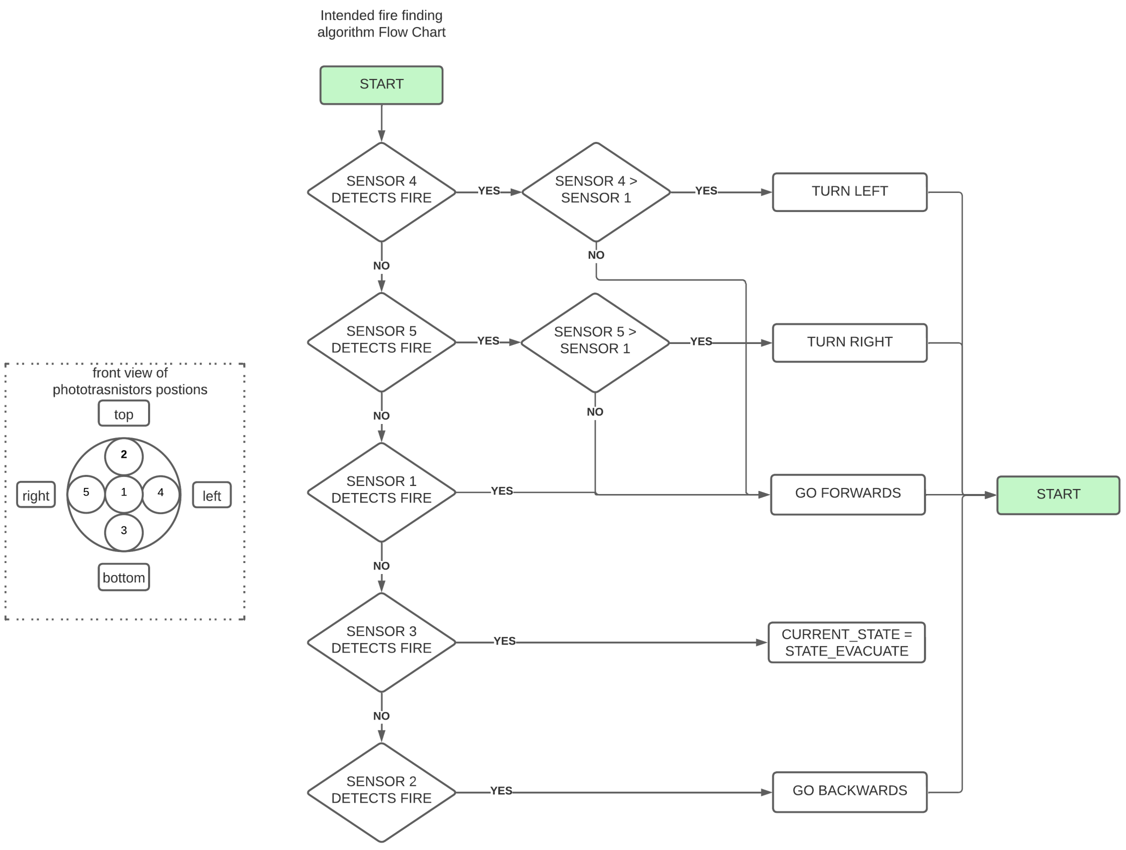


Figure 7.5: Advanced fire detection

The main principle in this way of locating the fire is comparing the values of the phototransistors to steer the robot towards the fire. If one of the side sensors detects fire, then it is compared with the center one and turns if it is necessary. If the center one detects fire and neither one of the side sensors have detected fire, it means the fire is in front of the robot. If the bottom sensor detects fire, the robot is close enough to extinguish the fire. Lastly, for safety, if the top sensor detects fire, the robot is too close to the fire and must back up to get a safe distance.
With this logic implemented, the next improvement would be, after spraying water, to go back to this state to recheck for fire. If no values outside the threshold are received, it means that the fire is extinguished. Otherwise, there is still fire, and the robot must reenter this state and keep positioning itself and when fire is found call state_extinguish to eradicate the fire.

8. Testing

To get an overview of the project results, a requirements check was made in table 8.1, going through each of the primary requirements of the project.

Requirements	**Test result**
Include at least two analog sensors	PASSED
Include at least one analog filter	PASSED
Stay within 1000kr. development budget	PASSED
Able to locate fire	PARTIALLY PASSED
Have a PCB for electronics	PASSED
Able to extinguish/suppress fire	FAIL
Move to fire autonomously, without moving into fire	PARTIALLY PASSED

Table 8.1: Requirements check

Include at least two analog sensors: The project includes phototransistors and microphones, therefore it passed.
Include at least one analog filter: The microphones utilizes a bandpass filter and the phototransistors use a highpass filter, therefore also passed
Stay within 1000kr. development budget: The total was about 550 DKK, including PCB, components and pump, which is within our budget
Able to locate fire: Partially passed, since it was able to detect fire at very close range and very long sensor intervals
Have a PCB for electronics: All of the electronics were included on a physical PCB, hence passed.
Able to extinguish/suppress fire: The robot was able to spray water, but only in a straight line, presumably not hitting fire.
Move to fire autonomously, without moving into fire: It was able to move around and avoid obstacles, as well as detect some fires if directly in front. It would also back off from fire if found, therefore partially passed.

9. Conclusion & Reflections

A lot of learnings have been made doing this project. For starters there was some challenges controlling the motors since the MCU is not getting any feedback of their velocity, and therefore cannot respond actively (it is not possible to use any control algorithms like e.g. PID to control the motors). Including two opto interrupters would have been beneficial.
Next there is the issues with the update rate of the phototransistors. since a measurement cycle of all 5 phototransistors takes approx 4.5 seconds, it is not going to be feasible to actively seek out fire. If another prototype was to be made, each phototransistor should probably have its own adc channel, so the array could be removed.
At last; although the microphones were not completely implemented, some improvements have been considered. First of all a filter does not really make sense for this application. Since the microphones is only really listening for one frequency, a better performing MCU and ADC combined with some digital signal processing like a Fast Fourier Transform would have been a better solution. However since the project needed an analog filter, the current solution is fitting.
Other factors that definitely impacted our project includes loosing a group member early on in the project and a slow PCB delivery time halting the project and thereby postponing the debugging of electronics. Although the project didn't succeed completely, it still fulfilled most requirements and proved the concept of a fire supression robot. With more development time it could definitely prove a useful addition to firefighting.

Bibliography

[1] G. F. P. Services, *Fire protection for shopping malls in maryland.* [Online]. Available: https://www.guardianfireprotection.com/blog/fire-protection-for-malls-maryland/ (visited on Nov. 9, 2022).

[2] S. G. Systems, *What is the cost of a commercial fire sprinkler system?* [Online]. Available: https://smokeguard.com/blog/2022/february/02/what-is-the-cost-of-a-commercial-fire-sprinkler-system (visited on Nov. 9, 2022).

[3] F. T. International, *Three levels of data center fire protection.* [Online]. Available: https://www.firetrace.com/fire-protection-blog/three-levels-of-data-center-fire-protection (visited on Nov. 9, 2022).

[4] fourobot, *Cobra-i - portable fire robot.* [Online]. Available: https://www.fourobot.com/Cobra-IFirefightingRescueRobot.html (visited on Nov. 9, 2022).

[5] Ansys, *What is dfmea.* [Online]. Available: https://www.ansys.com/blog/what-is-dfmea (visited on Dec. 30, 2022).

[6] R. PRO, *Brushed dc geared motor.* [Online]. Available: https://docs.rs-online.com/7113/A700000007082457.pdf (visited on Dec. 14, 2022).

[7] Bete, *Nozzle selection guidelines.* [Online]. Available: https://bete.com/how-to-select-a-nozzle/ (visited on Nov. 14, 2022).

[8] Bete, *Standard fan nozzle.* [Online]. Available: https://bete.com/wp-content/uploads/2022/01/BETE_NF.pdf (visited on Nov. 14, 2022).

[9] U. C. P. S. Commission, *The audibility of smoke alarms in residential homes.* [Online]. Available: https://www.cpsc.gov/s3fs-public/audibility%20%281%29.pdf (visited on Nov. 9, 2022).

[10] B. Carter, *Op Amps for Everyone - Fifth Edition.* Mara E. Conner, 2017, ISBN: 978-0-12-811648-7.

[11] T. Instruments, *Single-supply rail-to-rail operational amplifiers - opa340-ep*. [Online]. Available: https://www.ti.com/lit/ds/symlink/opa340-ep.pdf?HQS=TI-null-null-alldatasheets-df-pf-SEP-wwe&ts=1672332139395&ref_url=https%253A%252F%252Fpdf1.alldatasheet.com%252F (visited on Dec. 4, 2022).

[12] P. Shaw, *Microphone pre amplifier using lm358*. [Online]. Available: https://digitalab.org/2016/08/microphone-pre-amplifier-using-lm358/#.YPWuzy2FDVs (visited on Dec. 4, 2022).

[13] S. Safaei, *A practical method of fire radiant heatflux sensing*. [Online]. Available: https://web.wpi.edu/Pubs/ETD/Available/etd-090816-094929/unrestricted/A_Practical_Method_of_Fire_Radiant_Heatflux_Sensing_Samim_Safaei.pdf (visited on Dec. 30, 2022).

[14] OSRAM, *Silicon npn phototransistor - sfh 313 fa*. [Online]. Available: https://docs.rs-online.com/42d8/0900766b808b2f31.pdf (visited on Dec. 30, 2022).

[15] Atmel, *Atmega328p*. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (visited on Dec. 2, 2022).

[16] elecfreaks, *Ultrasonic ranging module hc - sr04*. [Online]. Available: https://datasheetspdf.com/pdf-file/912326/ElecFreaks/SR04/1 (visited on Dec. 2, 2022).