CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS

May 2020

Ondřej Procházka

Robotic Fire Extinguisher Mounted to an Unmanned Aerial Vehicle

Department of Cybernetics

Thesis supervisor: **Ing. David Žaitlík**

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Procházka Ondřej**	Personal ID number:	**474744**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Robotic Fire Extinguisher Mounted to an Unmanned Aerial Vehicle

Bachelor’s thesis title in Czech:

Robotický hasící systém pro bezpilotní dron

Guidelines:

This thesis will focus on the design and development of a fire extinguishing system for an Unmanned Aerial Vehicle (UAV). The task is motivated by the MBZIRC 2020 robotic competition, where a group of robots is tasked with automatic fire fighting within a building. The designed system will consist of a custom robotic arm with a nozzle, a pump, and a water storage tank. Moreover, a control unit with a dedicated microcontroller shall be
used to automate the pump control. The thesis shall consist of the following tasks:
1) Design and test a fire extinguishing device based on an electrical water pump.
2) Design a robotic arm that will allow pointing the water nozzle in the desired direction.
3) Implement a control system for the manipulator. Utilize the provided STM32 microcontroller embedded platform.
4) Devise and implement an Inverse Kinematic Task for the manipulator.
5) Implement USB communication between a computer and the microcontroller in order to control the fire extinguishing system from the Robot Operating System.

Bibliography / sources:

[1] M. Fumagalli, S. Stramigioli and R. Carloni, "Mechatronic design of a robotic manipulator for Unmanned Aerial Vehicles," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, 2016, pp. 4843-4848.
[2] Carmine Noviello; “Mastering the STM32 Microcontroller“; Lean Publishing; 2016.
[3] Anis Koubaa; “Robot Operating System (ROS): The Complete Reference (Volume 3)“; Springer; 2018.
[4] Lynch, K. M., & Park, F. C.; “Modern Robotics: Mechanics, Planning, and Control”; Cambridge University Press; 2017.

Name and workplace of bachelor’s thesis supervisor:

Ing. David Žaitlík, Multi-robot Systems, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **10.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. David Žaitlík	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.
Date of assignment receipt — Student’s signature

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

Acknowledgements

I would like to thank my adviser for his advice and also for his patience. I would also like to thank my family that they supported me during the study, although it was not easy. Furthermore, I thank my girlfriend that she had patience with me. Finally I would like to thank my cousin who helped me with English.

Abstract

This thesis contains design a realisation of a robotic fire extinguisher mounted to an unmanned aerial vehicle. The fire extinguisher system is able to direct water stream into the fire location detected by sensors. The two degrees of freedom manipulator is based on inverse kinematics enumeration, and the water pump turns on only if the set range condition is fulfilled. During this project's development, the maximal weight of the drone, as well as the maximal weight of the fire extinguisher system with full water storage, had to be taken into account. First, the suitable hardware was developed, then the mathematical enumeration was performed and implemented into the control program. The simulations in this project were performed in Matlab. The functionality of the fire extinguisher system was verified in real-life experiments.

Keywords: UAV, extinguisher system, ROS, robotic manipulator

Abstrakt

Tato práce obsahuje návrh a realizaci robotického hasícího systému upevněného na autonomní dron. Hasící systém je schopný směřovat paprsek vody do určené pozice ohně, která je detekována pomocí senzorů. Manipulátor se dvěma stupni volnosti je patřičně nastavován na základě výpočtu inverzní kinematické úlohy a voda je vypouštěna jen v okamžiku splnění podmínek dosahu. Celou dobu byl také kladen důraz na celkovou nosnost drony a tedy i celkovou hmotnost hasícího systému včetně nádrže naplněné vodou. Nejprve byl tedy navržen hardware, potom byly provedeny matematické výpočty, které byly následně implementovány do řídícího programu. Veškeré simulace byly provedeny v prostředí Matlab. Během reálného testování bylo ověřeno, že hasící systém je funkční.

Klíčová slova: UAV, hasící systém, ROS, robotický manipulátor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs), also known as micro aerial vehicles (MAVs) or drones, have experienced a boom in recent years. Their technological improvements are probably the reason why drones are so popular these days. Due to the miniaturisation of electronic parts, the control units are smaller. The DC brushless motors are more efficient and the batteries have higher energy density than their predecessors. All of the above leads to a decrease in drone prices which makes them more accessible to the general public.

The most widespread drones are camera drones, most of them come from DJI factory. These drones are usually equipped with four motors with propellers and with the camera mounted on a bottom part of the drone. Thanks to the drones' many sensors, the drones are easily controllable. The drone's software also includes an obstacle detection system which significantly lowers the risk of crashing. These drones also carry a powerful control unit, which allows them to do complex calculations. For example, video from drone's camera can be processed in real-time which allows tracking object in motion (person tracking). This description also fits drone which is displayed in picture 1.1.

Figure 1.1: Mavic Air 2[1]

[1]Source: https://store.dji.com/product/mavic-air-2?vid=91101

Research in the field of UAV is dealing (among other things) with how the drones can be used to perform complicated tasks, to help people, simplify their work and improve safety. Therefore a large number of competitions emerges. On those competitions, many teams from all around the world compete in many different tasks. One of these competitions is DARPA Subterranean Challenge (DARPA stands for Defense Advanced Research Projects Agency). In this challenge, the main assignment is to map, navigate, and search dynamic underground environments with unmanned vehicles. This challenge is regularly attended by Center for Robotics and Autonomous Systems (CRAS) group from Czech technical university in Prague.

Another competition is Mohamed Bin Zayed International Robotics Challenge (MBZIRC), which has been already attended twice by Multi-robot Systems (MRS) group, also from Czech technical university in Prague, under the leadership of Dr. Martin Saska. This year, one of the tasks was to extinguish fire somewhere on a high-rise building. Therefore my task in this thesis was to invent and create the *Robotic Fire Extinguisher Mounted to an Unmanned Aerial Vehicle.*

1.1 State of the art

In recent years we have been hearing about wildfires more often. The global temperature rises, and the global climate is dryer. Moreover, blazes occur mostly in inaccessible places. Therefore, these areas can only be extinguished by helicopters or planes. But what if this fire is on a steep side of a hill. How can the helicopters or planes drop water to that area? Unfortunately, fire does not exist only in the wilderness. Especially in densely populated areas, fire can be very dangerous to people and their property. For example, it is almost impossible to extinguish a fire in high-rise buildings.

Usage of drones seems to be logical for this task. The drones can fly high, hover and carry various equipment. They can also be very fast. The first of mentioned drone design [6] is inspired by an existing method of extinguishing which is used by helicopters. A bucket is filled with water and dropped down onto the fire area. In this publication, authors also studied the dependence of drop height on the water covered area. Another solution is based on drones dropping fire extinguisher balls into the fire. Layout of this drone is described in publication [7]. But neither solution can be effectively used during extinguishing on a side of a building.

In order to reach the desired location on a side of a building, it is necessary to either use a nozzle, for the solution with water storage, or to use the special type of gun for fire extinguisher balls. In publication [8] the author introduces electric spring-operated gun mounted on a drone. This gun is capable of shooting small fire extinguisher balls up to 20m. In [9], the authors explain their drone concept with on-board water storage with a nozzle. In another paper [10] author describes creation of a drone with the ability to squirt water with a pump connected to a reservoir on a tank based vehicle. Because in these papers the nozzle is rigidly mounted on the drone, the aiming to the right position depends on drone control. This might be a disadvantage. The drone must hover in the exact position to reach the required location of fire.

This brings up the idea to attach a robotic arm, with ability to direct a stream of water to the required position, on the drone. We need to realise that the movement of the manipulator mounted on the drone can negatively affect the drone. Lots of publications [11] [12] [13] can be found on this topic, but this problem is not in the scope of this project. Moreover, it is expected in this project, that the manipulator will not be big enough to create unwanted forces on the drone.

There are a few companies that are trying to develop firefighting drones with required abilities. But they do not use manipulator. Moreover, they do not usually use the on-board water storage. For example, the Aerones company developed a drone connected with a water tank on the ground via a hose. Unfortunately, there is only a video of their concept, because there is no evidence it is still work in progress. Fortunately, this does not apply to the International Armour Group of Companies and affiliates. They created a firefighting drone[2] with an on-board powder tank, but without the use of a manipulator.

1.2 Used Drone Description

The drone I used is built on Tarot T650 Sport frame. The motors are BLDC Tarot 4114 320KV with ESC Turnigy Multistar 51A. Control core of the drone is computer NUC8i7BEH to which a Pixhawk 4 (an autopilot) with GPS module is connected. The drone is also equipped with sensors - Garmin Lidar-Lite that measures distance from the ground and Camera mvBlueFOX-MLC that uses optic flow as fallback for the drone's stabilisation. Everything is powered using LiPo 6S 8000mAh battery pack. This is a standard UAV equipment in MRS group. However in this situation, special sensors are required.

For this thesis the drone was equipped with these sensors - Garmin Lidar-Lite, which is used indoors to measure distance from the ceiling and Intel Realsense D435, which is used to detect window on a building. RP-Lidar AR was used for simultaneous localisation and mapping (SLAM). The fire was detected by three TeraRanger Evo Thermal 33 thermal cameras that are placed to face 33 degrees apart to reach the 99 degrees high and 33 degrees wide field of view.

[2]Source: http://www.armour.gr/firefight-drone.html

1.3 Outline

This thesis is split into four main chapters. Chapter 2, named Hardware, introduces all the hardware used in this project. The next chapter, Chapter 3, provides information about the manipulator. The manipulator itself, as well as mathematical derivation for Inverse Kinematics task, is described in this chapter. Chapter 4 is called Software. This chapter includes information about the development tools, data transition ways, the Dynamixel protocol and also description of the whole system. The last chapter, Chapter 5, is devoted to experiments including images taken during those experiments.

Chapter 2

Hardware

Since I am doing the whole project from scratch, I dedicated this chapter to the hardware solution, especially for the robotic arm creation. For rotatory parts of the arm, servomotors will be used, and for stationary parts, I will need a nozzle, bag holder and some linking components which will link the servomotors and other elements together. Some of these components can be created and printed on a 3D printer. But there were also parts I needed to buy. I had to determine which pieces fit the best requirements. I will outline these below. In this chapter, I also describe components which I decided not to use in the final design.

2.1 Servomotors

A servomotor is an essential part of the manipulator because it ensures joint rotation. There are many types of servomotors. A servo is a motor, but unlike an engine, the servo has position control. Instead of the servo, I can also use a stepper motor. Therefore I would like to examine the servomechanism itself and try to compare it with other types of position control.

2.1.1 Servomotor Mechanism

The servomotor mechanism is used as a position control mechanism close-loop which allows controlling position by feedback. The closed-loop transfer function enumerates from the set angle if the servo is in the goal position or not. And if the error which is in this situation difference between set position and actual position, is not equal to zero then the transfer function initialises movement until the servo reaches the goal position.

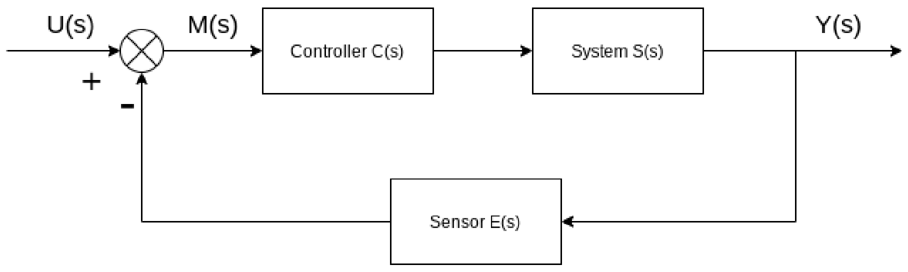


Figure 2.1: Feedback loop with descriptions

$$
\begin{aligned}
Y(s) &= M(s)C(s)S(s) \\
M(s) &= U(s) - E(s)Y(s)
\end{aligned}
\tag{2.1}
$$

Now I can substitute M to the first equation and enumerate transfer function which is in the essential form $H(s) = \dfrac{Y(s)}{U(s)}$.

$$
\begin{aligned}
Y(s) &= [U(s) - E(s)Y(s)]C(s)S(s) \\
Y(s) &= U(s)C(s)S(s) - E(s)Y(s)C(s)S(s) \\
Y(s)(1 + E(s)C(s)S(s)) &= U(s)C(s)S(s) \\
\frac{Y(s)}{U(s)} &= \frac{C(s)S(s)}{[1 + E(s)C(s)S(s)]} = H(s)
\end{aligned}
\tag{2.2}
$$

The position can be determined by incremental or absolute rotary encoder. I mentioned the stepper motor only to emphasize that there is also an alternative way to create the rotary joints. The stepper motor uses open-loop transfer function for position control. That means the stepper motor operates in ticks without any feedback control. I can not determine whether the motor actually is in the goal position because during its movement there is a possibility of the wrong action caused by outer environment or a wrong tick, which I am not able to determine. Based on these facts, I decided to use servo instead of the stepper motor.

2.1.2 Used Servomotor

Before this project began, I did not have any experience with the servomotors. Therefore I turned to my supervisor who recommended me to ask the Computational Robotics Laboratory (ComRob)[14] because they worked with the robots that used servomotors for movement. Based on their experience, they recommended I use the AX-12A servo. The

Figure 2.2: AX-12A servo [1]

servo uses electrically erasable programmable read-only memory (EEPROM) and Random-Acces-Memory (RAM). The difference between these types of memories is that the EEP-ROM is a non-volatile memory, but on the other hand, the RAM is wiped clean every time power is cut off. The servo uses an asynchronous half-duplex serial communication. On the EEPROM memory are model number, firmware version and other information stored. There is also a possibility of setting maximum torque and clockwise limits. On the RAM, we can set the goal position or even moving speed. The servo can operate in a wheel mode, but in this mode it does not support position control, on the other hand, the servo behaves as the continuous motor. Therefore I used joint mode, which allows me to use the servo in 0 to 300 degrees range with position control. The angle 0 corresponds with goal position 0, and the 300 degrees corresponds with 1023 goal position coordinates.

The servo has feedback with some error messages - for example, movement indicator or overheating. For more details, you can look at table 2.1, and the servo can be seen in picture 2.2. The servo has lots of other functions, but in this thesis, I decided to describe only the function I used for this project. For further details about this servo, I recommend the official website ROBOTIS e-Manual.

2.2 Pump

There are many ways to transport water into the fire. Since I am doing the robotic extinguisher system purpose of which is to transport water to the fire zone, I had to decide which method to use. The first of many ways of taking water and letting it fall into the fire location is used during wildfire suppression nowadays. The helicopter dips its bucket in the reservoir before dropping all of its capacity to the ground.

Another technique is based on compressing water into a vessel, which is placed be-

Item	Specifications
Baud Rate	7843 bps ∼1 Mbps
Resolution	0.29°
Running Degree	0° ∼ 300° Endless Turn
Weight	54.6g(AX-12A)
Dimensions (W x H x D)	32mm x 50mm x 40mm
Gear Ratio	254 : 1
Stall Torque	1.5 N*m (at 12V, 1.5A)
No Load Speed	59rpm (at 12V)
Operating Temperature	$-5°C \sim +70$°C
Input Voltage	9.0 ∼12.0V (Recommended : 11.1V)
Command Signal	Digital Packet
Protocol Type	Half Duplex Asynchronous Serial Communication
Physical Connection	TTL Level Multi Drop Bus
ID	0 ∼253
Feedback	Position, Temperature, Load, Input Voltage, etc
Material	Engineering Plastic

Table 2.1: Servo specifications [5]

neath the drone, and then if the fire is detected, the vessel is uncorked, and water is hosed out into the fire.

The third method is a special kind of gun that can shoot projectiles with some sort of gas which is designed to put out the fire.

The first method does not ensure successful transportation of water in vertical places. Another problem is that the first two methods are designed to dispose of all the water at once. Furthermore, the over-pressured vessel and gun mechanism can be dangerous. Therefore I decided for a fourth method which uses a pump. In comparison with the previous techniques, the pump can control the release and suspension of water flow which allows me to divide the water reservoir into many small fires.

2.2.1 Type of Pump

We can divide pumps into two categories. There are pumps made to be submerged in a fluid, and there are pumps meant to be placed externally to the liquid. Because I have only small storage of water, I decided to use the type of mechanical pump which is placed externally to the fluid. I am also very limited by the weight of the pump and that means that I can not use the pump with the self-suction.

2.2.2 Used Pump

At first, I tested a pump whose function is to pump fluid to windshield washers. The type of the pump is B805 D1841 for Alfa-Romeo and the operation voltage of this pump is 12 Volts with direct current. But this pump did not manage to squirt the required distance; therefore, I bought a different pump which is twice as powerful but also twice as big and heavier than the first one. The name of the second pump is Comet VIP-Plus-Inline. The main difference is that this pump operates on a higher Voltage. Therefore I can power it directly from the battery which has 23-25 Volts in full charge and gives direct current. The remaining parameters can be seen in table 2.2 below. Figure 2.3 displays the pump Comet VIP-Plus-Inline.

Pump Type (category)	Low-voltage flow pump
Operation voltage	24 V
Input current (max.)	1,25 A - 1,88 A
Pressure connector	10 mm
Time use	30 min
Shipping height (max.)	10 m
Pump pressure (max.)	1 bar
Flow (max.)	1200 l/h
Cable length (max.)	1 m
Height	144 mm
Diameter Ø	48 mm
weight	180 g

Table 2.2: Outdoor flow pump Comet VIP-Plus-Inline [2]

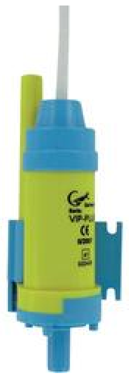


Figure 2.3: Pump Comet VIP-Plus-Inline [2]

2.3 Water Bag and Tubes

As a water reservoir I chose a water bladder, pictured in figure 2.4, which is mainly made for sport. I have got two sizes of the water bag. First one is 1.5 litre and the second is 2 litres in volume. This type of container gives many bonuses. For example, it does not resist compression, which allows using all of its capacity. To connect with the pump, I used a pipe of a inner size diameter 8.3 mm. For connection with the pump, I created a reduction, but for testing, we removed it and put the tubes together.

Figure 2.4: Water bag [3]

2.4 Control Board

For the configuration of the servomotors, control pump and communication with a computer placed on-board (main PC of a drone) I used a microcontroller STM32F042K6T6 [15] with clock rate at up to 48 MHz and a electronic (solid-state) non-volatile memory (FLASH) with 32 KB capacity and 6 KB of static random-access memory (SRAM). The microcontroller (MCU) has lots of enhanced peripheries - Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), Universal Synchronous/Asynchronous Receiver/Transmitter (USART), USB and more. Because I needed USART and USB communication I used only these peripheries. The USB corresponds to USB version 2.0 with maximal speed of 480 Mbit/s.

The same microcontroller is also placed on Nucleo-F042K6 which I used as a prototype to my solution. One of Nucleo's benefits is the ST-LINK and serial wire debugger on the board. There is also Mbed IDE support. This Nucleo follows the Arduino compatibility headers which can be seen in picture 2.5.

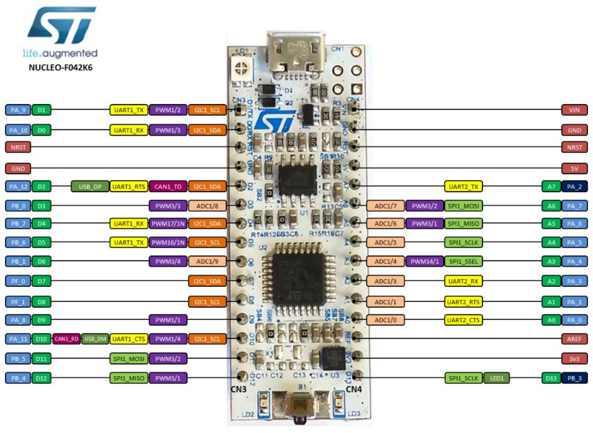


Figure 2.5: Compatible headers [4]

2.4.1 Breadboard Assembly

On breadboard was create the prototype but in final design was used PCB which is mentioned in section bellow 2.4.2. Before assembling components on breadboard I created a schematic in KiCAD. KiCAD is a software that allows drawing circuit connections and models of PCB. The schematic can be seen in figure 2.6. Afterwards I created a real circuit based on the schematic on breadboard which can be seen in picture 2.7. Using this circuit connection I tested communication with servomotors.

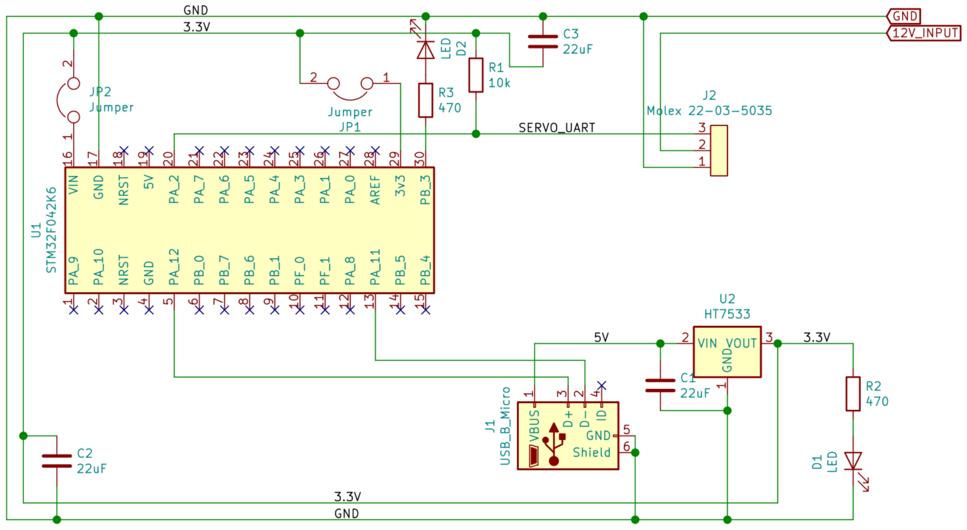


Figure 2.6: Connection schematic

In picture 2.7 can be seen coloured circuit, where the green wires are ground, red wires are for 3.3 volts, a yellow wire is for 5 volts, orange wires are data cables and grey is 12 voltage.

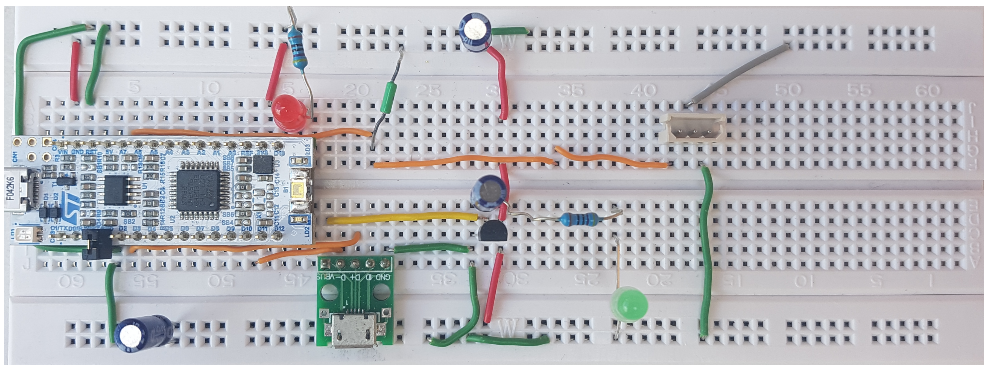


Figure 2.7: Connection on breadboard

2.4.2 Printed Circuit Board (PCB)

Because the Nucleo has lots of components which are not needed a PCB was designed in the final design. ST-LINK was removed because it can be used separately for uploading program into microcontroller and LEDs are not needed at all. PCB also reduces size of the control board. I would like to thank my supervisor for helping me by creating this PCB.

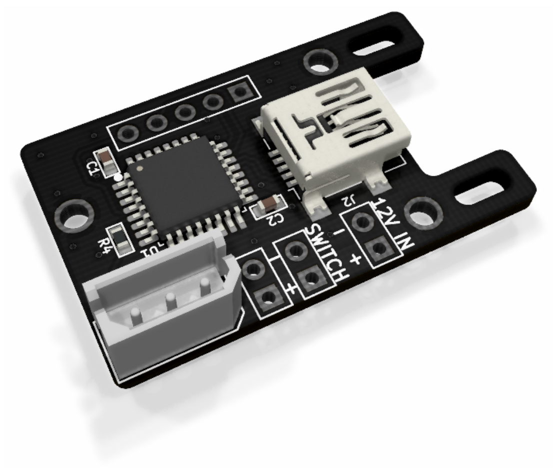


Figure 2.8: PCB

2.5 3D Models

For the purpose of this project I created several 3D models. All of them were created using Autodesk Fusion 360 and printed on Original Prusa i3 MK3 with nozzle size 0.6 mm. Printing materials used were polylactic acid filament (PLA) and polyethylene terephthalate filament (PETG).

2.5.1 Nozzle

Nozzle with Laminar Flow

Laminar flow is a non-turbulent stream of water. It can be better described by mathematical equations which I am using for inverse kinematics enumeration. Because of its nature, the pump can only create a turbulent flow which is going through the tube and nozzle. The chaotic stream in the nozzle affects the flow on the output - the output flow consists of many streams that flow in slightly different directions. While searching for a solution to this problem, I found a special nozzle which can create laminar flow from the turbulent stream. Unfortunately, there is a problem with this solution regarding the size. There is no way to create this nozzle in the required size I need due to limitations of the 3D printer available.

Normal Nozzle

Since I do not know how far from the fire location, the drone will be, and I also do not know how the size of the nozzle affects the speed of pumping water. I also have to take into account the nozzles' diameter, because smaller diameter means high resistance. I decided to design three sizes of one nozzle. Based on tube size and mathematical enumeration of how far will the water squirt without nozzle, I decided to create three sizes 2 mm, 3 mm and 4 mm in diameter on the output. The picture of the 4 mm nozzle can be seen in figure 2.9.

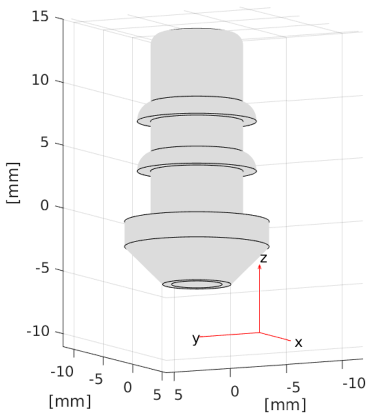


Figure 2.9: Nozzle with a 2 mm radius

2.5.2 Bag Holder

Another task that needed to be accomplished was choosing a method of hanging up the bag of water under the drone. I had to take a full container and its mass into account. The biggest problem I encountered here was the bag movement, which affected the drone. It created inappropriate forces on the drone, which complicated manoeuvrability. Because of that, I considered creating a bag holder. I expected the extra weight, but in the end, the downside of the weight outweighs the control issue; therefore, I created a bag holder. The 3D render can be seen in picture 2.10. The bag in the holder must be placed in a position that guarantees that the pump will be below the water surface. To reduce the holder weight, I connected the printed parts by small carbon sticks.

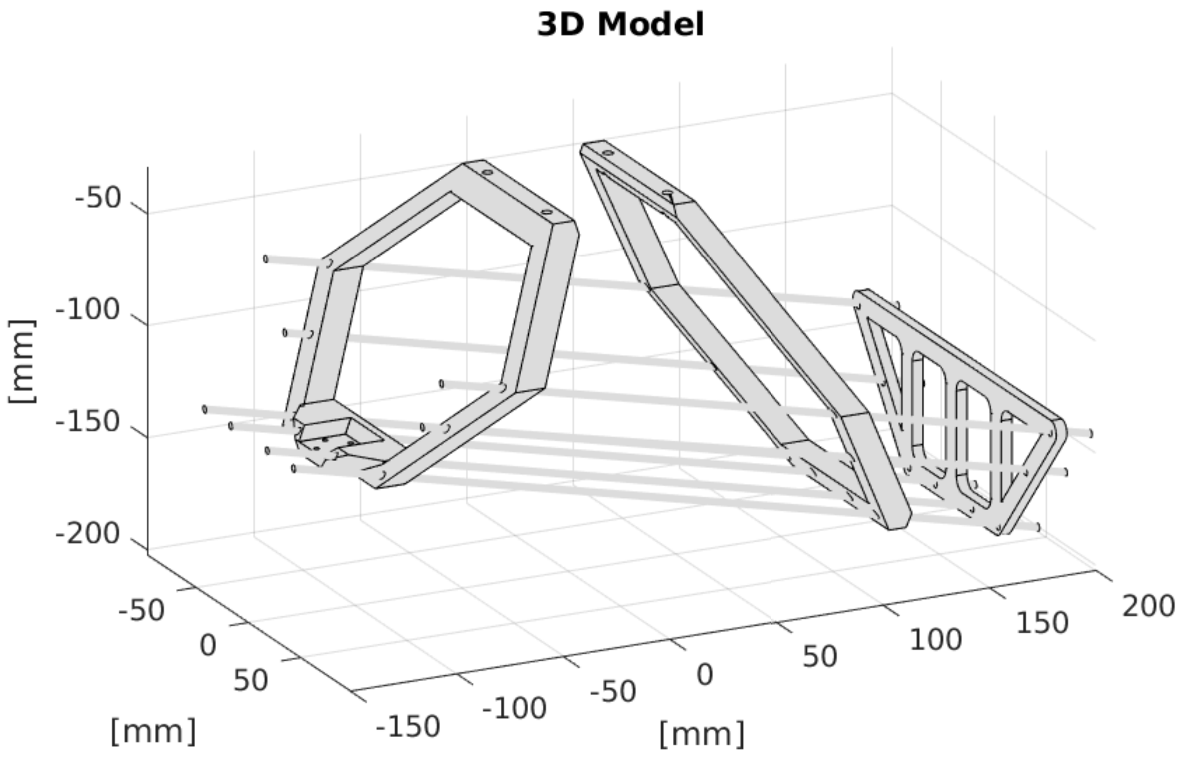


Figure 2.10: Bag holder

2.5.3 Robotic Arm

I designed a robotic arm in Fusion 360. In this subsection, I describe the process. Firstly I downloaded the STL (file format) files which represent individual components used in the robotic arm. The servo model was taken from the official Dynamixel web page. [16] For putting the whole arm together I, used components from robot kit BIOLID. Models from this kit were taken from the Dynamixel download page as well. [17] I used these components because it was the easiest and fastest way to create the robotic arm that fits together very precisely. This composition also complies with the idea that the manipulator must be as small and compact much as possible. The 3D render can be seen in picture 2.11.

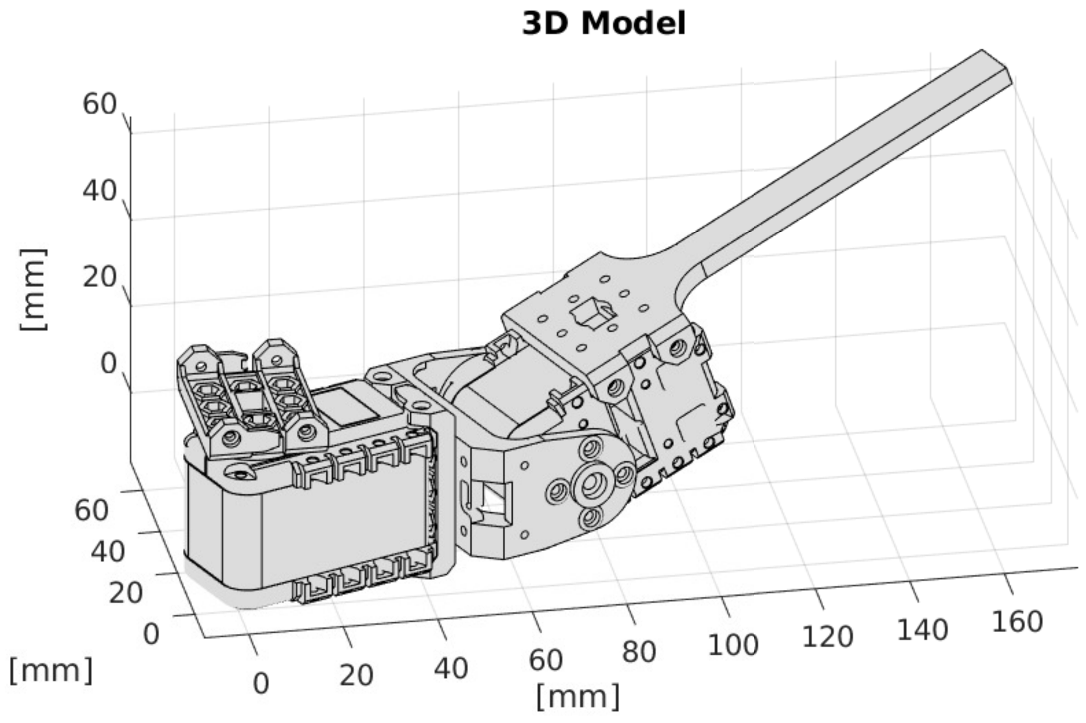


Figure 2.11: Manipulator

Chapter 3

Manipulator

3.1 Manipulator Description

1. A manipulator can be described as an open kinematic chain which can be explained by an acyclic graph.
2. The manipulator has 2 degrees of freedom (DOF).
3. Both degrees of freedom are represented as rotational joints, where the first one can manipulate in angle called β and the second one can manipulate in angle called α.

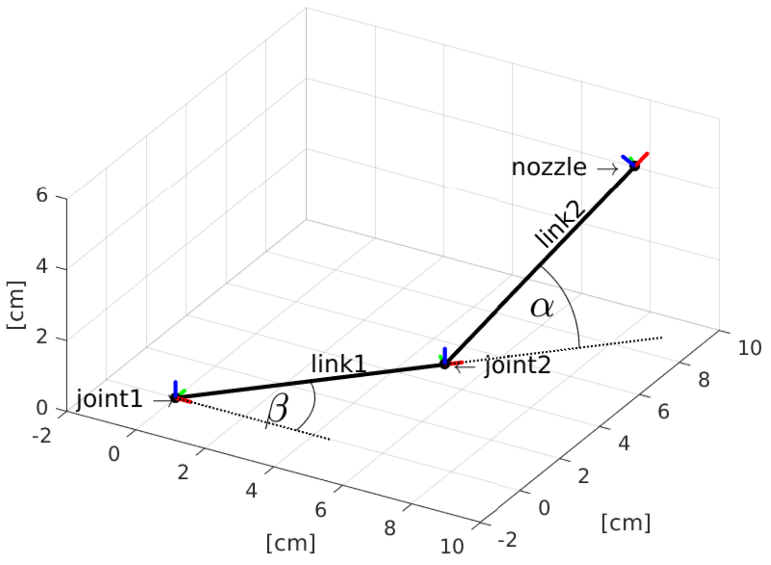


Figure 3.1: Scheme of a manipulator

In figure 3.1 you can see scheme of a manipulator with coloured axes. Red arrow represents x-axis, green arrow y-axis and blue arrow z-axis. In the picture you can see a joint1 which represents the first servo. This joint is connected with the joint2 which represents the second servo. This connection consists of a link1. A link2 is an arm which holds a nozzle on its end. Proportions of the manipulator:

- link1 = 7 cm
- link2 = 14.5 cm

3.2 Inverse Kinematics Task

Inverse kinematics is a mathematical enumeration of finding angles in individual rotation joints and finding a shift in individual linear joints from knowing the end-point position in its coordinate system. Inverse kinematics is often used in the robotic industry and I am also building a robotic arm. Because I have only rotatory joints, I just need to calculate two angles. These angles will be set on the servos.

The end-point represents the point on the wall, where the fire is detected by sensors. I am dependent on the sensors post-processing which gives me x, y and z coordinates of the end-point. x means the short distance between the drone and the wall, y means deflection from drone direction and z represents height which the water jet must reach to reach the end-point. For a better understanding look at picture 3.2, where the wall distance is 3 meters and fire is located at 1 meter height and deflected 0.4 meters. Image from the top view is in figure 3.3a and image from the side view is in figure 3.3b.

So I am searching for two angles where one of them is angle α that sets the rotation on the second servo (counting from drone construction) which regulates the vertical inclination. The second angle is named β and regulates horizontal inclination on the first servo.

3.2.1 Calculation of Angle Beta

The first angle I need to calculate is angle β - it sets the horizontal inclination and I can get it from knowing only two coordinates which are coordinate x and coordinate y. Angle β is given by the mathematical formula

$$\beta = \arctan\left(\frac{y}{x}\right) \tag{3.1}$$

where x is distance and y is a deflection of the wall on which the fire is located .

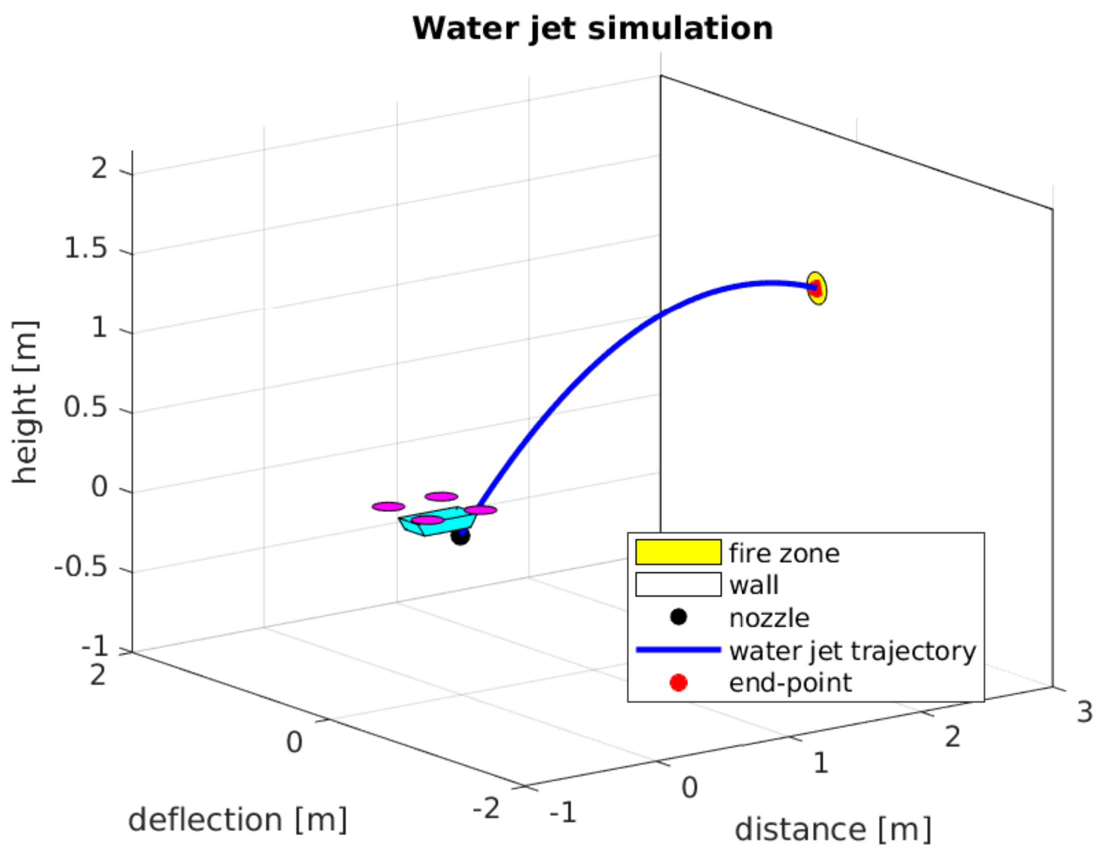
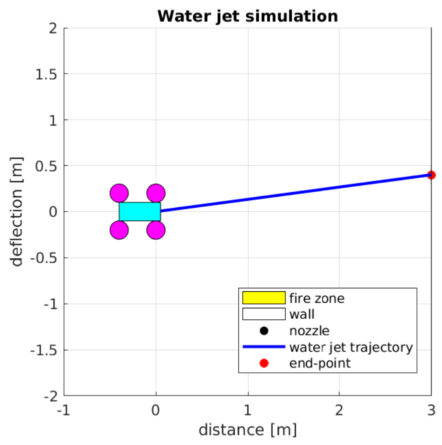
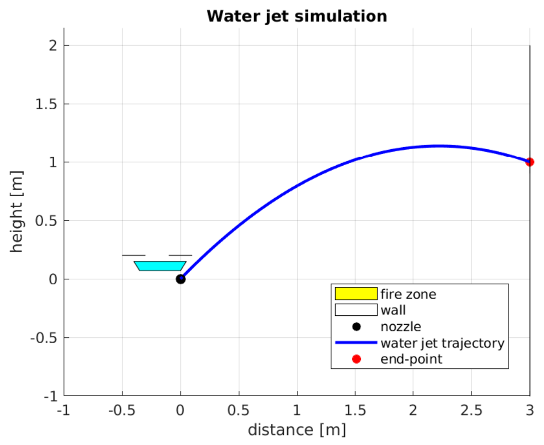


Figure 3.2: 3D simulation water jet



(a) Top view



(b) Side view

Figure 3.3: Different views to water jet simulation

3.2.2 Calculation of Angle Alpha

To calculate the second angle α I need to get the distance to the end-point. That is given by the mathematical formula:

$$d = \sqrt{x^2 + y^2} - link1 \tag{3.2}$$

where d is squirting distance. There comes the idea not to use link2 size. Length of link2 changes the distance dimension based on different α angles, therefore the equation which I tried to enumerate was unnecessarily hard. Taking into account that the stream of water will be under the air flow from propellers, the issue of distance variation is insignificant.

Because I do not know how fast a stream of water is leaving the nozzle I need to calculate it from already known volumetric flow rate Q_V and circular cross-section area of the pump tube. Speed of water on the output of the pump is given by this mathematical formula:

$$v_1 = \frac{Q_V}{S} \tag{3.3}$$

There I am able to calculate Q_V which is also mentioned in table 2.2, but in hours, and I need to transform it to elementary units, which is seconds.

$$Q_V = \frac{dV}{dt} \tag{3.4}$$

By substitution I get

$$v_1 = \frac{\frac{dV}{dt}}{S} \tag{3.5}$$

I have got output speed from the pump. But that is obviously not enough because the plastic tube is attached to the pump which means slide resistance by chafing. Since I used only a small piece of the tube and I am considering ideal fluid I decided to leave the resistance out. On the other hand, I can not ignore narrowing of a nozzle. Here I can use the continuity equation in differential form 3.6 or integral form 3.7:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{3.6}$$

integral form:

$$\iint_S \rho \vec{v} d\vec{S} + \frac{\partial}{\partial t} \iiint_V \rho dV = 0 \tag{3.7}$$

for vector $\vec{v}$ perpendicular to the surface S

$$\iint_S \rho \vec{v} d\vec{S} = -(\rho_1 v_1) \iint_{S_1} dS_1 + (\rho_2 v_2) \iint_{S_2} dS_2 = -\rho_1 v_1 S_1 + \rho_2 v_2 S_2 \tag{3.8}$$

now I can substitute equation 3.8 to 3.7.

$$\rho_1 v_1 S_1 - \rho_2 v_2 S_2 = \iiint_V \frac{\partial \rho}{\partial t} dV \tag{3.9}$$

adjustment for stable flow

$$\rho_1 v_1 S_1 = \rho_2 v_2 S_2 = konst \tag{3.10}$$

Thinking ideal liquid the equation will be simplified to:

$$\begin{aligned} v_1 S_1 &= v_2 S_2 \\ v_2 &= \frac{v_1 S_1}{S_2} \end{aligned} \tag{3.11}$$

where v_2 is output speed from the nozzle and also output speed of water which I use to calculate projectile motion. Because v_2 can be expressed as a sum of vertical and horizontal speed as shown in the following equations 3.12

$$\begin{aligned} d &= v_0 t \cos(\alpha) \\ h &= v_0 t \sin(\alpha) - \frac{1}{2} g t^2 \end{aligned} \tag{3.12}$$

where v_0 matches v_2 velocity, d means squirting distance and h is height.

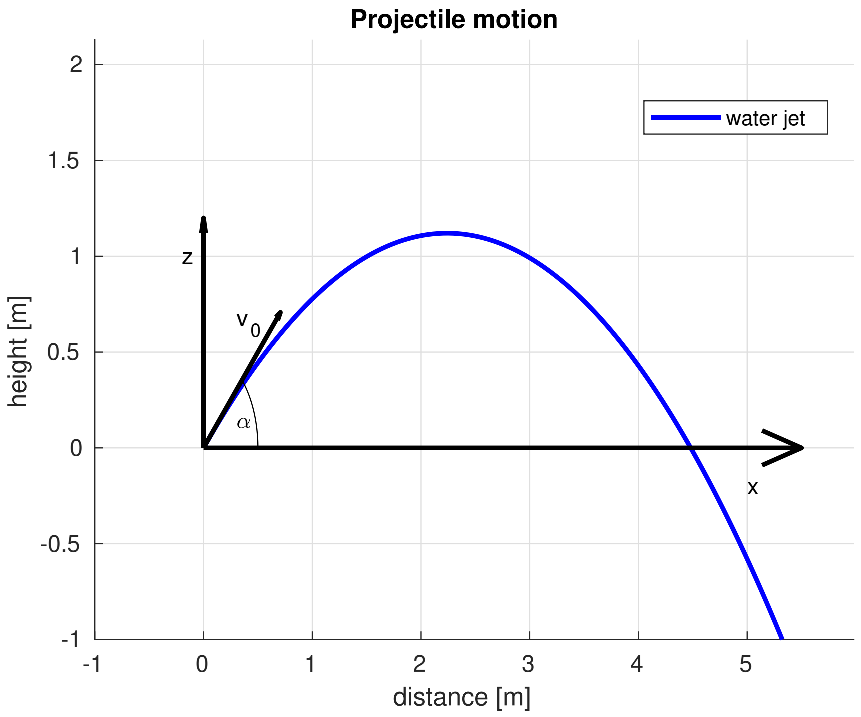


Figure 3.4: Graph of projectile motion

I have already enumerated the squirting distance. For the purpose of control, I can enumerate a maximal distance that my device is able to squirt, but that gives me only horizontal distance, that means I can squirt further but the end-point will be under the drone in the horizontal axis, which is in my cases z-axis. The maximal distance the device is able to squirt in the horizontal axis is for angle $\alpha = 45°$ upwards measured from x-axis and is enumerate in next formula:

$$d_{max} = \frac{v_2^2}{g} \sin(2\alpha) \tag{3.13}$$

If I want to calculate the angle for a distance which is at the same height as the nozzle I can use the following equations.

$$d = \frac{2v_0^2}{g} \sin(\alpha) \cos(\alpha) \tag{3.14}$$

So I can use double-angle formula $2\sin(\alpha)\cos(\alpha) = \sin(2\alpha)$ and substitute.

$$\frac{dg}{v_0^2} = \sin(2\alpha) \tag{3.15}$$

Now I get the angle α.

$$\alpha = \frac{\sin^{-1}\left(\frac{dg}{v_0^2}\right)}{2} \tag{3.16}$$

I get the angle α but, as I said, that is not enough for me, because If you imagine that the end-point is not at the same height as the nozzle it will be a problem. So I decided to go back to the common formula and enumerate it universally. I used equations 3.12. I took at the first one, expressed time t and after that I substituted it to the second equation:

$$h = \frac{v_0 d \sin(\alpha)}{v_0 \cos(\alpha)} - \frac{gd^2}{2v_0^2 \cos^2(\alpha)} \tag{3.17}$$

By simplification, I get the next formula.

$$\begin{aligned} h &= d\frac{\sin(\alpha)}{\cos(\alpha)} - \frac{gd^2}{2v_0^2}\frac{\cos^2(\alpha) + \sin^2(\alpha)}{\cos^2(\alpha)} \\ h &= d\tan(\alpha) - \frac{gd^2}{2v_0^2}(1 + \tan^2(\alpha)) \end{aligned} \tag{3.18}$$

I use substitution $\tan(\alpha) = a$ and convert to the quadratic formula

$$\frac{gd^2}{2v_0^2}a^2 - da + \left(\frac{gd^2}{2v_0^2} + h\right) = 0 \tag{3.19}$$

From this equation can be enumerated a discriminant D.

$$D = d^2 - 4\frac{gd^2}{2v_0^2}\left(\frac{gd^2}{2v_0^2} + h\right) \tag{3.20}$$

$$a_1, a_2 = \frac{d \pm \sqrt{d^2 - 4\frac{gd^2}{2v_0^2}\left(\frac{gd^2}{2v_0^2} + h\right)}}{\frac{gd^2}{v_0^2}} \tag{3.21}$$

In the end, I have got two solutions for angle α so I needed to decide which one is better for me. At this point I can expect that the smaller angle means shorter water trajectory. Choosing the smaller angle is also better for avoiding collision with drone propellers because the nozzle is set up under them.

$$\begin{aligned}\alpha_1 &= \arctan(a_1)\\ \alpha_2 &= \arctan(a_2)\end{aligned} \tag{3.22}$$

For this enumeration, I used prepared minimisation functions in python and in Matlab, which choose the smaller angle.

$$\min(\alpha_1, \alpha_2) \tag{3.23}$$

3.3 Reachable Area

3.3.1 Safety Parabola

Safety parabola or parabola of safety is a vertical parabola that marks out a boundary line of the reachable area which is group of points from every projectile motion trajectories in different elevation angles. That means the robotic extinguisher can shoot only in a zone which is located under the safety parabola. To find this parabola I looked for one solution of equation 3.20 when the discriminant $D = 0$ which as a result has one real double root.

$$\begin{aligned} D = d^2 - 4\frac{gd^2}{2v_0^2}\left(\frac{gd^2}{2v_0^2} + h\right) &= 0\\ d^2v_0^2 - \frac{g^2d^4}{v_0^2} - 2gd^2h &= 0\\ h &= \frac{v_0^2}{2g} - \frac{gd^2}{2v_0^2}\end{aligned} \tag{3.24}$$

This enumeration is used for control whether the end-point is reachable or not. Based on the result the pump is switched on or off. If the end-point is not in the reachable area, the Inverse Kinematics enumeration is terminated until the end-point is in reachable area. A graph is displayed in figure 3.5.

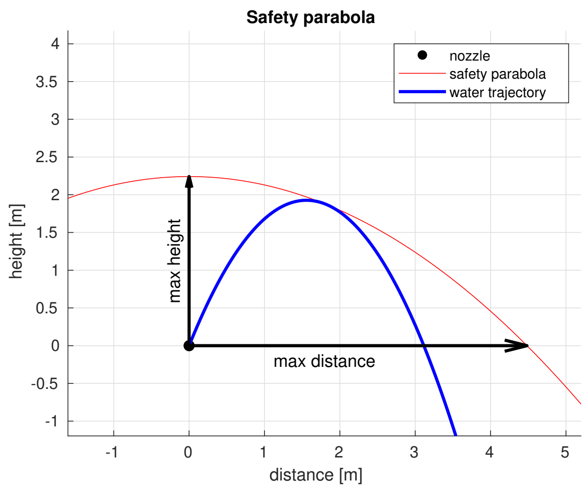
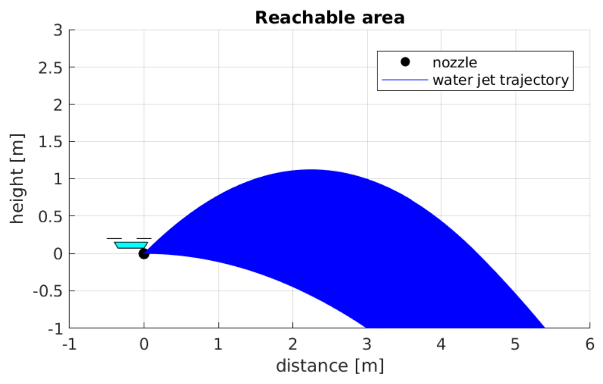


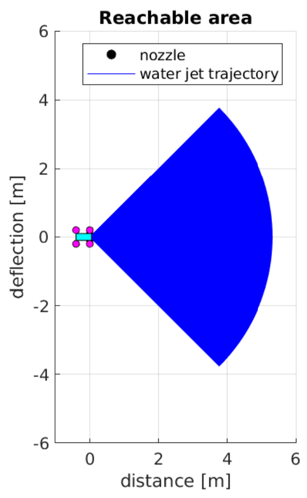
Figure 3.5: Graph of safety parabola

3.3.2 Area Restricted by Drone

Because the robotic arm is located under the drone and choosing the smaller angle alpha is not enough to prevent further collision with propellers I must set certain restrictions on angles which you can see in picture 3.6b and 3.6a. In the first one you can see the maximal distance and also a restriction on setting the angle β which is set to ± 45 degrees where the degree zero is in forward drone direction. In picture 3.6a you can see how the reachable area looks like from side view, where I set the restriction on angle α from -10 degrees to 45 degrees and the degrees are calculated from the x-axis.



(a) Side view



(b) Top view

Figure 3.6: Reachable area

Chapter 4

Software

One of the most important things in my thesis is software. Particularly the control programs, that were created to regulate the manipulator. Before describing the programs themselves I decided to introduce development tools that I used for programming and debugging, because these tools were published recently and are not widely known. After that, I shortly introduce two types of serial communication, which I used for my thesis. In DYNAMIXEL section I outline the basis of Dynamixel protocol which was used to control the servomotors. Afterwards I wrote about utilisation of Dynamixel wizard tool, which I used for controlling the right settings on servomotors and for operation verification. The rest of this chapter is devoted to programs. These control software consist of two component - there is a program on main computer which communicates with a second program on the microcontroller. There is another program, Matlab, mentioned because I used it as a simulation program with ability to create pictures from static visual simulations.

4.1 STM32Cube

STM32Cube, or just Cube, is a set of tools developed by STMicroeletronics company, which specialises on development and distribution of 32-bits microcontroller integrated circuits. To the Cube belong tools like STM32CubeIDE, STM32CubeMX and others which I do not talk about because I do not need them for this solution. I used this development tool because it uses HAL libraries which are good for fast and easy peripheries and microprocessor configuration.

4.1.1 STM32CubeIDE

This tool is designed for writing the code itself and then its debugging. The code can be written either in C or in C++. The C language was sufficient for this project. This development tool is based on Eclipse framework and it therefore allows integration of many existing plugins that can make the programming easier and more transparent. There is also a Build analyser in this development tool, which helps the developer to have control over the code size. Then, using this software, the code can be easily uploaded into the microcontroller as well. In this tool there is also a very powerful tool implemented. It has the ability to generate functional code for chosen type of STM32 microcontroller. This tool is called STM32CubeMX and it is described in its own section bellow. [18]

4.1.2 STM32CubeMX

This powerful tool is able to generate graphic representation model on which different properties can be set on peripheries or microcontroller. After the required values are set the program can generate, with a code generator, functions and structures from HAL libraries which ensure required settings. It also sets up required communication. For example I used this tool to generate code for USB and USART peripheries. [19]

4.1.3 Hardware Abstraction Layer (HAL)

HAL library is a complete set of future oriented application programming interfaces (API), which ensures high portability among different types of STM microcontrollers therefore I used HAL library for my task as well. One of its advantages is it hides peripheral and microcontroller complexity thus the HAL is good for end user who does not need to know the protocols in detail. On the other hand, disadvantages are slow speed and size of the HAL libraries but that is understandable given the compatibility for so many types of controllers. In spite of its size the HAL library is a powerful tool, mainly because run time failure detection and allowing multi instance driver layers belong to the HAL. Moreover there is also Mbed, which is another development platform that includes HAL. [20]

4.2 Data Transmission

4.2.1 Universal Serial Bus (USB)

USB is an asynchronous serial communication protocol based on Master-Slave technology and is used for connecting peripheries to computer. The USB consists of host, hubs and ports. It supports different types of transmissions with different transfer speed, for example, last generation of USB (USB4) provides 40 Gb/s transmission speed.

4.2.2 Half Duplex UART

Half duplex UART provides communication between two devices in both directions using one wire. But the data can only be send in one direction at a time. Before replying, the transmitter must stop sending data and send a message declaring that transmitter has stopped sending data and started listening.

4.3 Robot Operating System (ROS)

MRS group uses Robotic Operating System on their multi-rotor drones therefore I needed to implement my code into it. ROS is a software environment for programming robots that was created at Stanford university in 2007. The ROS's functionality is connecting small subsets of tasks together. These smaller tasks are called nodes, there are usually pieces of software that can be written in C++ or Python. That gives developers a way to easily divide their programs into small parts. For example, one node can read sensor data and another can set the servo position. These nodes are connected together using publish subscribe protocol. The node which has an information to share uses the topic. Another node which is interested in that information subscribes to that topic. Its compatibility protocol enables cooperation with international teams on the same project. For example I created a ROS node which subscribes to the node that publishes a vector with information about fire location. The vector was based on sensor data by another team member from MRS group.

4.4 DYNAMIXEL

4.4.1 Dynamixel Protocol 1.0

Dynamixel protocol 1.0 is one of Dynamixel Protocols, there is a Dynamixel Protocol 2.0 as well. As I said in section 2.1.2 the communication is provided by half-duplex which was explained in section 4.2.2. Based on the type of communication we can distinguish two types of packets. First one is sent from microcontroller to the servo and is called Instruction Packet. The second one serves as an answer to the first one and is called Status Packet. [21]

Instruction packet

Structure of instruction packet is shown in table 4.1.

H1	**H2**	**ID**	**Length**	**Instruction**	**Param 1**	**...**	**Param N**	**Checksum**
0xFF	0xFF	`ID`	`Length`	`Instruction`	`Param1`	...	`ParamN`	`CHKSUM`

Table 4.1: Instruction packet

Status Packet

Status packet is sent from servo as an acknowledgement to the Instruction Packet. The Status Packet can be seen in table 4.2.

H1	**H2**	**ID**	**Length**	**Error**	**Param 1**	**...**	**Param N**	**Checksum**
0xFF	0xFF	`ID`	`Length`	`Error`	`Param1`	...	`ParamN`	`CHKSUM`

Table 4.2: Status packet

4.4.2 Dynamixel Wizard 2.0

Dynamixel Wizard is a program created under the manufacturer ROBOTIS and it is built to handle Dynamixel protocols. It has a graphic interface, thus I can check settings which were set via STM32 microcontroller or I can directly set the required parameters. This software is also able to create some graphs, manipulate with servomotors with graphic intuitive interface and has a recovery tool for servomotor firmware.[22] If you want to connect the Dynamixel servomotor to the Dynamixel wizard on your PC then you will need a USB2DYNAMIXEL[23] connector.

4.5 System Description

Picture 4.1 shows application's diagram. The first box called Sensors means that my application depends on data from sensors. This data must be processed and must be in a vector format, where the first dimension means distance, second deflection and third height of the fire location. But creating this vector from raw data is not the subject of this thesis. Therefore I will use the vector which will be published from a ROS node created by another team member. For testing I created a ROS node which generates random fire location and publishes it.

In my thesis I decided to use two different programming languages to create application software. Firstly I programmed Inverse Kinematics in Python which I chose for its simplicity. This part of application software runs on drone computer which gives me sufficient performance for enumerating more complex calculations like Inverse Kinematics Task which was mentioned and mathematically derived in section 3.2. The enumeration is based on data which comes from post-processed sensor data. This data is published from the ROS node, therefore this program is another ROS node called listener. This node listens the node which publishes the vector. Afterwards the vector is processed, the Inverse Kinematics Task is enumerated, and calculated values are sent through serial link.

The second part of application software was programmed in C language. This language was used because the code runs on MCU that is commonly programmed using this language. Moreover, I used MCU from STM32 and I wanted to use HAL libraries for peripherals control. This part of application receives data from an on-board computer because the Python program is transmitting messages which contain the information about how to set servomotors. The message is than processed and sent in the correct format with half duplex UART into servomotors.

When servomotors receive the Instruction Packet, the data is processed and stored in RAM or EEPROM memory. Based on the Instruction Packet type, servomotors are moved or just have their parameters set. After that, the servomotors send the Status Packet.

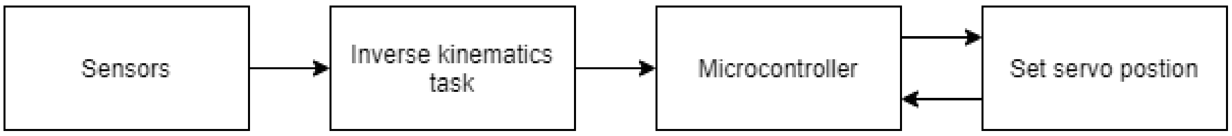


Figure 4.1: Overall programs scheme

4.5.1 IKT Implementation

Because the Inverse Kinematics itself was already mentioned in separate section 3.2 I only point out that the program is divided into functions which are shown in programming flowchart in picture 4.2. Lots of functions are only rewritten mathematical enumerations, therefore I do not mention them again. Moreover, the code can bee seen in attached programs.

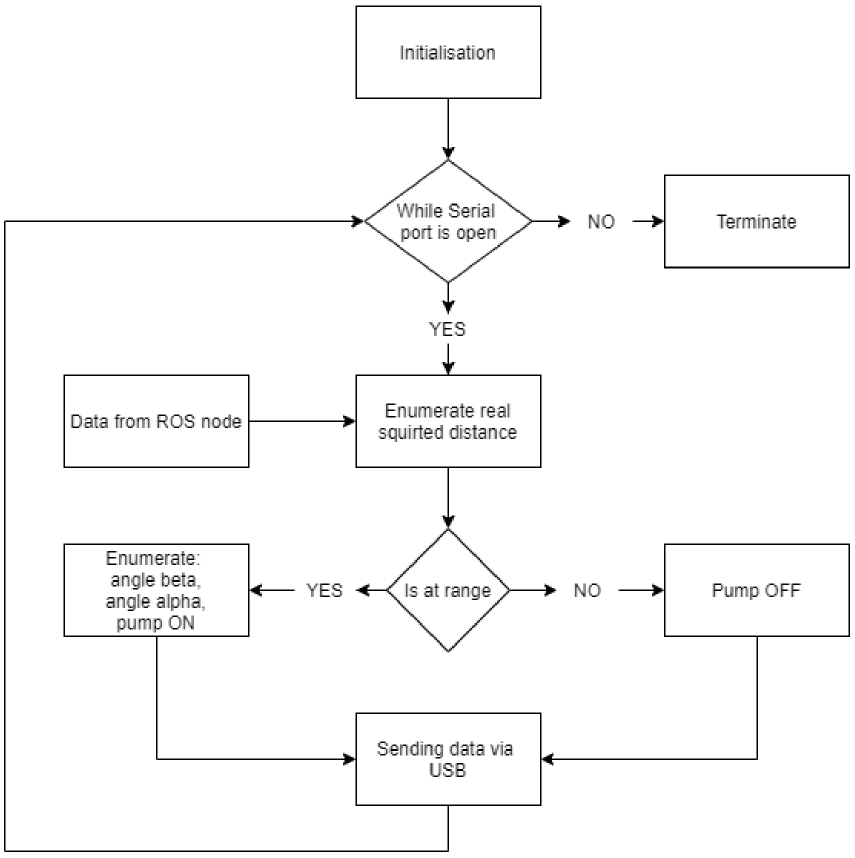

Initialisation
While Serial port is open
NO
Terminate
YES
Data from ROS node
Enumerate real squirted distance
Enumerate: angle beta, angle alpha, pump ON
YES
Is at range
NO
Pump OFF
Sending data via USB

Figure 4.2: Programming flowchart

Sending Data via USB

The last block called Sending Data via USB needs to be mentioned, because I created my own protocol which can be seen below. The data is being sent in a string format, therefore I can send `alpha`, `beta` and `pump` values in one message. The individual data is separated by backslashes. Before the message creation, angles (in degrees) are divided by 0.29 which is the resolution unit for the servo and is stated in the table 2.1. To avoid float enumeration on microcontroller I also converted these numbers into integer format. The variable `pump` is set to 1 or 0, where the 1 means the pump is ON and the 0 means the pump is OFF. As the last step of the creation process protocol the data is converted to string format. Sending the data is accomplished with serial port.

$$\texttt{alpha} \setminus \texttt{beta} \setminus \texttt{pump}\setminus$$

4.5.2 MCU Firmware

The application software on MCU has its layout shown in programming flowchart 4.3. Firstly, initialisation that sets servomotors into the right setting, which is mentioned in a separate section bellow, is performed. After that, the main infinity loop runs and asks in every iteration whether the queue is not empty.

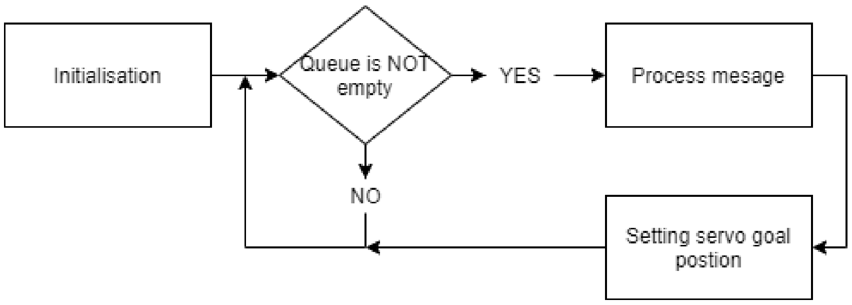


Figure 4.3: Program on microcontroller

Initialisation

In initialisation, Clockwise Angle limit, Counter-Clockwise Angle limit, Torque limit and Goal position are set on the servomotors. This servo initialisation is performed with the DYNAMIXEL protocol which was mentioned in section 4.4.

1. Clockwise Angle limit and Counter-Clockwise Angle limit
   These two angle limitations restrict angle range that the servo can reach. Because as I wrote in section 3.3.2 the angle must be limited for safe drone operation.

2. Torque limit
   To make the movement between two positions on the servo more smooth I decided to use torque limit reduction. When the servo is moving slower it decreases strength of vibrations.

3. Start goal position
   This sets servomotors to starting positions.

Process Message

The microcontroller is waiting for a message which is being received on the USB periphery. After the message comes it is inserted into the queue. There can the "Queue is not empty" condition be set to true. The data from the message is translated into specific angles (`alpha`, `beta`) and information about the pump (the pump is controlled with microcontroller uses N-Mosfet). Because the speed of MCU is higher than the frequency of the data received, there is a maximum of one message in a queue at a time. This ensures there is always the latest available sensor data being processed.

Setting Servo Goal Position

Setting Servo Goal Position is the most common task of the microcontroller therefore I decided to show how it works in detail. Especially how to create the Instruction packet as mentioned in table 4.1. The `Header1` and the `Header2` are both strictly given by the protocol. `ID` is given by the servo on which the goal position is set, thus I chose one of my servomotors, a servo with ID 20. At this moment I skip `Length` because it is based on the packet size, which is sum of variables without `Header1`, `Header2`, `ID` and `Length`. The data needs to be written into the servo therefore the instruction is ”Write” and number for this instruction type is 0x03. As a first parameter there is memory location which is number 30 (0x01E) for this address. Because the highest number for the goal position is 1023, the memory size to store this number must be 2 bytes and the number is written into the memory with Little-endian layout. For example, the number 512 (0x200), which means the centre position, is sent with instruction packet 4.3 as shown bellow. The checksum is given by following formula

$$\text{CHECKSUM} = \sim (\texttt{ID} + \texttt{Length} + \texttt{Instruction} + \texttt{Param1} + \texttt{Param2} + \texttt{Param3})$$

where the $\sim$ is the bitwise NOT.

H1	**H2**	**ID**	**Length**	**Instruction**	**Param1**	**P2**	**P3**	**Checksum**
0xFF	0xFF	0x14	0x05	0x03	0x1E	0x00	0x02	0xC3

Table 4.3: Goal position instruction packet

After creating Instruction packet the packet is sent via half duplex UART to the servo.

4.6 Matlab

I used Matlab version 2018b. Because I am not familiar with the Gazebo simulation program, which is commonly used in MRS group, I decided to use Matlab to do some simulations. Specifically it was position simulations and reachable area check simulations. Most of the figures I created for my thesis were designed in Matlab, therefore I attach a code example of how to convert STL file into Matlab plot 4.1.

```matlab
function stl_to_plot(file_name)
    hold on
    model = createpde;
    importGeometry(model,file_name);
    pdegplot(model);
    %% delete red axis in plot
    delete(findobj(gca,'type','Text'));
    delete(findobj(gca,'type','Quiver'));
    title("3D Model")
    xlabel('[mm]');
    ylabel('[mm]');
    zlabel('[mm]');
    grid on
    view(3)
end
```

Listing 4.1: Matlab code example

Chapter 5

Real life experiments

The experiments were carried out during the whole process of the fire extinguisher creation. Firstly I tested two different pumps with different nozzles to find out which pump is better and which nozzle size (its hole radius) fits better. The experiment is captured in picture 5.1.

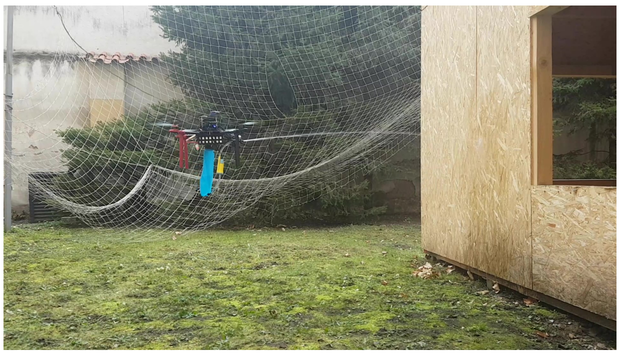
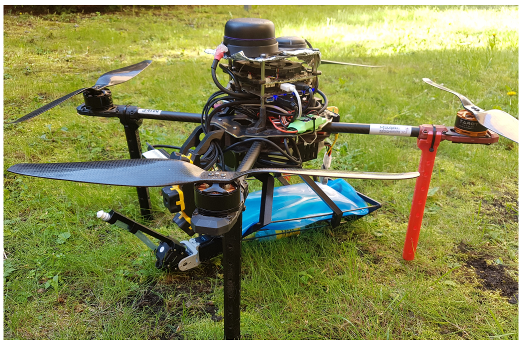


Figure 5.1: Pump (Comet) and nozzle test on the drone without manipulator and bag holder

Based on the experiment was chosen the powerful pump (Comet) and the nozzle with a 1.5 mm hole radius. After that, I tested the manipulator itself without the drone. I tested it on a stand which is captured in figure 5.3. During these experiments, I verified the functionality of the Inverse Kinematics Task. After that, I attached the bag holder, the manipulator, the pump and also the water storage on the drone. A photo of the completed robotic fire extinguisher mounted on the drone is in picture 5.2 and a photo taken during the experiment is in picture 5.4. The drone on this experiment was controlled manually. The fire position was randomly generated because we do not have test fire available. Based on that experiment was verified the functionality of the manipulator. Based on observation, the manipulator is able to point the nozzle in the desired direction to reach with the water stream entered position.


(a) UAV with mounted fire extinguisher

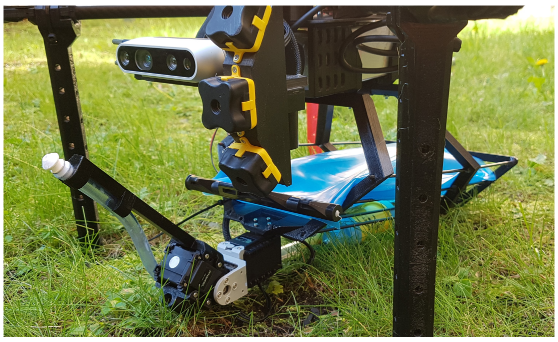

(b) Mounted fire extinguisher detail

Figure 5.2: Fire extinguisher mounted on UAV

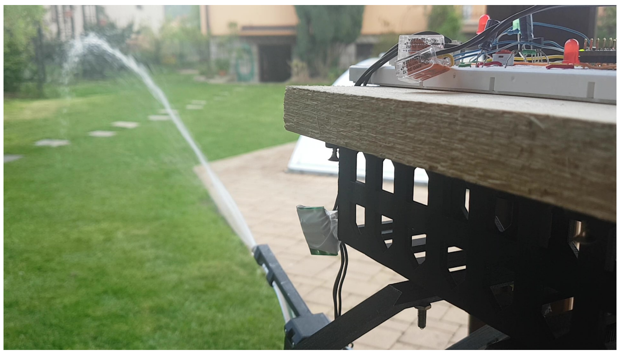


Figure 5.3: Robotic fire extinguisher on stand

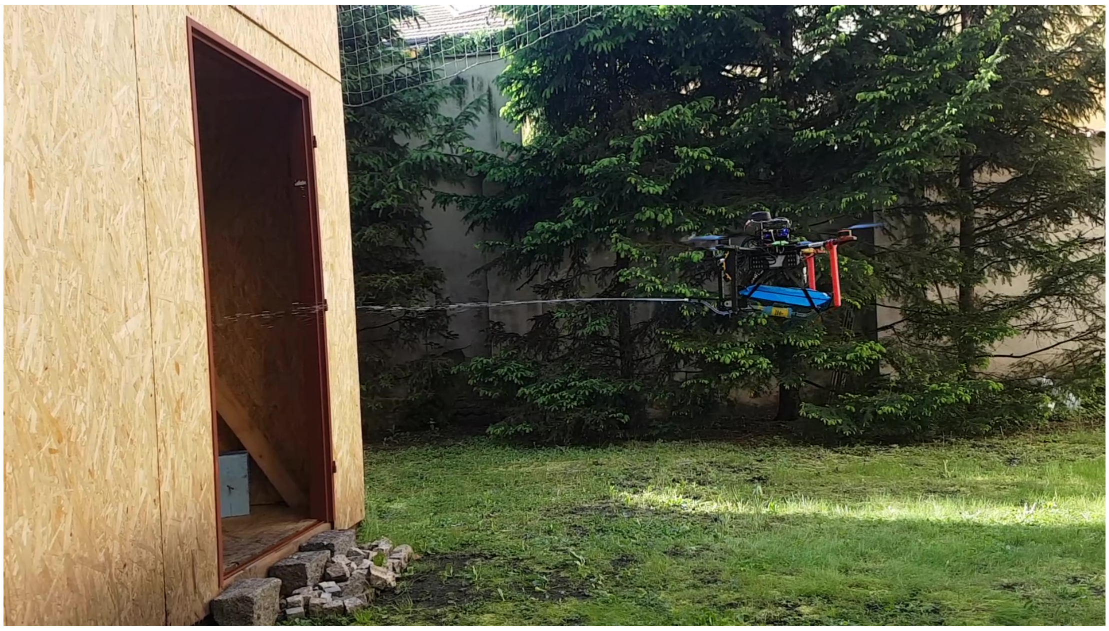


Figure 5.4: Robotic fire extinguisher mounted on the drone during experiment

Chapter 6

Conclusion

The goal of this thesis was to develop hardware and software for robotic fire extinguisher mounted on the small unmanned aerial vehicle. The developed fire extinguisher was created for the MBZIRC competition. The fire extinguisher consists of the manipulator, water pump and also on-board water storage. All of those parts are attached to the drone using a bag (storage) holder that was also designed in this thesis. Maximal amount of water the prototype was tested to carry is 1.2 litres.

Software, which was developed to control the fire extinguisher, is also presented in this thesis. Software, in this case the system control is split into Python program and MCU firmware. Python program was used to create a ROS node. This node subscribes to the ROS topic with fire position. Due to the enumeration complexity the Inverse Kinematics Task, this process was implemented in Python program and runs on the main computer which has sufficient computing capabilities. Results are then sent over the serial link (via USB) to the MCU firmware which runs on STM32 microcontroller. This part of control software provides servomotors' control based on the messages from the Python program.

Before performing real-life experiments, the conceptual design was tested in Matlab. Matlab programs which were used to simulate the manipulator can be found in the attachments. Inverse Kinematics Task integrated into Matlab can also be found in attachments. Some presented graphs were also created using Matlab. After that, the extinguisher was tested in real experiments but without the fire. Therefore to verify the functionality of the extinguisher and to verify if the manipulator is able to point the water nozzle in the desired direction for the water stream to reach the required area, a random position generator to simulate the fire position was created. Based on the observations, the robotic manipulator was able to point the nozzle to the desired generated position.

6.1 Future work

This conceptual design is ready to be tested for its autonomy. The drone is able to fly autonomously as well as autonomously detect fire. Therefore in future experiments, the fire extinguisher should be also tested with real fire. Furthermore, some control system improvements could be made. For example, bidirectional communication between the microcontroller and the main computer could be implemented. And adding the possibility to change the servomotors settings from the main computer. It might also be interesting to see this design tested on a larger scale (using a bigger drone).

Bibliography

[1] ROBOTIS. (2020, Mar.) e-manual. Robotis website. [Online]. Available: http://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

[2] I. T. S. T. U. D. I. O. s.r.o. (2020, Feb.) Venkovní průtokové čerpadlo 24v comet 551273. SIGMAshop.CZ. [translated]. [Online]. Available: https://www.sigmashop.cz/cerpadla-a-cerpaci-technika/venkovni-prutokove-cerpadlo-24v-comet-551273

[3] DECATHLON. (2020, Apr.) Hydrovak trek-500 2l modrý. Decathlon e-shop. [Online]. Available: https://www.decathlon.cz/hydrovak-trek-500-2-l-modry--id_8493245.html?gclid=Cj0KCQjw6sHzBRCbARIsAF8FMpVmIj_JtCF_xJPz2xBpS061VDV0ZnOff1qsHNOYHYGWtiWenzW1jUQaAtQsEALw_wcB

[4] Mbed. (2020, Apr.) Nucleo-f042k6. armMBED. [Online]. Available: https://os.mbed.com/platforms/ST-Nucleo-F042K6/

[5] ROBOTIS. (2020, Mar.) e-manual. Robotis website. [Online]. Available: http://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

[6] A. Cervantes, P. Garcia, C. Herrera, E. Morales, F. Tarriba, E. Tena, and H. Ponce, "A conceptual design of a firefighter drone," *2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–5, 2018.

[7] C. Manuj, A. Rao, and S. Rahul, "Design and development of semi-autonomous fire fighting drone," 2019.

[8] A. I. Alshbatat, "Fire extinguishing system for high-rise buildings and rugged mountainous terrains utilizing quadrotor unmanned aerial vehicle," *International Journal of Image, Graphics and Signal Processing*, vol. 10, pp. 23–29, 2018.

[9] M. Manimaraboopathy, H. Christopher, S. Vignesh, and P. T. Selvan, "Unmanned fire extinguisher using quadcopter," *International Journal on Smart Sensing and Intelligent Systems*, vol. 10, pp. 471–481, 2017.

[10] A. D. Gupta, Z. Bin Akhtar, M. C. Sarkar, T. Dhar, and P. Das, "Unmanned disposal rover along with fire extinguishing capacity on both ground and air," in *2019 Global Conference for Advancement in Technology (GCAT)*, 2019, pp. 1–5.

[11] G. Heredia, A. E. Jimenez-Cano, I. Sánchez, D. Llorente, V. Vega, J. Braga, J. Á. Acosta, and A. Ollero, "Control of a multirotor outdoor aerial manipulator," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3417–3422, 2014.

[12] A. E. Jimenez-Cano, J. Martín, G. Heredia, A. Ollero, and R. Cano, "Control of an aerial robot with multi-link arm for assembly tasks," *2013 IEEE International Conference on Robotics and Automation*, pp. 4916–4921, 2013.

[13] F. Ruggiero, M. A. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Perez, V. Lippiello, A. Ollero, and B. Siciliano, "A multilayer control for multirotor uavs equipped with a servo robot arm," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4014–4020, 2015.

[14] C. R. Laboratory. (2020, Jan.) Website. [Online]. Available: https://comrob.fel.cvut.cz/

[15] STM32. (2020, Mar.) Microcontroller stm32f042k6t6. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f042k6.html

[16] ROBOTIS. (2020, Jan.) Drawing dynamixel servo ax-12a. Robotis download website. [Online]. Available: http://en.robotis.com/service/downloadpage.php?ca_id=7010

[17] ——. (2020, Jan.) Drawing biolid components. Robotis download website. [Online]. Available: http://en.robotis.com/service/downloadpage.php?ca_id=7040

[18] STMicroelectronics. (2020, Apr.) STM32CubeIDE. [Online]. Available: https://www.st.com/en/development-tools/stm32cubeide.html

[19] ——. (2020, Apr.) STM32CubeMX. [Online]. Available: https://www.st.com/en/development-tools/stm32cubemx.html

[20] STM32. (2020, Apr.) Description of stm32f0 hal. [Online]. Available: https://www.st.com/resource/en/user_manual/dm00122015-description-of-stm32f0-hal-and-lowlayer-drivers-stmicroelectronics.pdf

[21] ROBOTIS. (2020, Feb.) Communication overview. Robotis e-manual. [Online]. Available: http://emanual.robotis.com/docs/en/dxl/protocol1/

[22] ——. (2020, Feb.) Dynamixel wizard 2.0 manual. Robotis e-manual. [Online]. Available: http://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/

[23] ——. (2020, Feb.) Communication overview. Robotis e-manual. [Online]. Available: http://emanual.robotis.com/docs/en/parts/interface/usb2dynamixel/

Appendices

CD Content

In Table 1 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
thesis_sources	latex source codes
src	C firmware source code
python	Python source code
models	STL files for 3D printing
matlab	Matlab plot scripts for simulations

Table 1: CD Content

List of abbreviations

In Table 2 are listed abbreviations used in this thesis.

Abbreviation	**Meaning**
CRAS	Center for Robotics and Autonomous Systems
DARPA	Defense Advanced Research Projects Agency
EEPROM	electrically erasable programmable read-only memory
FLASH	electronic (solid-state) non-volatile memory
HAL	hardware abstraction layer
I2C	inter-integrated circuit
MAV	micro aerial vehicles
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
MCU	microcontroller unit
MRS	Multi-robot Systems
SLAM	simultaneous localisation and mapping
SPI	serial peripheral interface
SRAM	static random-access memory
STL	file format for 3D systems
PETG	polyethylene terephthalate filament
PCB	printed circuit board
PLA	polylactic acid filament
RAM	random-access memory
UAV	unmanned aerial vehicles
UART	universal asynchronous receiver-transmitter
USART	universal synchronous/asynchronous receiver/transmitter
USB	universal serial bus

Table 2: Lists of abbreviations