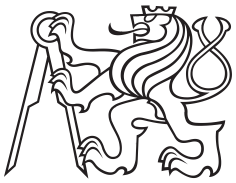
Bachelor's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Robotic Manipulator for an Unmanned Aerial Vehicle

David Štych

Supervisor: Ing. David Žaitlík
Field of study: Cybernetics and Robotics
May 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Štych David** Personal ID number: **483700**

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 Manipulator for an Unmanned Aerial Vehicle

Bachelor's thesis title in Czech:

Robotický manipulátor pro bezpilotní dron

Guidelines:

This thesis will focus on the design and development of a robotic manipulator for an Unmanned Aerial Vehicle (UAV). The task is motivated by the MBZIRC 2023 robotic competition where a group of aerial and ground robots is tasked with tracking and capturing stationary and moving objects on the surface of the water and autonomous crops harvesting. The designed system will consist of a custom robotic arm with an end-effector mounted to a UAV. Moreover, a control unit with a dedicated microcontroller shall be used to control the manipulator and the end-effector.
The thesis shall consist of the following tasks:
1) Design a robotic arm that will allow safe manipulation with objects below a UAV.
2) Design a suitable end-effector for the manipulator.
3) Implement a control system for the manipulator. Utilize the provided STM32 microcontroller.
4) Devise and implement an Inverse Kinematic Task for the manipulator.
5) Implement communication between a computer and the microcontroller in order to control the manipulator and the end-effector from the Robot Operating System (ROS).

Bibliography / sources:

[1] Carmine Noviello; Mastering the STM32 Microcontroller; Lean Publishing; 2016
[2] Anis Koubaa; Robot Operating System (ROS): The Complete Reference (Volume 3); Springer; 2018
[3] 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: **26.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. David Žaitlík	prof. 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

Acknowledgements

I would like to thank the following people who helped me to overcome the inevitable difficulties I encountered.

The thesis supervisor Ing. David Žaitlík, whose knowledge and valuable advice was essential to complete this thesis. His guidance was really helpful, especially with the PCB design.

My employer for letting me use a CNC machine to manufacture the MDF prototype.

And my friend Ondřej K. for helping me with carbon fiber machining.

Declaration

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.

In Prague, 21. May 2021

Signature:.....

Abstract

This thesis focuses on the design and development of a three-axis robotic manipulator.

The thesis gradually describes the subject matter. Firstly, the manipulator design is introduced, and used hardware is described, including the custom-made control PCB. The following chapter is focused on manipulator kinematics. Forward and inverse kinematics are solved and explained, and collision detection is introduced. In the last chapter, the manipulator control software is described.

The manipulator will be mounted under a UAV and used for various tasks motivated by the MBZIRC 2023 competition.

Keywords: robotic manipulator, UAV, end-effector, kinematics, servomotor, STM32, ROS

Supervisor: Ing. David Žaitlík
Praha, Resslova 307/9, room: E-118

Abstrakt

Tato práce se zaměřuje na design a vývoj tříosého robotického manipulátoru.

Práce postupně rozebírá danou problematiku. V první části práce je představena konstrukce manipulátoru dle požadavků a popis použitého hardwaru včetně řídící desky, která byla v rámci práce navržena. Následující část práce je zaměřená na kinematiku manipulátoru. Je vyřešena dopředná i inverzní kinematická úloha a představen způsob detekce kolizí. Poslední kapitola je věnována popisu softwarové části. Popisuje řízení manipulátoru v prostředí ROSu a firmware mikrokontroléru.

Tento manipulátor bude uchycený pod dronem a bude použitý pro různé úkoly, které jsou motivovány soutěží MBZIRC 2023.

Klíčová slova: robotický manipulátor, UAV, koncový efektor, kinematika, servomotor, STM32, ROS

Překlad názvu: Robotický manipulátor pro bezpilotní dron

Contents

Figures

Tables

Chapter 1

Introduction

This thesis is focused on designing and controlling a robotic manipulator mounted on a drone. The manipulator itself had to be designed and manufactured. The control system had to be designed along with a custom-made printed circuit board.

1.1 Motivation

The manipulator will be mounted under a UAV and can be used for lifting and carrying various items. Platform Tarot T650 (Figure 1.1) is used.

Figure 1.1: Used drone [1]

The task is motivated by MBZIRC 2023 [5] international robotics competition, where MRS group competed on multiple occasions already. There are many challenges planned for the 2023 competition [6]:

- Agriculture Applications
  - Harvesting
  - Precision Delivery
- Marine Applications
  - Surface Capture
  - Sub-Surface Capture
- Hospital Applications
  - Bed Side Assistance
  - Sterilization

The manipulator is primarily intended for *Marine Applications - Surface Capture* challenge.

1.2 Design

Chosen manipulator architecture consists of three degrees of freedom using servomotors as rotational joints. Figure 1.2 shows the joints labeling convention used throughout this thesis. Structural links were made out of carbon fiber to minimize weight. More complicated parts, which would be difficult to machine, were 3D printed. As an end-effector, an electro-permanent magnet was used. However, the magnet can be easily swapped for a different end-effector type. It is possible to add another servomotor to be used as a mechanical gripper and use the same circuit without any modifications to control it.

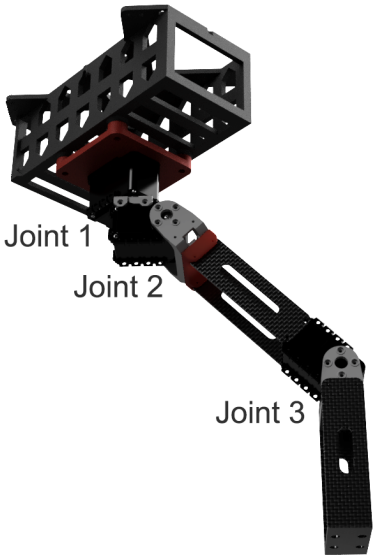


Figure 1.2: Joints labeling convention

The inverse kinematics problem of this specific manipulator has up to four solutions. The manipulator is intended to manipulate objects on the ground, so the end-effector orientation is considered when choosing the optimal solution. Also, UAV's legs, frame, and especially propellers have to be avoided. Therefore, the manipulator work region is defined, and a collision check is performed. No structural part of the manipulator is allowed to reach out of the work region 1.3. Furthermore, this defined region simplifies IKT because it usually automatically discards two solutions.

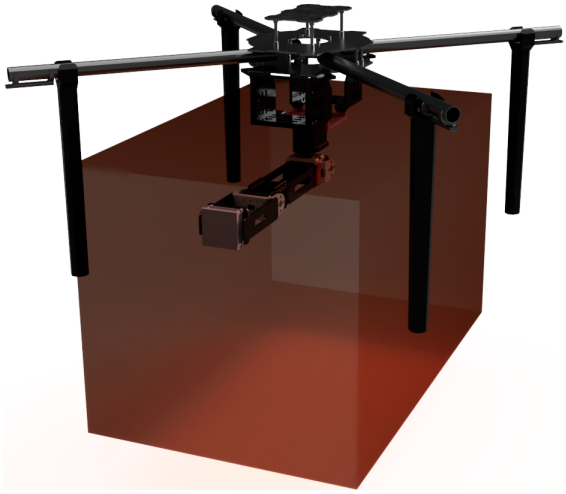


Figure 1.3: Manipulator work region

Servomotors and the end-effector are controlled using a custom-made printed circuit board based on an STM32 microcontroller. Inverse kinematics and other necessary calculations are done in ROS, which runs on the drone. Commands are then sent to the PCB using UART, interpreted, and executed by the STM32 microcontroller.

Chapter 2

Hardware

2.1 Mechanical Design

The manipulator was designed using Fusion360 CAD software. RRR manipulator structure was chosen. It consists of three Dynamixel AX-12A servomotors as revolute joints, joined together by series of links.

The first draft consisted of two single links (Figure 2.1). However, this was rejected, and additional structural parts were added in order to increase rigidity.

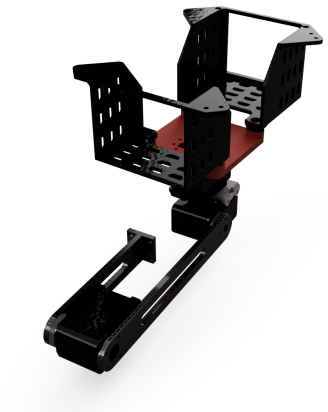


Figure 2.1: First manipulator draft

Link sizes and other dimensions of the finished manipulator are shown in the Figure 2.2, all dimensions are in millimeters. Links are held together using the mechanical components included with AX-12A servomotors. Prototype (Figure 2.3) had links made from 4 mm MDF. Other necessary structural parts (such as the mounting bracket) were 3D printed with PLA filament. Only one prototype was made and appeared to be successful, and no modifications were needed.

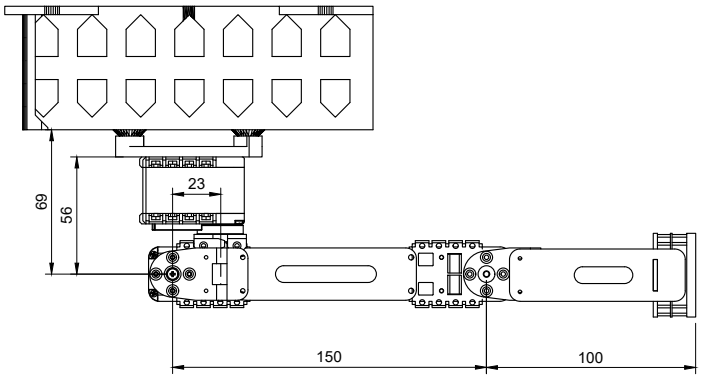


Figure 2.2: Manipulator drawing

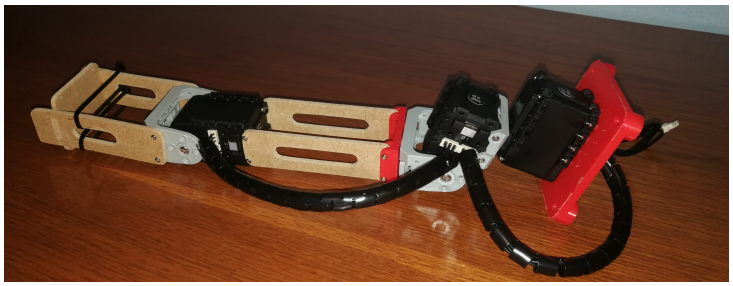


Figure 2.3: Prototype

The robotic manipulator will be mounted on a drone Tarot 650 (Figure 1.1). Therefore, low weight was a priority. In order to minimize weight, the final version (Figure 2.4) was fitted with links made from 2 mm carbon fiber. And other parts were 3D printed using a Creality CR-10 printer with PETG filament, which has good properties for such application.

Figure 2.4: Finished manipulator with the control board

2.2 Manipulator Work Region

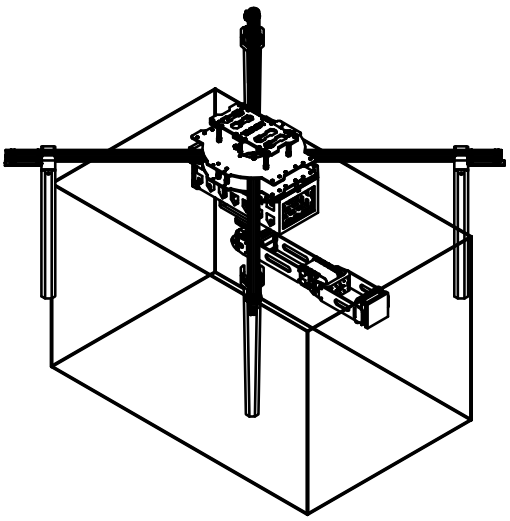
Servomotors travel limits and maximum allowed rotational speed (joints numbers are based on the convention described in the Figure 1.2):

- Joint 1
  - Travel limit: ± 180 °
  - Maximum rotational speed: 10 RPM
- Joint 2
  - Travel limit: $0 - 100$ °
  - Maximum rotational speed: 15 RPM
- Joint 3
  - Travel limit: ± 100 °
  - Maximum rotational speed: 15 RPM

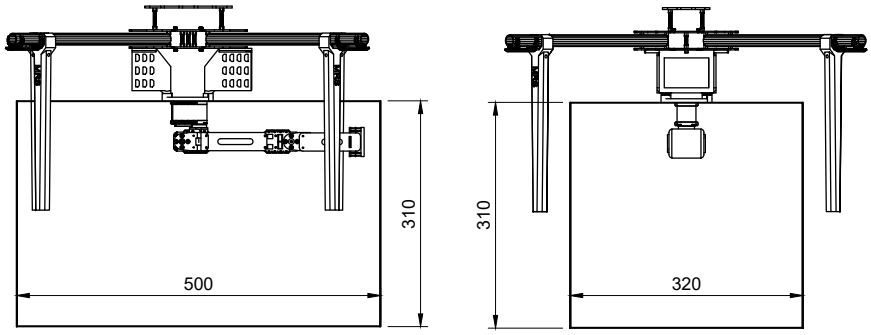
The limits on the previous page show the maximum that can be achieved before the robot collides with itself. However, collision with the drone must

also be avoided for apparent reasons. Therefore, a work region was established (Figure 2.5). It does not represent a reachable area by the robot. It simply defines a safe space where the manipulator can move. When choosing the optimal inverse kinematics solution, the position of joint 3, as well as the end-effector, is calculated using transformation matrices mentioned in section 3.2. If the end-effector position or any of the joints exceeds the allowed work region, such an inverse kinematics solution is discarded.

In case the drone's legs are moved or modified, or the manipulator is mounted on an entirely different UAV, the region can be easily modified in the manipulator control ROS node to suit the current requirements.



(a) : Manipulator workspace - isometric view



(b) : Manipulator workspace - front and side view

Figure 2.5: Manipulator workspace

2.3 AX-12A Servomotors

A servomotor is a type of motor with a closed-loop control system, which allows precise positioning of the output shaft by utilizing position feedback. Dynamixel AX-12A servomotors were used as revolute joints of the manipulator. Table 2.1 shows specifications of the used servomotors [3].

Baud Rate	*7843 bps - 1 Mbps*
Weight	*54.6 g*
Dimensions (W x H x D)	*32 mm x 50 mm x 40 mm*
Resolution	*0.29 °*
Running Degree	*0 ° - 300 °* *Endless Turn*
Motor	*Cored*
Gear Ratio	*254:1*
Stall Torque	*1.5 Nm (at 12 V, 1.5 A)*
No Load Speed	*59 RPM (at 12 V)*
Operating Temperature	*-5 °C - +70 °C*
Input Voltage	*9.0 - 12.0 V (Recommended: 11.1 V)*
Command Signal	*Digital Packet*
Protocol Type	*Half Duplex Asynchronous Serial Communication (8 bit, 1 stop bit, no Parity)*
Physical Connection	*TTL Level Multi Drop Bus*
ID	*254 ID (0-253)*
Feedback	*Position, Temperature, Load, Input Voltage*
Gear Material	*Engineering Plastic(Full)*
Case Material	*Engineering Plastic(Front, Middle, Back)*

Table 2.1: AX-12A specifications [3]

Servomotors are connected to the control board with three pins (12 V, GND, and Data), and all motors are connected to those three pins. Data messages are sent using half-duplex UART at 112500 bps. Message structure is shown by Table 2.2. Communication with STM32 microcontroller was partially implemented by Ondřej Procházka in his thesis *Robotic Fire Extinguisher Mounted to an Unmanned Aerial Vehicle* [7].

Header1	***Header2***	***Servo ID***	***Length***	***Instruction***	***Param 1***	***...***	***Param N***	***Checksum***
0xFF	*0xFF*	*ID*	*Length*	*Instruction*	*Param 1*	*...*	*Param N*	*CHKSUM*

Table 2.2: Servo message definition [4]

Before the servomotors can be controlled, the correct baud rate, as well

as servo ID, must be stored into EEPROM. The proper way to do this is to use a USB2DYNAMIXEL module together with Dynamixel Wizard software. Dynamixel Wizard was not used in this project, but other hardware capable of UART communication can be used (SMT32 microcontroller in this case). The communication baud rate is set in the EEPROM control table of the servomotor and might be unknown before the first use. Desired baud rate can be selected set by sending a *change baud rate* message at all typical baud rates and by using the *Broadcast ID(254, 0xFE)* [4]. This ID targets the message at every connected servomotor, no matter its stored servo ID. Once the desired communication baud rate is chosen, the servo ID needs to be selected. That can be done by connecting a single servomotor that needs to be changed and sending a *change servo ID* message at broadcast ID (254, 0xFE).

The maximum speed of AX-12A servomotors is 59 RPM, which is more than enough for the manipulator. In fact, the speed is limited to 15 RPM. It would be beneficial to change the gear ratio of the servomotor in order to trade the speed for additional torque.

The used servomotor's resolution is 0.29 °, this value is calculated by diving the maximum servo travel (300 °) by the position value range (AX-12A uses 10-bit value to set the position), $\frac{300}{1024} \approx 0.29$ °. Assuming link length 150 mm, which theoretically means 0.78 mm resolution at the end of the link. However, the AX-12A has a gearbox, which means there is also inevitable backlash present. Backlash is a lost motion caused by the clearance between individual gears in the gearbox, manufacturing tolerances, etc. The result is that the output shaft might not hold position completely rigidly, there is some unwanted movement in the gearbox. Another inaccuracy might be caused by the elasticity and deformation of the used materials, especially with longer links which increase leverage, weight, and subsequently torque on the servomotor.

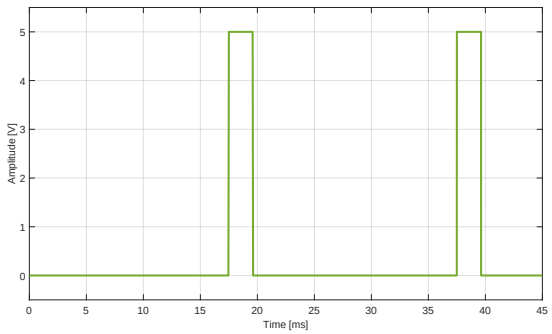
Each servomotor is capable of delivering 1.5 Nm of torque. It was enough for this application. Nonetheless, if more torque is needed in the future, then AX-18A servomotors could be used. It is a higher performance servomotor also made by Dynamixel, which is fully compatible with the used AX-12A servomotors. It has the exact dimensions and is controlled in exactly the same way. It is also only 1.3 grams heavier. But crucially, it provides 20% higher torque than AX-12A. However, AX-18A's current draw peaks at 2.2 A [8]. Unfortunately, that means that the current control circuit cannot handle three AX-18A servomotors at full load.

2.4 End-effector

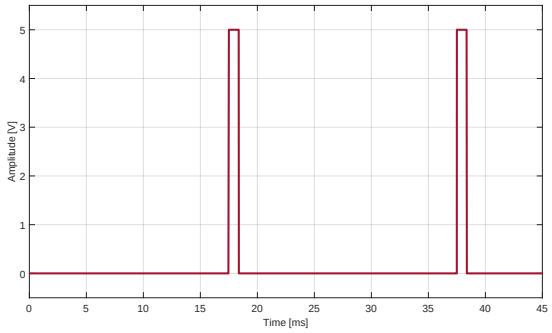
The electro-permanent magnet EPM V3 R5C (Figure 2.6) used as an end-effector is mounted on the last link using four screws. It is a type of permanent magnet made of material that is magnetized and demagnetized by applying a current pulse. The used magnet can achieve maximum holding force up to 300 N while using only 50 mW of power in a steady state. This performance with low standby power consumption is achieved by magnetizing AlNiCo material. The magnet control board uses short and powerful pulses (up to 300 A at 5 V) to magnetize or degauss the alloy, which takes roughly one second [2].

Figure 2.6: Used electro permanent magnet [2]

The magnet can be controlled by RCPWM, UAVCAN, UART or using a button. Pulse width modulation at 50 Hz frequency was chosen for simplicity. When the pulse duration is 1.75 to 2.5 ms, then the magnet is on. Pulse width 0.5 to 1.25 ms the magnet will demagnetize, as can be seen in Figure 2.7.



(a) : Pulse width 2125 μs - magnet turned on



(b) : Pulse width 870 μs - magnet turned off

Figure 2.7: PWM for controlling the magnet

As mentioned in Chapter 1, the end-effector can be swapped for a different type, for example, a mechanical gripper.

2.5 Control Board Design

An electrical circuit had to be designed to control the servomotors and other hardware. A suitable control board needs to do the following tasks:

- stepping down battery voltage to 12 V (for servomotors) and 5 V to 3.3 V (for microcontroller)
- parse incoming commands from ROS via UART

- send commands to servomotors via UART

- control the end-effector

Prototyping and testing of the circuit were done on a breadboard using Nucleo L031K6. The finished manipulator uses a custom-made PCB, which was designed in KiCad.

The circuit is based on STM32F042K6 microcontroller (Figure 2.8 - MCU) with 16 MHz oscillator. Messages from the ROS node are received via UART using pins 19 and 20. There is a UART output for servomotors on pin 8 with a pull-up resistor. The electro-permanent magnet is controlled using PWM, which is shifted up to 5 V using a level shifter (Figure 2.9).

LMZM33606RLXR step-down converter made by Texas Instrumens was needed to step-down battery voltage to 12 V for servomotors. This converter can accept up to 36 V at the input and outputs a fixed voltage (up to 20 V) based on resistor R_6. Resistor value in $k\Omega$ can be calculated using formula 2.1. Where V_{OUT} is the desired voltage value and V_{FB} is a typical value $V_{FB} = 1.006$ V [9]. LMZM33606RLXR can output 6 A maximum, which sufficient for this application. In fact, the AX-12A servomotor draws 1.5 A maximum, so the circuit is powerful enough to supply four AX-12A servomotors. That means the same circuit can be used without any modifications for an additional manipulator joint, a gripper with a servomotor, or other hardware in the future.

$$R_6 = 10 \cdot (V_{OUT} - V_{FB}) \tag{2.1}$$

Linear regulator MCP1825S-3302E (Figure 2.8 - LDO) was used to reduce 5 V from the drone to 3.3 V needed by STM32F042K6T6. Maximum current draw is 0.5 A, more than enough for the microcontroller [10].

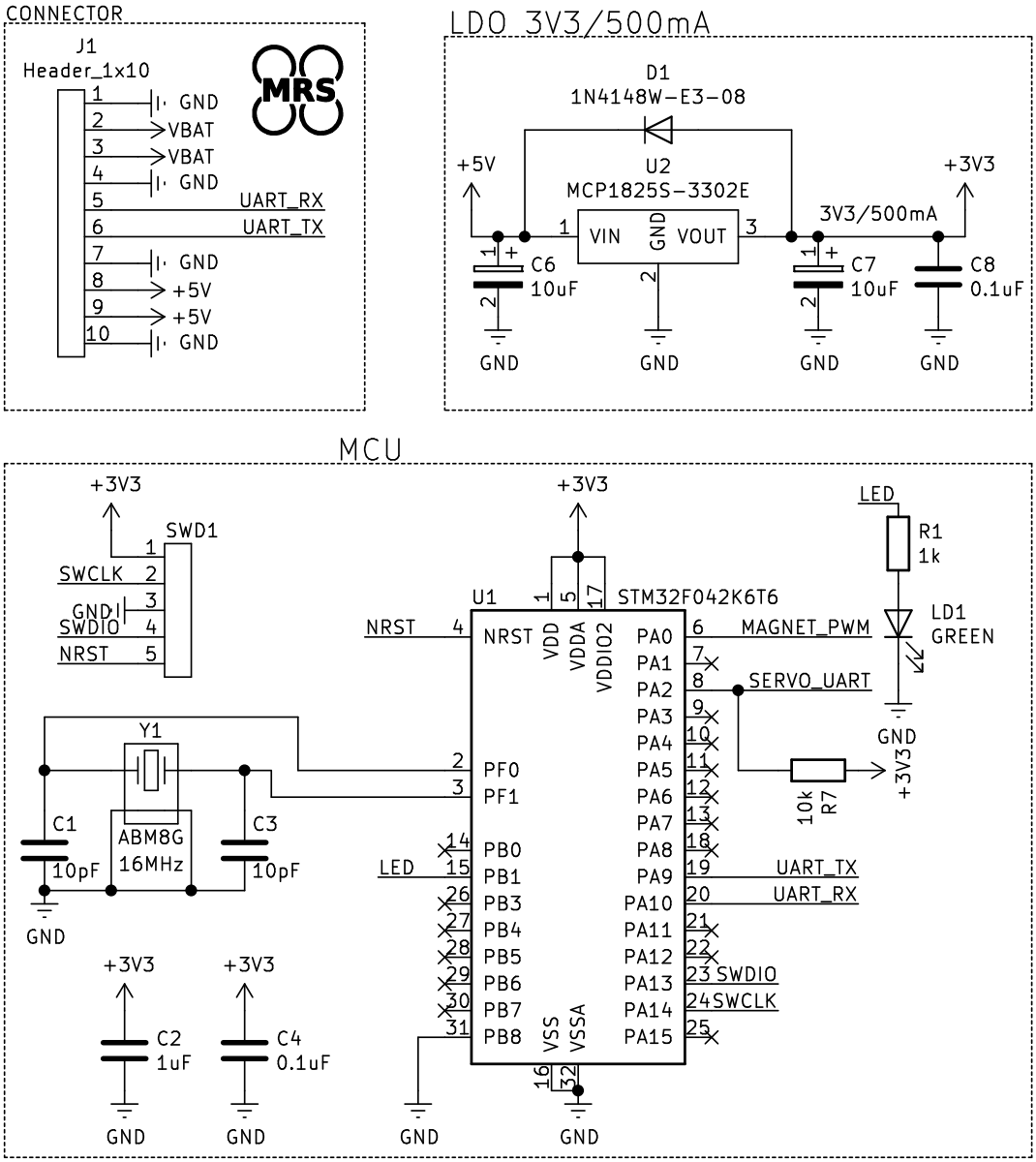


Figure 2.8: KiCad schematic - part 1

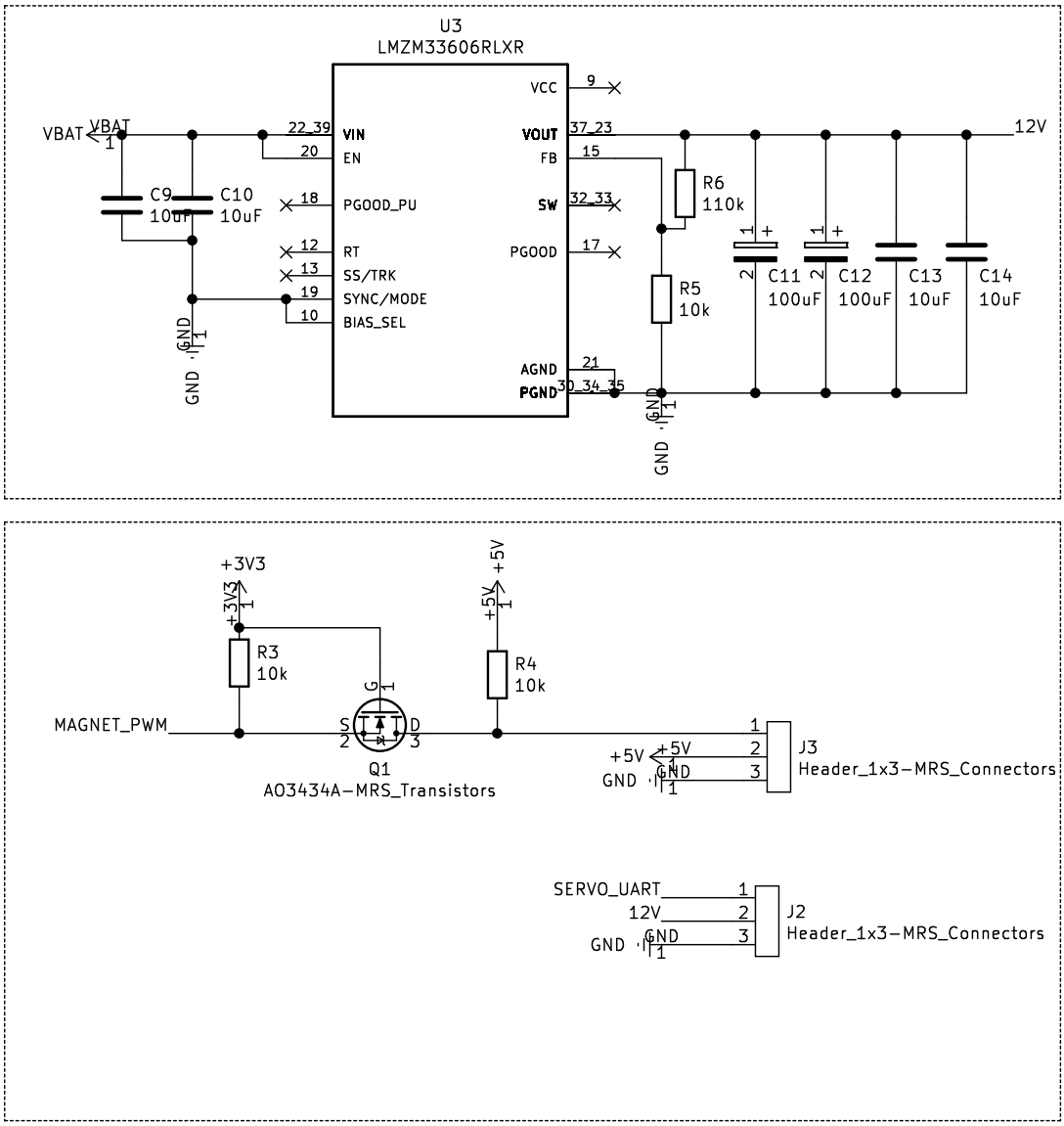


Figure 2.9: KiCad schematic - part 2

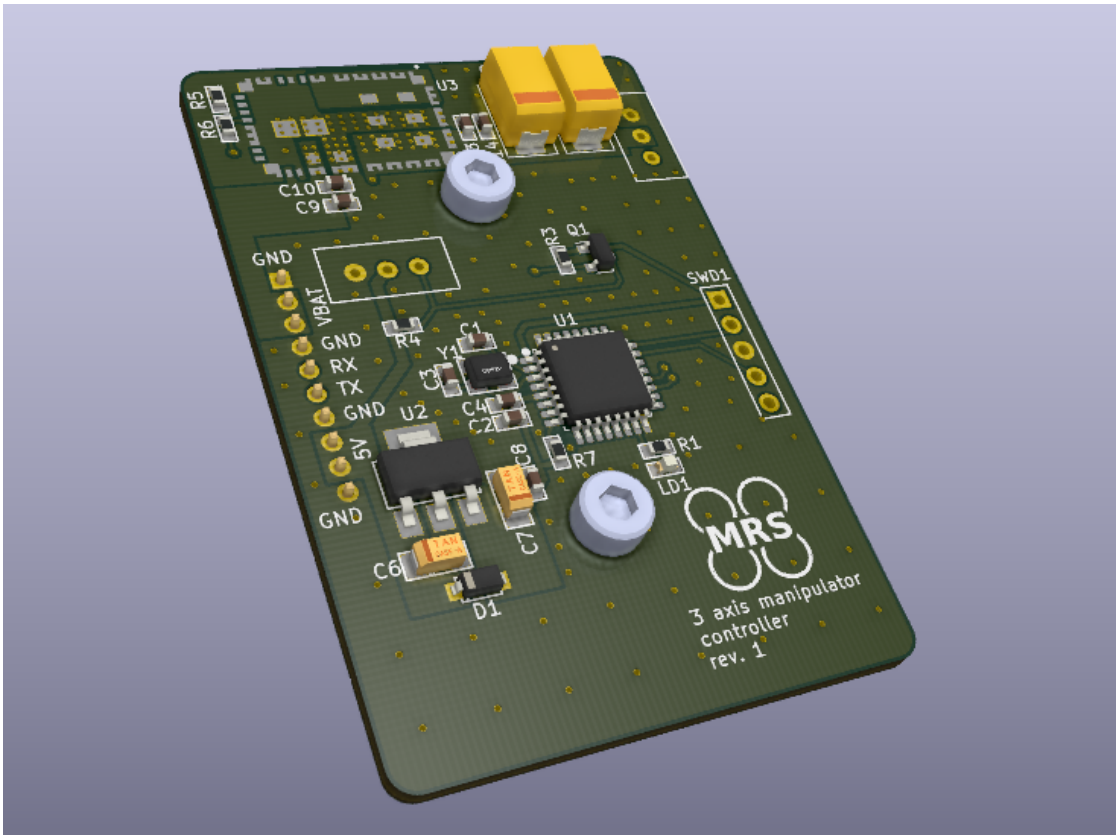


Figure 2.10: Designed PCB visualization

Chapter 3

Kinematics

In order to move the arm into the desired position, a base coordinate system needs to be established (Figure 3.1). Assuming the UAV is in a level flight, the base coordinate system has a Z-axis facing toward the ground, a Y-axis facing towards the side of the drone, and an X-axis facing forward of the UAV. The coordinate system origin lies at the intersection of the first joint rotational axis and the bracket which mounts the first servomotor. This coordinate system is particularly convenient for deriving inverse kinematics. Nevertheless, this system is only a convention, and a position can be expressed in any other coordinate system if one chooses to do so.

Inverse and forward kinematics derived in the following text is always meant with respect to this coordinate system. Distances or angles in formulas are always meant in millimeters or radians, respectively. Figure 3.2 shows the manipulator in two positions defined by joint coordinates. Figure 3.2b shows the positive direction of the joints.

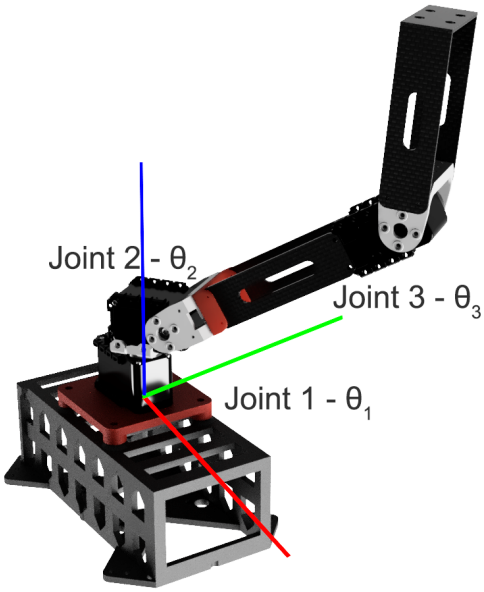
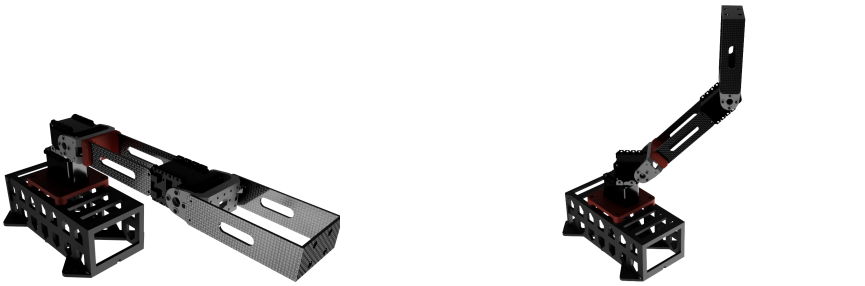


Figure 3.1: Manipulator base coordinate system



(a) : $\Theta_1 = 0\ °,\ \Theta_2 = 0\ °,\ \Theta_3 = 0\ °$ **(b) :** $\Theta_1 = 45\ °,\ \Theta_2 = 45\ °,\ \Theta_3 = 45\ °$

Figure 3.2: Manipulator in two positions

3.1 Manipulator Denavit-Hartenberg Parameters

Denavit-Hartenberg is a convention that can be used to designate reference frames for each joint. By doing so, DH parameters can serve as a quick and convenient manipulator description. Subsequently, it is trivial to derive transformation matrices from one joint to the next and obtain formulas to calculate end-effector position from joint coordinates (forward kinematics).

DH parameters can be used only for an open kinematic chain, which a three axis RRR manipulator structure has [11]. Each link is characterized by

four numbers (DH parameters) in a format shown by equation 3.1.

$$J_i = \begin{bmatrix} \alpha_i & a_i & \Theta_i & d_i \end{bmatrix} \tag{3.1}$$

- α_i is the angle between the Z-axis of the previous joint and the Z-axis of the following joint
- a_i is the length of the link (euclidian distance of joint axis)
- Θ_i is the servo angle (joint coordinate - not a constant)
- d_i is the joint Z offset

DH parameters describing the manipulator are written in equations 3.2, 3.3, and 3.4.

$$J_1 = \begin{bmatrix} \frac{\pi}{2} & 23 & \Theta_1 & 56 \end{bmatrix} \tag{3.2}$$

$$J_2 = \begin{bmatrix} 0 & 150 & \Theta_2 & 0 \end{bmatrix} \tag{3.3}$$

$$J_3 = \begin{bmatrix} 0 & 100 & \Theta_3 & 0 \end{bmatrix} \tag{3.4}$$

3.2 Direct Kinematics Task

3.2.1 Introduction

Direct (also forward) kinematics is used to calculate end-effector position and orientation from joints coordinates. For this specific application, forward kinematics can be particularly useful to determine end-effector orientation to decide which inverse kinematics solution is preferable. It is also used to calculate the position of each link to find out whether it exceeds the defined work region and causes a collision.

3.2.2 Link Frame Transformations

Using DH parameters introduced in section 3.1, it is trivial to derive forward kinematics. Each transformation from one joint to the next is of the form

shown by an equation 3.5. By simply plugging DH parameters into this matrix, transformation matrices from each joint to the next joint are obtained. These transformation matrices (link frame transformation) are used for collision detection, and by multiplying them together, the forward kinematics is achieved.

$$T_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

Matrix 3.6 is the transformation matrix from base coordinate system to the second joint frame.

$$T_{1\to2} = \begin{pmatrix} \cos\Theta_1 & 0 & \sin\Theta_1 & 23\cos\Theta_1 \\ \sin\Theta_1 & 0 & -\cos\Theta_1 & 23\sin\Theta_1 \\ 0 & 1 & 0 & 56 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

Matrix 3.7 is the transformation matrix from the second joint frame to the third joint frame.

$$T_{2\to3} = \begin{pmatrix} \cos\Theta_2 & -\sin\Theta_2 & 0 & 150\cos\Theta_2 \\ \sin\Theta_2 & \cos\Theta_2 & 0 & 150\sin\Theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Matrix 3.8 is the transformation matrix from the second joint frame to the end-effector frame.

$$T_{3\to E} = \begin{pmatrix} \cos\Theta_3 & -\sin\Theta_3 & 0 & 100\cos\Theta_3 \\ \sin\Theta_3 & \cos\Theta_3 & 0 & 100\sin\Theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

3.2.3 End-effector Position

By multiplying matrices in section 3.2.2, a transformation matrix from the base coordinate system to the end effector is calculated. By applying such transformation to the null vector in homogeneous coordinates (equation 3.9),

an end-effector position (desired forward kinematics) is acquired (equation 3.10).

$$T_{1\to2} \cdot T_{2\to3} \cdot T_{3\to E} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 23\cos\Theta_1 + 150\cos\Theta_1\cos\Theta_2 + 100\cos\Theta_3\cos\Theta_1\cos\Theta_2 - 100\sin\Theta_3\cos\Theta_1\sin\Theta_2 \\ 23\sin\Theta_1 + 150\cos\Theta_2\sin\Theta_1 + 100\cos\Theta_3\cos\Theta_2\sin\Theta_1 - 100\sin\Theta_3\sin\Theta_1\sin\Theta_2 \\ 150\sin\Theta_2 + 100\cos\Theta_2\sin\Theta_3 + 100\cos\Theta_3\sin\Theta_2 + 56 \end{bmatrix} \tag{3.10}$$

3.2.4 Using Link Frame Transformations for Collision Detection

Collision with the drone must be avoided. Avoiding collision between the end-effector and the drone is trivial. Simply check whether the end-effector lies within the defined work region. However, that is not enough. The rest of the manipulator also must be considered so that no collision can occur. Manipulator structural parts can be checked for collision by checking the positions of each revolute joint.

Joint 1 is rigidly connected to the mounting bracket. Therefore, it cannot cause a collision because it is stationary with respect to the drone.

Joint 2 might interfere with the drone, but if that situation would occur then joint 3 would collide with the drone first.

Joint 3 is the only thing that needs to be checked for collision (apart from the end-effector) because it might exceed the allowed work region even if the end-effector lies within the work region. Therefore, a collision check for joint 3 must be performed. By using transformations matrices in section 3.2.2, it is possible to calculate the position of the third servomotor to check for collisions. Transformation matrix from the base coordinate system to the third joint frame is shown by equations 3.11 and 3.12.

$$T_{1\to3} = T_{1\to2} \cdot T_{2\to3} \tag{3.11}$$

$$T_{1\to3} = \begin{bmatrix} \cos\Theta_1\cos\Theta_2 & -\cos\Theta_1\sin\Theta_2 & \sin\Theta_1 & 23\cos\Theta_1 + 150\cos\Theta_1\cos\Theta_2 \\ \cos\Theta_2\sin\Theta_1 & -\sin\Theta_1\sin\Theta_2 & -\cos\Theta_1 & 23\sin\Theta_1 + 150\cos\Theta_2\sin\Theta_1 \\ \sin\Theta_2 & \cos\Theta_2 & 0 & 150\sin\Theta_2 + 56 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Deriving the desired position of the third joint axis center is analogous to deriving the end-effector position. Applying the null vector in homogeneous coordinates to the transformation calculates the position of the third joint (equation 3.13). The third joint axis center position, based on joint coordinates 2 and 3, is defined in equation 3.14.

$$T_{1\to3} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \\ 1 \end{bmatrix} \tag{3.13}$$

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} 23\cos\Theta_1 + 150\cos\Theta_1\cos\Theta_2 \\ 23\sin\Theta_1 + 150\cos\Theta_2\sin\Theta_1 \\ 150\sin\Theta_2 + 56 \end{bmatrix} \tag{3.14}$$

3.3 Inverse Kinematics Task

3.3.1 Introduction

Inverse kinematics is a mathematical process of calculating joints coordinates given an end effector position. It is needed to move the manipulator into desired cartesian position. IKT of this specific manipulator has four solutions. Therefore, end effector orientation is taken into consideration when calculating inverse kinematics. A solution that points the magnet more towards the ground is preferred because the manipulator is intended for object manipulation. Figure 3.3 shows all possible solutions for end-effector position $(2, 0, 300)$.

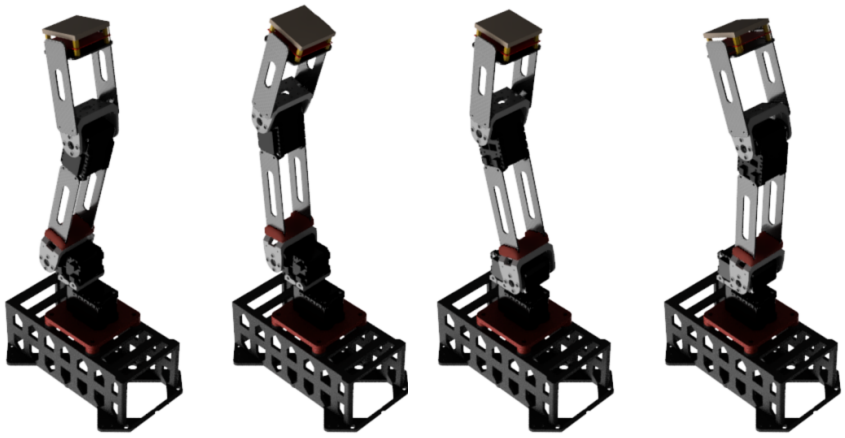


Figure 3.3: Possible solutions of inverse kinematics problem

For clarity sake assume a situation illustrated by the Figure 3.4.

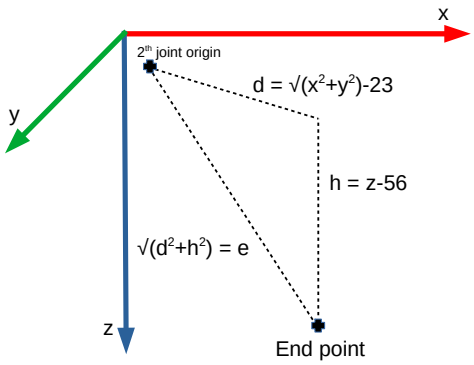


Figure 3.4: Assumed situation

Keep in mind that the first joint angle has two solutions, as is mentioned in section 3.3.2. Figure 3.4 assumes only one solution to keep things more clear. The second solution of the first joint has a different origin of the second joint. For this reason, different values of d and e need to be defined for both solutions of joint 1.

$$h = z - 56 \tag{3.15}$$

$$d_1 = \sqrt{x^2 + y^2} - 23 \tag{3.16}$$

$$e1 = \sqrt{d_1^2 + h^2} \tag{3.17}$$

$$d_2 = \sqrt{x^2 + y^2} + 23 \tag{3.18}$$

$$e1 = \sqrt{d_2^2 + h^2} \tag{3.19}$$

- (3.15) - h is the distance in z direction from the second joint axis center to the end effector. h is a constant defined by chosen base coordinate system and physical manipulator structure
- (3.16) - d_1 is the distance in the xy plane from the second joint axis center to the end effector (valid only for the first solution of joint 1)
- (3.17) - e_1 is the total distance from the second joint axis center to the end effector (valid only for the first solution of joint 1)
- (3.18) - d_2 is the distance in the xy plane from the second joint axis center to the end effector (valid only for the second solution of joint 1)
- (3.19) - e_2 is the total distance from the second joint axis center to the end effector (valid only for the second solution of joint 1)

3.3.2 First Joint

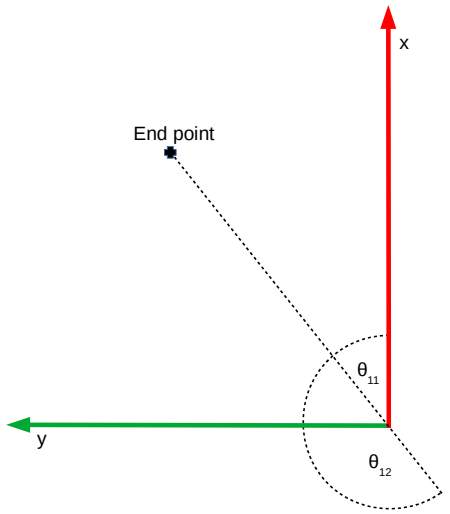


Figure 3.5: Top view (from the drone)

Angle of the first servomotor depends only on x and y coordinate. Equations 3.20 and 3.21 are the two possible solutions of the first joint. Figure 3.5 is a visualization of these two solutions.

$$\mathbf{\Theta_{11}} = \text{atan2}(y, x) \tag{3.20}$$

$$\mathbf{\Theta_{12}} = \text{atan2}(-y, -x) \tag{3.21}$$

3.3.3 Second and Third Joint

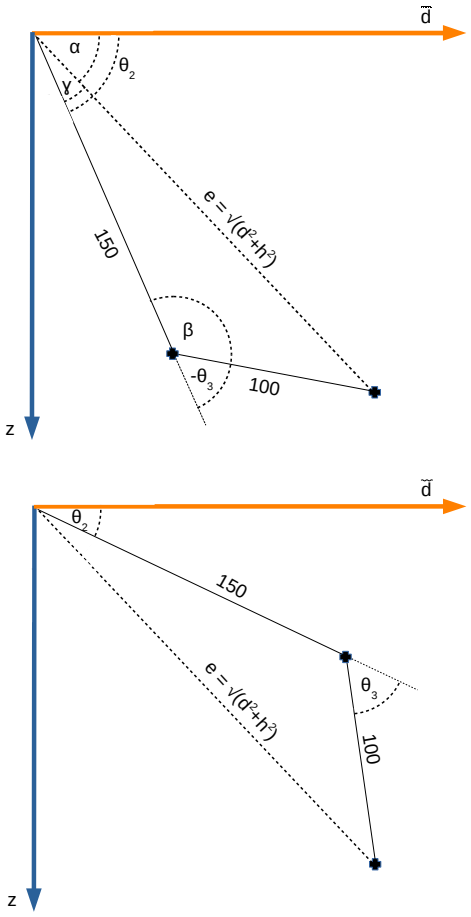


Figure 3.6: Two possible solutions of joints 2 and 3

For each solution of joint 1 exist two solutions of joints 2 and 3. That means four solutions of the inverse kinematics problem in total.

Firstly, angles α, β, and γ (equations 3.22) need to be solved using the law of cosines. Those angles are the same for both solutions of joints 2 and 3.

Once those angles are calculated, solutions for joints 2 and 3 are effortlessly expressed in equations 3.23.

$$\begin{aligned} \alpha &= \text{atan2}(h, d) \\ \beta &= \cos^{-1}(-\frac{e^2 - 150^2 - 100^2}{2 \cdot 150 \cdot 100}) \\ \gamma &= \cos^{-1}(-\frac{100^2 - 150^2 - e^2}{2 \cdot e \cdot 150}) \end{aligned} \tag{3.22}$$

$$\begin{aligned} \boldsymbol{\Theta_{21}} &= \alpha + \gamma \\ \boldsymbol{\Theta_{22}} &= \alpha - \gamma \\ \boldsymbol{\Theta_{31}} &= \beta - \pi \\ \boldsymbol{\Theta_{32}} &= \pi - \beta \end{aligned} \tag{3.23}$$

3.3.4 Inverse Kinematics Solutions

As was already mentioned, four solutions of inverse kinematics exist.

Let $P = (x, y, z)$ be an IKT input (desired end effector position).

Define variables for intermediate calculations (equations 3.24), that helps to express final solutions very neatly.

$$\begin{aligned} h &= z - 56 \\ d_1 &= \sqrt{x^2 + y^2} - 23 \\ d_2 &= \sqrt{x^2 + y^2} + 23 \\ e_1 &= \sqrt{d^1 + h^2} \\ e_2 &= \sqrt{d_2^2 + h^2} \\ \alpha_1 &= \text{atan2}(h, d_1) \\ \alpha_2 &= \text{atan2}(h, -d_2) \\ \beta_1 &= \cos^{-1}(-\frac{e_1^2 - 150^2 - 100^2}{2 \cdot 150 \cdot 100}) \\ \beta_2 &= \cos^{-1}(-\frac{e_2^2 - 150^2 - 100^2}{2 \cdot 150 \cdot 100}) \\ \gamma_1 &= \cos^{-1}(-\frac{100^2 - 150^2 - e_1^2}{2 \cdot e_1 \cdot 150}) \\ \gamma_2 &= \cos^{-1}(-\frac{100^2 - 150^2 - e_2^2}{2 \cdot e_2 \cdot 150}) \end{aligned} \tag{3.24}$$

Then the four possible solutions (if they exist) can be elegantly expressed in a form $S_i = [\Theta_1, \Theta_2, \Theta_3]$. The solutions are formulated in equations 3.25.

$$\begin{aligned} \mathbf{S_1} &= [\text{atan2}(y, x), \alpha_1 + \gamma_1, \beta_1 - \pi] \\ \mathbf{S_2} &= [\text{atan2}(y, x), \alpha_1 - \gamma_1, \pi - \beta_1] \\ \mathbf{S_3} &= [\text{atan2}(-y, -x), \alpha_2 + \gamma_2, \beta_2 - \pi] \\ \mathbf{S_4} &= [\text{atan2}(-y, -x), \alpha_2 - \gamma_2, \pi - \beta_2] \end{aligned} \tag{3.25}$$

For $x = y = 0$ an infinite number of solutions exist. In this case position of joint 1 can be arbitrarily chosen. This is handled by the function atan2(), which either returns 0 or π and solves the singularity issue.

3.3.5 Choosing the Optimal Solution

Inverse kinematics is calculated in the ROS node responsible for controlling the manipulator. The most optimal solution is picked and sent to the control board. Due to the limited servomotor travel, it is very sporadic that all four solutions are valid.

The first thing that is checked is whether the solution is within the travel limit of the servomotors. Then a collision check is performed. Also, when deciding which solution to choose end-effector angle is a factor. The purpose of the robotic arm is manipulation with objects on the ground. Naturally, it is desirable to keep the end-effector pointed towards the ground. Therefore, for each IKT solution, an angle with the ground plane is calculated using a formula 3.26.

$$\phi = \frac{\pi}{2} - \Theta_2 - \Theta_3 \tag{3.26}$$

If $\phi = 0$, then the magnet is pointed precisely towards the ground (assuming the drone is in a level flight above level ground). A minimal value of $|\phi|$ is desirable in order to have the end-effector in an optimal position.

After all these things are considered, a single inverse kinematics solution is usually left (assuming that desired point is in the manipulator reach and in servomotor travel limits). However, there can exist two solutions in some positions that do not cause collisions and have the same angle with the ground plane. When such a situation occurs, the ROS node remembers the last position and chooses the solution that minimizes servo travel from the last position.

Chapter 4

Software

The UAV's system is running on Robot Operating System. It was necessary to implement software in ROS to allow controlling the manipulator and make it compatible with the rest of the MRS system. Position request is made in ROS environment, necessary calculations are performed, and the control system creates messages and sends them to the created circuit with STM32 microcontroller. The control board interprets incoming messages, handles servomotor movement and end-effector state.

Diagram 4.1 shows manipulator control system. Position request starts in ROS environment by publishing on the relevant topic. The request is read, IKT is calculated, collision check is performed, and byte array for serial communication is created. This array is published to another node *mrs_serial*, which is a ROS package created by the MRS group, responsible for handling serial communication. The message is interpreted by STM32, and commands to servomotors and end-effector are sent.

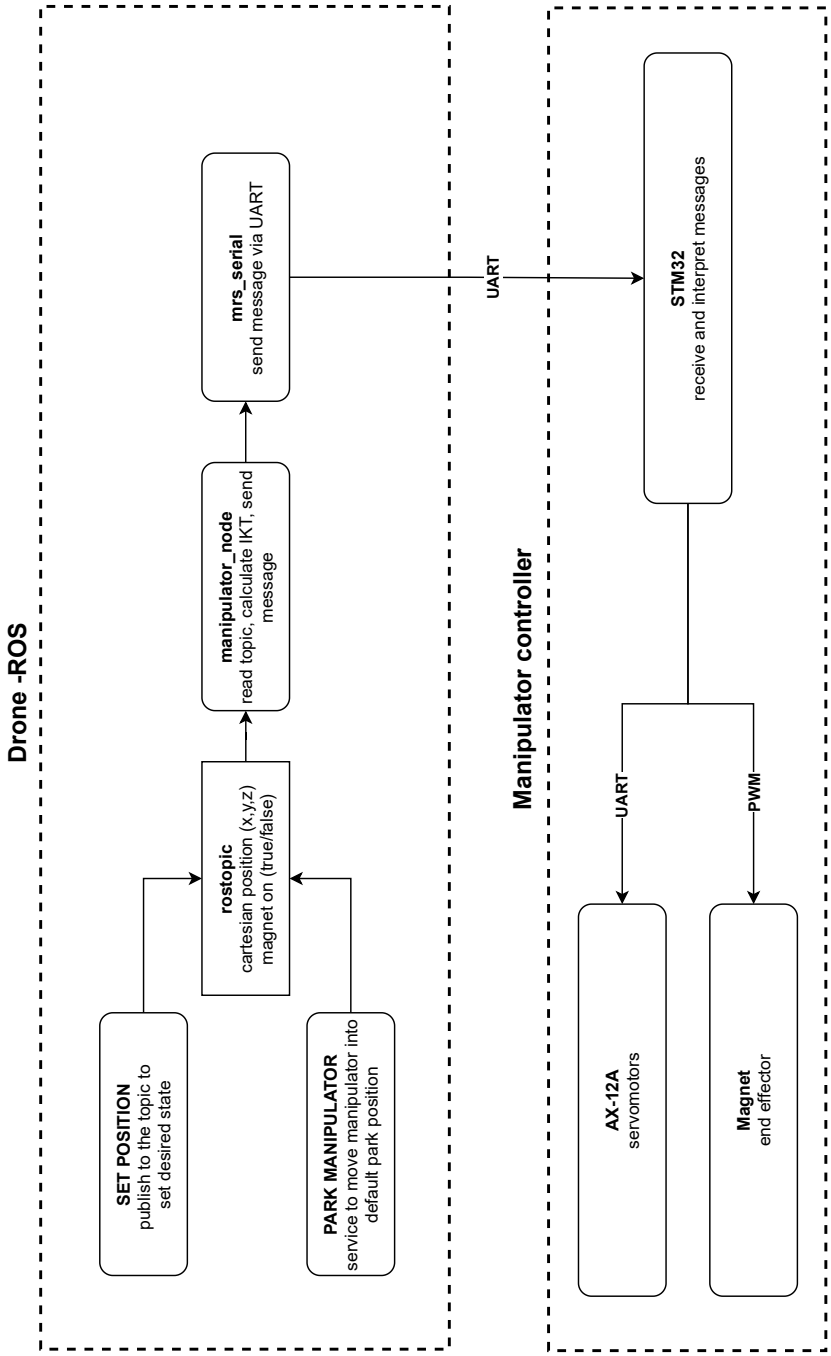


Figure 4.1: Manipulator control system

4.1 ROS Software

4.1.1 Indroduction

The control system of the drone is based on ROS architecture. It is a collection of tools that are organized into so-called ROS packages. It is also distributed under an open-source license which makes it ideal for the development of robot applications [12]. This section will explain the core components of the Robot Operating System, which are then used to control the manipulator.

- ROS nodes

  ROS node is an executable within a ROS package that can be used for various computation tasks. Multiple nodes can be combined together and share data to build complex systems. Each node can publish messages on a ROS topic. Other nodes then can subscribe to this topic and read messages. Such an approach provides many advantages. Each node can be dedicated to simpler tasks that reduce implementation complexity. Also, potential errors will be isolated to the specific node and will not propagate to the rest of the system [13].

  Nodes can also provide or call services, which are explained in the following text.

- ROS topics and messages

  Communication between nodes is done using ROS topics. The structure of the message must be defined. This definition includes the name and data type of the shared variables. Standard messages can be used, or custom messages can be defined in *msg* file.

  Nodes can send (or publish) messages on a topic, and another node can subscribe to the topic to receive the message. Multiple nodes can be connected to the same topic [12].

- ROS services

  Nodes can provide or call services. Services are a different way to exchange data in ROS. By using a service, it is possible to call a function within a node from outside. It is generally not recommended for long-lasting tasks because the service caller expects a reply before the service can be used again [12]. That makes services ideal for irregular and quick tasks. For example, move an actuator, switching lights, take a picture with a camera, etc.

In this application, a service is used to move the manipulator into a parking position. Standard services can be used, but using a service usually necessitates defining the request and reply in a *srv* file.

4.1.2 Manipulator ROS Package

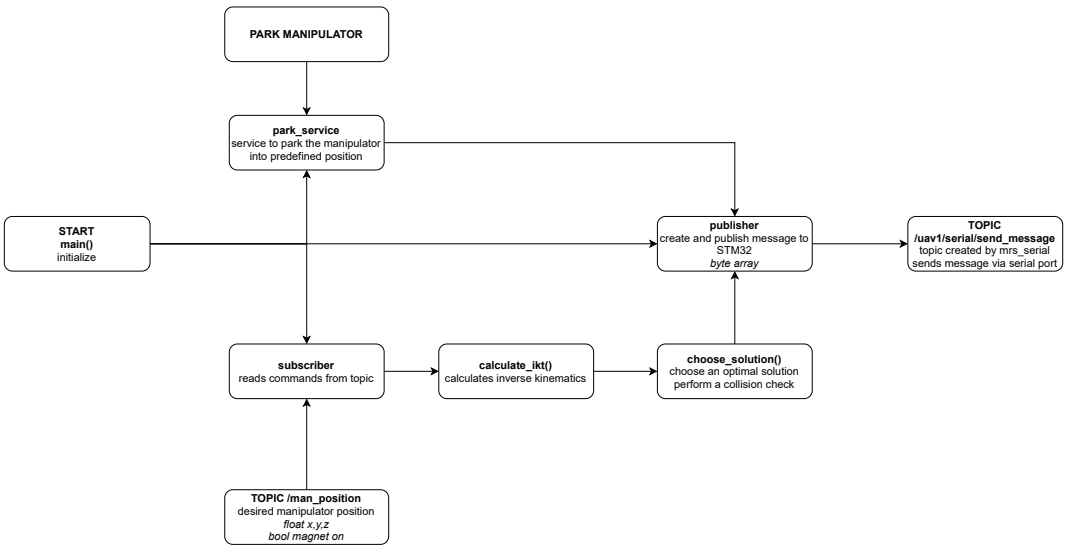


Figure 4.2: Flow chart - manipulator control node

Diagram (Figure 4.2) shows how the manipulator control is done in ROS. The node initializes a subscriber, publisher, and service for moving the manipulator into a predefined parking position.

Subscriber creates a topic */man_position*. To this topic, the desired position is published by external nodes. When a request for new position is made the function *calculate_ikt()* is called. Inverse kinematics is calculated, which returns up to four solutions. Then function *choose_solution()* is called, where the optimal solution is chosen as mention in section 3.3.5 and a collision check is performed. If a position request is not valid because the manipulator cannot reach such position or it causes a collision, then the last position is restored on topic */man_position*. If a valid solution exists, then function *choose_solution()* returns it.

After that, a byte array is created and published to a topic */uav1/serial/send_message*. This is a topic created by node *mrs_serial* (ROS package created by MRS group), which handles the communication via serial port.

For testing purposes, a node that publishes to topic */man_position* based on keyboard input was programmed.

4.2 STM32

4.2.1 Indroduction

The designed PCB utilizes an STM32F042K6T6 microcontroller to receive and interpret UART messages, controlling servomotors and the electro-permanent magnet. The code was written in STM32CubeIDE using the HAL library.

STM32CubeIDE is a development tool that provides various features. It provides various features such as code editor, debugger, or STM32CubeMX tool. STM32CubeMX provides a graphical interface for easy configuration of peripherals, clocks, timers, interrupts, serial communication, etc. It was used to set up clocks of the microcontroller and automatically generate initialization code for necessary features (UART, timer, DMA, etc.)

The HAL driver library provides a user-oriented API to simplify implementation. It provides an upper-level interface to initialize and control peripherals, handle interrupts, timers, pulse-width modulation, DMA, etc. The advantage of using this library is the relative simplicity of the implementation. However, for optimal performance and advanced features, LL (low-level) drivers should be used instead. [14]

4.2.2 STM32 Firmware

Firstly an initialization of all necessary features is done. UART1, UART2 (for ROS messages and servomotors) instances are created and initialized. Correct baud rate, mode, word length, parity, etc., is set. Peripherals are correctly set as well as timers for PWM, which controls the magnet. Servomotors are initialized, the maximum rotational speed is set. Also, the manipulator is moved into a predefined parking position when the PCB circuit is powered for the first time.

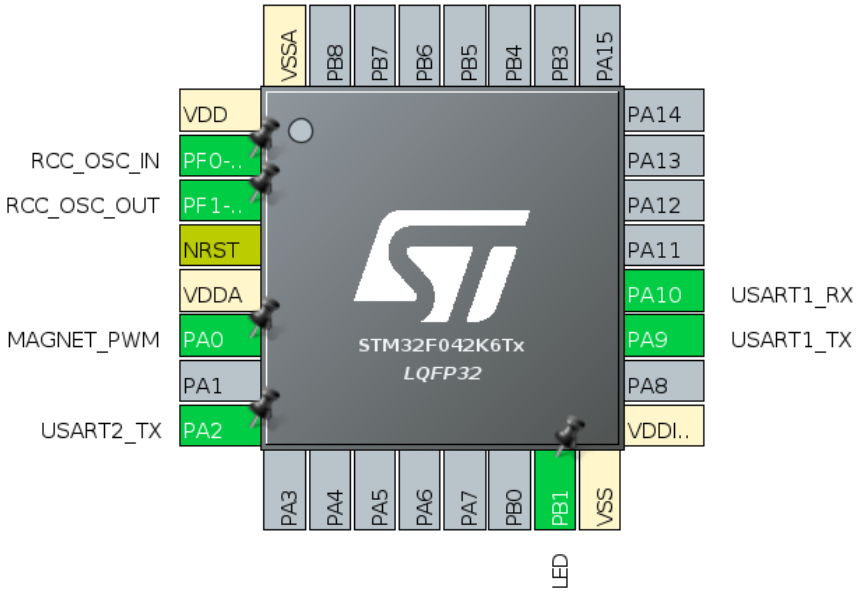


Figure 4.3: Microcontroller pinout

Direct memory access (DMA) was established to receive messages from ROS in a non-blocking manner. Direct memory access is a way of data transfer without processor intervention. This approach gives performance benefits [15]. Timer was also used for PWM generation with direct registry control. The PWM pulse width is controlled by direct manipulation of a registry value. Figure 4.3 shows the pinout in Device Configuration Tool.

Then the main while loop is entered. Received messages are directly written into memory without processor intervention and are processed. Then the necessary commands are sent to the servomotors. Only valid commands should be received from ROS. However, the STM32 firmware also checks the servomotor travel limits, which is redundancy to increase the safety of the manipulator and protect the UAV from collisions. The magnet end-effector is controlled by directly changing the registry value of the PWM timer. Diagram 4.4 shows the microcontroller program function.

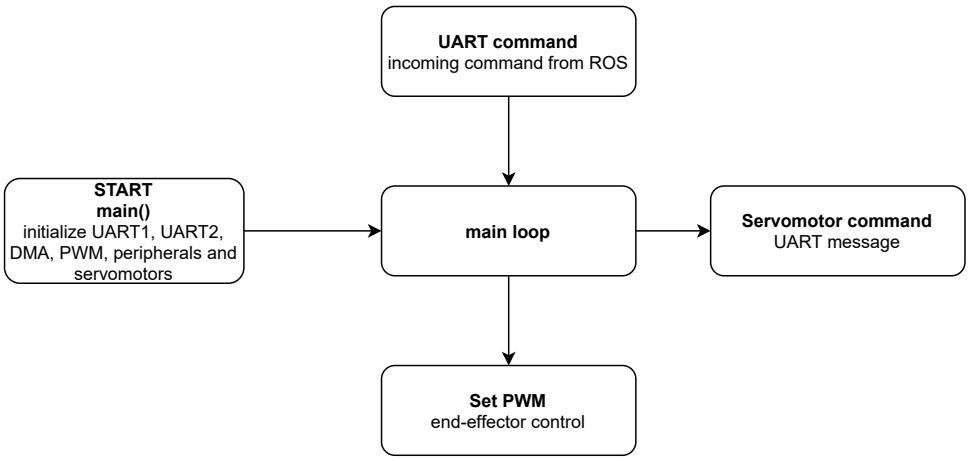


Figure 4.4: Microcontroller program diagram

Chapter 5

Conclusion

The goal of this thesis was to design and develop a robotic manipulator. The task was motivated by MBZIRC 2023 robotic competition, where drones will compete in capturing stationary or moving objects. The task assignment had multiple objectives:

1. **Design a robotic arm that will allow safe manipulation with objects below a UAV**

   A robotic arm with three revolute joints was the chosen architecture. Dynamixel AX-12A servomotors were utilized as revolute joints. The mechanical structure was designed in Fusion360 CAD software. A prototype was built using the hardware included with the AX-12A servomotors, structural links from MDF, and 3D printed parts from PLA. The prototype was put into test, and only minor changes were made for the second version. The final version had structural links machined from 2 mm thick carbon fiber in order to decrease weight while maintaining rigidity, and mounting brackets were printed from PETG plastic which provides better properties than PLA.

2. **Design a suitable end-effector for the manipulator**

   An electro-permanent magnet was chosen as an end-effector. Used OpenGrab EPM v3 electro-permanent magnet made by Nicadrone is capable of up to 300 N of holding force while keeping very low power consumption in a steady state. The magnet is mounted to the manipulator using four screws and can be swapped for a different end-effector type if necessary. The designed hardware can be used without any modifications to control a servo gripper, for example. Furthermore, implemented inverse

kinematics can be easily modified by changing parameters to move the end-effector reference point.

3. **Implement a control system for the manipulator. Utilize the provided STM32 microcontroller**

   Manipulator is controlled using ROS. A control node was implemented, which reads desired position on a ROS topic, calculates inverse kinematics, selects optimal solution. A collision check is performed, then a byte array is created and published for another node that handles the communication with the PCB.

   A circuit with a custom-made PCB was designed, which receives commands via UART, interprets them, and controls servomotors using half-duplex UART and magnet end-effector using pulse-width modulation.

4. **Devise and implement an Inverse Kinematic Task for the manipulator**

   Direct and inverse kinematics task problems were introduced and solved. Up to four solutions of inverse kinematics can exist. Because the manipulator is intended to manipulate objects on the ground, the optimal solution is selected based on the end-effector attitude with the ground plane. Simple collision avoidance was implemented by defining a bounding box around the manipulator and restricting the movement of the end-effector and the third joint inside this bounding box.

5. **Implement communication between a computer and the microcontroller in order to control the manipulator and the end-effector from the Robot Operating System (ROS)**

   ROS tools created by the MRS group were used for the implementation of communication through serial port. It was necessary to define a custom message with all the needed information. The message includes a boolean value for the end-effector state, the angles of all three joints, and other data blocks such as a checksum, message ID, or message length. Manipulator control ROS node creates a byte array a publishes it to the correct topic. ROS node *mrs_serial* then sends it using a serial port.

   STM32 microcontroller then receives the data and saves them using Direct Memory Access, a way of data transfer without processor intervention and thereby increasing performance. Data are then continuously read from memory in a loop, and UART messages for servomotors are sent as well as the PWM signal for the magnet end-effector.

In future work, it might be beneficial to use a servomotor with a different gear reduction. Currently, the servomotor speed is limited to around 25% of the maximal achievable speed. A different gear ratio would sacrifice unnecessary speed capabilities and provide additional torque. Used Dynamixel AX-12A servomotors could be swapped with a fully compatible AX-18A version, which provides up to 20% higher peak torque while being only 1.3 grams heavier.

Appendix A

Bibliography

[1] Multi-robot Systems MAV platforms. Tarot T650 Sport. http://mrs.felk.cvut.cz/research/micro-aerial-vehicles. Accessed: 2021-04-07.

[2] Pavel Kirienko. OpenGrab EPM v3 documentation. https://kb.zubax.com/display/MAINKB/OpenGrab+EPM+v3. Accessed: 2021-03-20.

[3] ROBOTIS. DYNAMIXEL AX-12A. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/. Accessed: 2021-04-05.

[4] ROBOTIS. DYNAMIXEL Protocol. https://emanual.robotis.com/docs/en/dxl/protocol1/. Accessed: 2021-04-07.

[5] MBZIRC 2023. https://www.mbzirc.com. Accessed: 2021-04-17.

[6] MBZIRC. THE CHALLENGE 2023. https://www.mbzirc.com/challenge/2023. Accessed: 2021-04-17.

[7] Ondřej Procházka. Robotic Fire Extinguisher Mounted to an Unmanned Aerial Vehicle, 2020. https://dspace.cvut.cz/handle/10467/87699. Accessed: 2021-04-09.

[8] ROBOTIS. DYNAMIXEL AX-18A. https://emanual.robotis.com/docs/en/dxl/ax/ax-18a/. Accessed: 2021-04-19.

[9] Texas Instruments. LMZM33606 datasheet. https://www.ti.com/lit/ds/symlink/lmzm33606.pdf. Accessed: 2021-04-10.

[10] MICROCHIP. MCP1825S-3302E datasheet. http://ww1.microchip.com/downloads/en/devicedoc/22056b.pdf. Accessed: 2021-04-10.

[11] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 1st edition, 2017.

[12] ROS wiki. `http://wiki.ros.org`. Accessed: 2021-05-02.

[13] Anis Koubaa. *Robot Operating System (ROS): The Complete Reference*, volume 3. Springer, 2018.

[14] Description of STM32F4 HAL and low-layer drivers. `https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf`. Accessed: 2021-05-07.

[15] Carmine Noviello. *Mastering the STM32 Microcontroller*. Lean Publishing, 2016.

Appendix B

List of Abbreviations

Abbreviation	Meaning
UAV	Unmanned Aerial Vehicle
MBZIRC	The Mohamed Bin Zayed International Robotics Challenge
DOF	Degrees of Freedom
RRR	3 revolute joints - manipulator structure
DKT	Direct Kinematics Task
IKT	Inverse Kinematics Task
RPM	Revolutions per Minute
DH parameters	Denavit-Hartenberg parameters
CAD	Computer-aided Design
CNC	Computer Numerical Control
MDF	Medium Density Fiberboard
PLA	Polylactic acid - material for 3D printing
PETG	Polyethylene terephthalate glycol - material for 3D printing
ROS	Robot Operating System
API	Application Programming Interface
DMA	Direct Memory Access
HAL	Hardware Abstraction Layer
UART	Universal Asynchronous Receiver-Transmitter
PWM	Pulse Width Modulation
PCB	Printed Circuit Board
LDO	Low Dropout Regulator
EEPROM	Electrically Erasable Programmable Read-Only Memory

Appendix C

Supplementary Material

File *robotic_manipulator_for_an_unmanned_aerial_vehicle_material.zip* is attached to the thesis. The content of this file is explained below.

Directory name	Content
manipulator_parts	STL files for 3D printing and DXF files of carbon fiber links
src/ros_package	Manipulator ROS package
src/stm_firmware	Microcontroller firmware