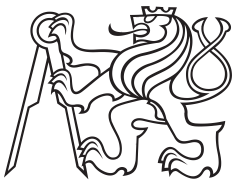
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Robotic Fire Extinguisher Mounted to an Unmanned Ground Vehicle

David Puchoň

Supervisor: Ing. David Žaitlík
Field of study: Cybernetics and Robotics
January 2022

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Puchoň David** Personal ID number: **474552**

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 Ground Vehicle

Bachelor's thesis title in Czech:

Robotický hasící systém pro bezpilotní pozemní robot

Guidelines:

This thesis will focus on the design and development of a fire extinguishing system for an Unmanned Ground Vehicle (UGV). 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] 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: **10.01.2020** Deadline for bachelor thesis submission: **04.01.2022**

Assignment valid until: **13.02.2022**

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 my supervisor that gave me a lot of advices and patience. My family supported me during the whole studies so the big acknowledgment belongs to them.

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.

Prague, 4. January 2022

Abstract

This thesis is about designing a fire extinguisher system for autonomous ground vehicle. System contains a built manipulator with two degrees of freedom that directs the water stream to the target. Firstly, hardware was designed, simulations were made and after that, software controlling the system was written. The functionality was tested during the static experiment.

Keywords: extinguisher system, UGV, manipulator

Supervisor: Ing. David Žaitlík

Abstrakt

Tato práce se zabývá návrhem hasicího systému upevněném na autonomním pozemním vozidlem. Systém mírí pomocí sestrojeného manipulátoru se dvěma stupni volnosti na cíl a následně spouští proud vody. Nejdřívě byl navržen hardware, provedena simulace na základě matematických výpočtů a následně i sepsán software pro řízení celého systému. Funkčnost samotného hasicího systému byla ověřena staticky.

Klíčová slova: hasicí systém, UGV, manipulátor

Překlad názvu: Robotický hasící systém pro bezpilotní pozemní robot

Contents

Figures

Tables

Chapter 1

Introduction

Unmanned ground vehicles (UGV) are trend of the last few years. Next to unmanned aerial vehicles, they create a group of vehicles that can solve problems autonomously. That is the reason why research in this field is running at FEE. A lot of competitions with unmanned vehicle are held. For example Defence Advanced Research Projects Agency (DARPA) Subterrean Challege that involves mapping, navigating and searching in the underground. This competition was attended by Center for Robotics and Autonomous Systems (CRAS) from FEE CTU.

The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) competition was attended twice by MRS group in 2017 and 2020. The team led by Dr. Martin Saska attended the tank to extinguish fire on the building. This task also inspired this thesis.

1.1 Ideas

The main idea to make this project was inspired by the competition. This task is mainly focused on drone usage bud ground vehicles can carry much more water than drones, so few ground vehicles could help fight the fire. Thus, the idea of miniature autonomous firefighter car is on point. However it seems that UGV design is not that common because majority of academic article is focused on fire extinguishing UAVs [6] [7] [9]. Ground vehicles used for fire extinguishing are common, but those are not unmanned. Some works

were for example focused on ground vehicle with 4DOF manipulator [5].

1.2 Outline

This thesis is divided into five chapters First chapter 2 is about hardware used in this project. Second chapter 3 introduces developed manipulator. Third chapter 4 describes used software. Chapter 5 is devoted to the real life experiments. Last chapter 6 sums up the whole thesis.

Chapter 2

Hardware

This chapter is discusses hardware used to construct the whole system which consists of servos, pump, control board, 3D models etc.

2.1 Pump

The system is going to be placed on a quite large ground vehicle with a relatively big water tank, so a big submersible pump will be used to transport water from the tank to the nozzle. A submersible pump is suitable for this usage because it can be placed into the tank and pump water to the hose. This eliminates the need for the second hose from the tank to the pump which would be necessary for the surface pump. Another option could be pushing water with air compressed into the tank. This would require an airtight tank which is not suitable for this usage. Thus, a pump will be used.

2.1.1 Used pump

The used pump is a no-name type from one well-known Chinese online store. It is powered by 12V and its flow is $1200\,\mathrm{l\,h^{-1}}$. Unfortunately, there is not much information about this pump on the internet, so the flow value has been verified and it corresponds with the declared value. The pump is shown in figure 2.1.

Figure 2.1: Used pump

In addition to the pump, garden hose with adaptor to the standard 1/2" thread was used to connect the pump with the nozzle.

2.2 Servomotors

There are two major types of motors used in manipulators - servomotors and stepper motors. Stepper motors are usually cheaper because they are not so complex. No feedback makes them much harder to control. The position of the stepper motor is only known incrementally. In case of missed step (this can happen with a big load or insufficient current), the motor's real position is different from desired.

On the other hand, servomotors are easier to control because they usually contain a control mechanism based on negative feedback, so the servo's position is always known. Servomotors are also more resistant to load and external forces because they contain a gearbox. Servos are mainly the source of effort (momentum) instead of speed. This makes servomotors more suitable for the majority of manipulators, including the manipulator described in this thesis.

2.2.1 Used servomotor Dynamixel AX-12A

Type of the servo was consulted with Computational Robotics Laboratory because they are constructing crawling robots. They use Dynamixel's AX-12A servos. This type of servomotor is cheap, reliable and easy to control, so it was also recommended for usage in the manipulator. Detailed information can be found at [**?**]. Servo is shown in figure 2.2.

Figure 2.2: Dynamixel AX-12A

2.2.2 Basic infromation

This type of servo can operate in two modes. It operates as usual motor in wheel mode but the position control is not available in this mode. Second option is joint mode which is used in this application.

The operation range in joint mode is 300 degrees divided to 1024 steps which makes the resolution of 0.29°. With the load on the servo, inaccuracies may occur. It is caused by gearbox in the servo, respectively tolerations between gears etc. However, those inaccuracies are insignificant for described usage, same as inaccuracies caused by deformation of materials and high leverage applied to the motor.

The maximum speed in torque mode is 59 RPM. That's more than enough. Speed could be lowered by changing the gear ratio to increase torque. Servo is able to deliver 1.5 N m of torque. Taking the size of the servo into account, this is really good performance.

Maximum torque, clockwise and anti-clockwise, servo identification number, voltage and temperature limits, baud rate and many others can be stored on EEPROM. Servo also has RAM volatile memory which is wiped after power off. The goal position of the servo, moving speed, LED status etc., can be set via RAM. Information like present position, speed, load, voltage or temperature can be also be obtained from RAM.

Other specifications of servo are in the table 2.1.

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: AX-12A servomotor specifications

Control signals are delivered to the servo via half-duplex UART. The communication protocol is described at 2.2.3.

2.2.3 Communcation protocol

This section describes communication protocol of Dynamixel AX-12A

Half-duplex UART

Servo uses Dynamixel Protocol 1.0. As mentioned in 2.2.2, servo uses half-duplex UART communication. This type of communication is realized using

only one wire. This creates a need to send a confirmation message after the end of transmission. This message contains information for the second that communication is over and the device is ready for receiving data after some delay. Each servo on the bus has its unique ID in range 0 253 used for addressing servo. ID 254 is broadcast ID.

Instruction packet

Commands for the servo are sent via instruction packets. Structure of instruction packet is shown in table 2.2. First two bytes are headers, third one represents servo ID, fourth is length (number of *parameters + 2*, where byte of the instruction and checksum are counted) followed by instruction, parameters and, finally, checksum. Possible instructions are ping, read, write, etc. List of all instructions can be found at [**?**].

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

Table 2.2: Instruction packet

Status packet

After servo receives instruction packet, it answers with status packet, which may contain error codes in case of failure (eg. overheating, instruction out of range, angle limit error, etc.). Structure of status packet is in table 2.3.

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

Table 2.3: Status packet

Using protocol

The protocol can be used with Dynamixel's USB2DYNAMIXEL converter and DynamixelWizard software, which provides a graphical interface for controlling the servos. But for common usage, it's not necessary to have such a device. All instructions can be sent via UART using STM32 processor. Implementation of the protocol is described in 4.2.2.

2.3 Control Board

Hardware assembly of control board will be described in this section.

2.3.1 ST32F042K6T6 microcontroller

ST32F042K6T6 is an ARM Cortex-M0 32-bit RISC microcontroller from STMElectronics operating up to 48 MHz. It has 32 KB of Flash memory and 6 KB of SRAM. This microcontroller offers one I^2C, two SPI, HDMI-CEC, two USART, USB and CAN interfaces. It also provides ten channel 12-bit ADC, four 16-bit timers, one 32-bit timer and a PWM timer. Its typical voltage is 3.3 V but I/O pins are 5 V tolerant.

This type is the lowest tier STM32 microcontroller with USB connectivity. USB is important, because microcontroller can emulate virtual COM port, so there is no need for using an external FTDI. It saves costs and also space on the board.

USB

USB (Universal Serial Bus) is a commonly used communication protocol. Transmission is an asynchronous serial based on master-slave technology. Using host, hubs, and ports, USB can be used for communication, connection, and power supply between computers, peripherals, and other devices. The connection is provided via eleven different connectors. USB comes in four generations. Our board uses USB2 and MiniUSB connector.

2.3.2 Nucleo-F042K6

Nucleo-F042K6 (on figure 2.3) is a development board with an ST32F042K6T6 microcontroller. Its pinout is Arduino compatible and integrates ST-LINK/V2-1. This makes the board ideal for development because ST-LINK provides debugging options without external debugger. It's also mbed enabled but

as mentioned in 4.1.1, Hardware Abstraction Layer in STM32CubeIDE was used.

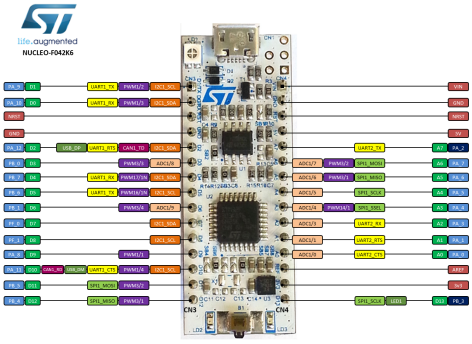


Figure 2.3: Nucleo-F042K6

2.3.3 PCB

The PCB 2.4 is very simple and it is the same board that was also used in [8]. This board design is shared between these two theses. Many thanks belong to the supervisor for help with creating this board.

The board assembly consists of Mini-USB connector, microcontroller, a molex connector for servos and few capacitors.

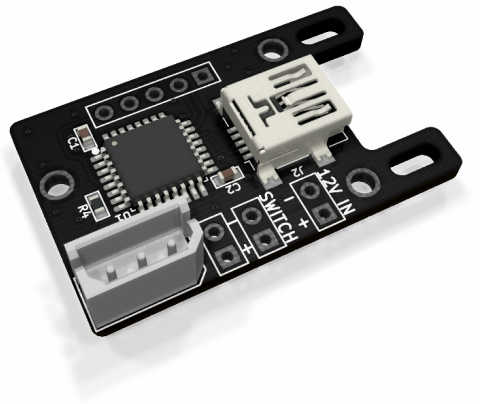


Figure 2.4: PCB

2.4 3D models

Manipulator consists of both parts that came with servos and also custom parts. These parts are described in this section. All models were designed in a Fusion360 software from Autodesk.

2.4.1 Nozzle

Laminar flow

A laminar flow of fluid is a flow where particles follow a smooth path in layers side by side without mixing. Unfortunately, the usage of a pump does not result in laminar flow but in a turbulent flow. This causes that the stream at the output of the nozzle is chaotic. It consists of many streams in different directions. Unfortunately, all laminar flow nozzle designs found are too big for the usage with discussed setup.

Nozzle design

Since nozzle with laminar flow is hard to make, let's stick to nozzle without laminar flow compensation. This design is as simple as possible. The inner diameter of the hose is 12 mm. The nozzle is 21.5 mm long. The inner diameter of the nozzle is 9 mm at the beginning and it gradually narrows to 4 mm after 15 mm. The diameter remains the same until the end of the nozzle. There are also protrusions to keep the hose attached to the nozzle.

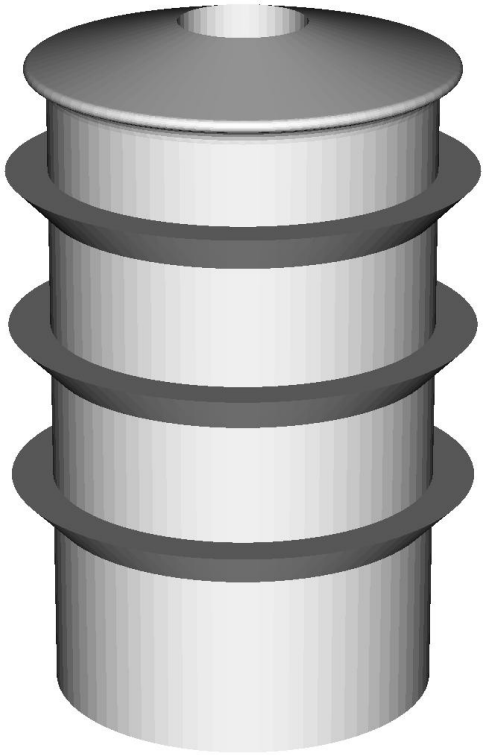


Figure 2.5: Nozzle

2.4.2 Manipulator design

A combination of pre-made parts that came with servos and custom parts was used to assembly a manipulator. Image 2.6 shows the whole assembly. The bottom part (2.7) was designed to be screwed into the first servo and to the base and counts with cables coming from the servo. The link between servos is a pre-made part from Dynamixel. Also part attached to the second servo is pre-made. Custom made part with hose holders (2.8) is screwed to the manipulator with bolts and nuts.

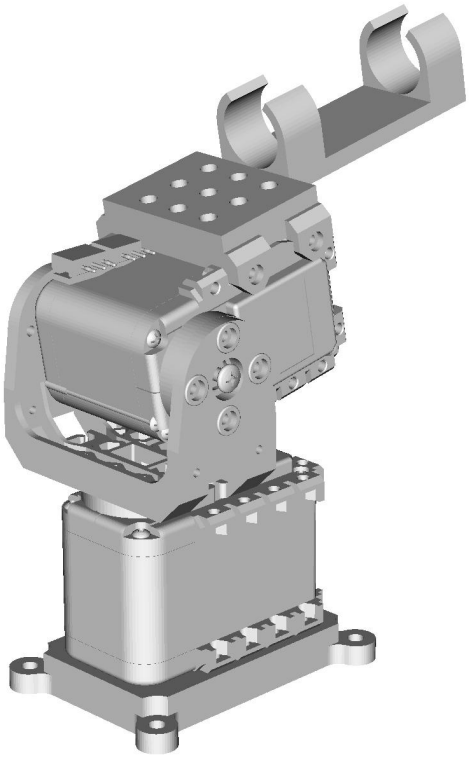


Figure 2.6: Manipulator

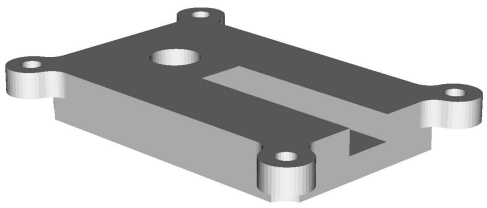


Figure 2.7: Manipulator holder

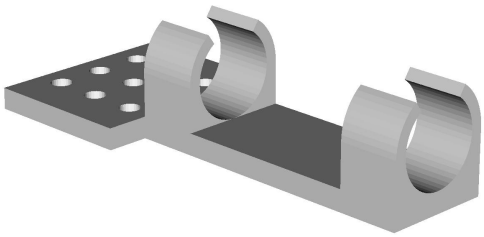


Figure 2.8: Hose holder

Chapter 3

Manipulator

3.1 Manipulator properties

The designed manipulator is an open kinematic chain that an acyclic graph can describe. The manipulator has two degrees of freedom. Both degrees of freedom are rotational joints. Servomotor attached to the base manipulates in angle α, the second servomotor manipulates in angle β.

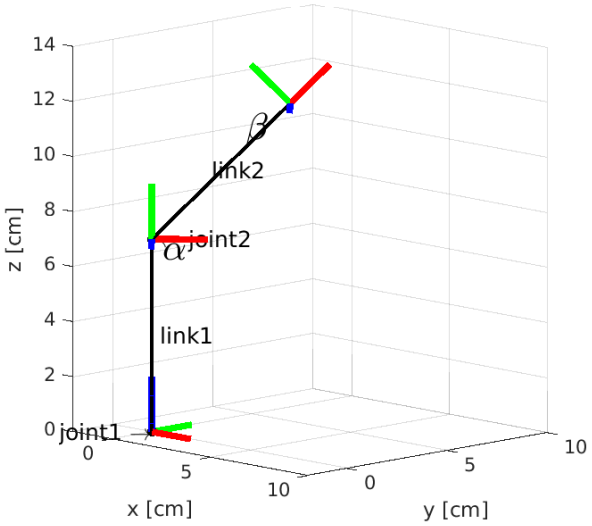


Figure 3.1: Manipulator scheme

Thanks to the positioning of servos, a kinematic chain is very simple. If we put the origin right under both actuators, it's straightforward to do an inverse kinematic task.

3.2 Inverse kinematic task

The inverse kinematic task is a method to find the shift in linear joints and angles in rotational joints from endpoint coordinates. It is often used in robotics. We have only two rotational joints in this case, so just two angles have to be found.

End-point is on the wall where the fire outbreak is located by sensors. The sensor post-processing is not part of this work, so we are given just the endpoint coordinates.

3.2.1 Angle α

Angle alpha is azimuth or direction of fire outbreak. It's can be calculated straightforwardly from coordinates x and y:

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

where x and y are coordinates of the endpoint.

3.2.2 Angle β

Calculation of angle β will be slightly harder but it is possible to do it analytically. Squirting water can be considered as a projectile. The equations considering the initial speed v_0 are:

$$d = v_0 t \cos(\beta), \tag{3.2}$$

$$h = v_0 t \sin(\beta) - \frac{1}{2} g t^2 + link_1, \tag{3.3}$$

where d is distance from target, t is time, g is gravitational acceleration and $link_1$ is length of the first link which offsets the whole manipulator up.

Notice that second $link_2$ is not involved in formula (3.2). It is almost impossible to involve the length of the second link into equations. It leads to complicated equations without an algebraic solution. Thus, the length of the second link is omitted. The error caused by this is minimal considering the turbulent flow in the nozzle and other inaccuracies. Projectile motion can be seen at figure **??**.

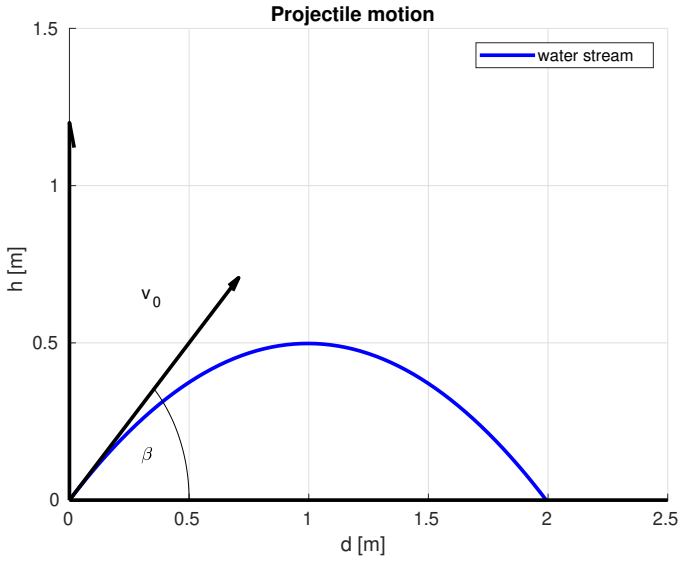


Figure 3.2: Projectile motion

The distance d between fire outbreak and origin can be computed as

$$d = \sqrt{x^2 + y^2 + (z - link_1)} \tag{3.4}$$

At this moment, we need to know the initial speed v_0. We can calculate this value from the volumetric flow of the pump (2.1). The speed at the output of the pump is

$$v_1 = \frac{Q_V}{S_1}, \tag{3.5}$$

where S_1 ís cross section at the output of the pump.

Now we need to get speed at the end of the nozzle. To calculate this we can use continuity equation for fluids in integral form:

$$\oiint_{S(V)} \rho \vec{v} d\vec{S} + \frac{\partial}{\partial t} \iiint_V \rho dV = 0 \tag{3.6}$$

Since we are considering the flow is continuous, we can omit the right part of the (3.6) equation because neither speed nor density is variable in time. So we get:

$$\oiint_{S(V)} \rho \vec{v} d\vec{S} = 0 \tag{3.7}$$

Now if we consider that speed $\vec{v}$ is perpendicular to the surface S, we get:

$$\oiint_{S(V)} \rho \vec{v} d\vec{S} = -\oiint_{S_1} \rho v_1 dS_1 + \oiint_{S_0} \rho v_0 dS_0 = 0, \tag{3.8}$$

where S_1 is cross section of the hose at the pump, S_0 is cross section of the end of the nozzle, v_1 is flow speed at the pump.

Because density is independent on time, we can omit it:

$$v_1 S_1 = v_0 S_0 \tag{3.9}$$

$$v_0 = \frac{v_1 S_1}{S_0} \tag{3.10}$$

If we substitute for v_1 from (3.5) we get:

$$v_0 = \frac{\frac{Q_V}{S_1} S_1}{S_0} \tag{3.11}$$

$$v_0 = \frac{Q_V}{S_0} \tag{3.12}$$

Now is the problem represented as the system of two equations ((3.2), (3.3)) with two unknowns, which can be solved straightforwardly. We express t (3.2) and put it to (3.3):

$$t = \frac{d}{v_0 \cos(\beta)} \tag{3.13}$$

$$h = \frac{v_0 d \sin(\beta)}{v_0 \cos(\beta)} - \frac{g d^2}{2 v_0^2 \cos^2(\beta)} + link_1 \tag{3.14}$$

We are now getting all occurrences of β in tan function using trigonometric identities:

$$h = d \tan(\beta) - \frac{g d^2}{2 v_0^2} (1 + \tan^2(\beta)) + link_1 \tag{3.15}$$

After substitution $u = \tan(\beta)$ we can solve the quadratic equation:

$$\frac{gd^2}{2v_0^2}u^2 - da + \left(\frac{gd^2}{2v_0^2} + h - link1\right) = 0 \tag{3.16}$$

Now we can enumerate the discriminant D:

$$D = d^2 - 4\frac{gd^2}{2v_0^2}\left(\frac{gd^2}{2v_0^2} + h - link1\right) \tag{3.17}$$

This leads to the solution of quadratic equation:

$$u_1, u_2 = \frac{d \pm \sqrt{d^2 - 4\frac{gd^2}{2v_0^2}\left(\frac{gd^2}{2v_0^2} + h - link1\right)}}{\frac{gd^2}{v_0^2}} \tag{3.18}$$

After substituting back is angle β computed.

$$\beta_1 = \tan(u_1) \tag{3.19}$$
$$\beta_2 = \tan(u_2) \tag{3.20}$$

Since we have two solutions, it is necessary to choose one of them. Smaller angle more likely leads to shorter distance that water has to pass. Too big angle can also lead to situation where vehicle is spraying itself, so the final value of beta is:

$$\beta = \min\left(\beta_1, \beta_2\right). \tag{3.21}$$

The inverse kinematics task is solved and now it is necessary to find the maximum reachable area.

3.2.3 Reachable area

Both solutions (3.20) allow water stream to reach different places but there is a boundary which bounds all the points the water stream can reach. This

boundary is located in the points where discriminant $D = 0$:

$$D = d^2 - 4\frac{gd^2}{2v_0^2}\left(\frac{gd^2}{2v_0^2} + h\right) = 0 \tag{3.22}$$

$$d^2v_0^2 - \frac{g^2d^4}{v_0^2} - 2gd^2h = 0 \tag{3.23}$$

$$h = \frac{v_0^2}{2g} - \frac{gd^2}{2v_0^2} \tag{3.24}$$

The boundary can be seen at 3.3

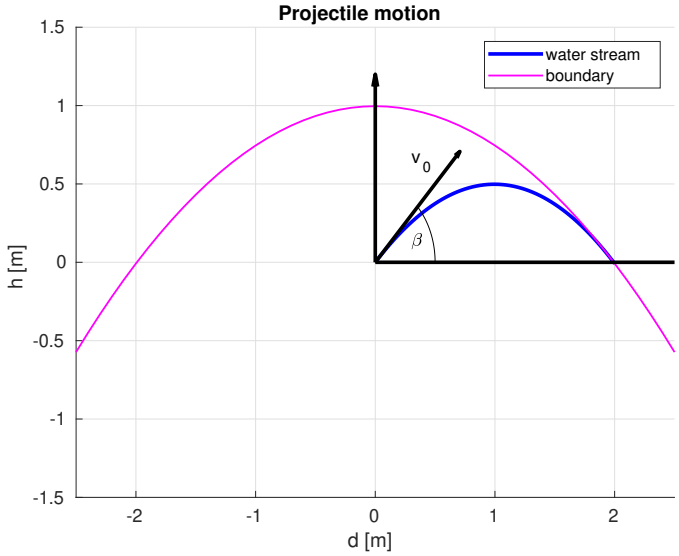


Figure 3.3: Reachable area

There are further limitations. The servo itself has only 300° range, so the limitation for angle α is $\alpha \in [-150°, 150°]$. The angle β is limited so the water can't hit the vehicle under it: $\beta \in [0°, 60°]$.

Chapter 4

Software

Software is a very significant part of this work. The used hardware has to be controlled appropriately. The main part of the program was written in the C programming language in the STM32Cube environment. It runs on the STM32F042 processor described in 2.3.1 and operates with the manipulator and other hardware. The other part was written in Python language and it runs on the computer that communicates with the board via USB (2.3.1). Finally, MATLAB was used to create graphs.

4.1 STM32

STM32 is family of 32-bit microcontrollers from STMicroelectronics. They use 32-bit ARM architecture with processor core Cortex. This provides a wide range of usage. They also develop a lot of related hardware and software

4.1.1 STM32Cube

STMicroelectronics developed a set of tools called STM32Cube. It provides tools STM32CubeIDE, STM32CubeMX and many others used to development on STM32 microcontrollers. This environment was chosen because it integrates Hardware Abstraction Layer mentioned in 4.1.1 so there is no need to access the registers directly. It makes the development a lot easier.

STM32CubeIDE

STM32CubeIDE is an integrated development environment (IDE) based on Eclipse IDE. It is used to write programs for STM32 microcontrollers in C or C++ language. Thanks to its integration to Eclipse it is possible to install many plugins for version control, documentation etc. The programmer used for uploading binary files to the microcontroller is integrated, too. It also contains a code analyzer that shows code size and its size in the memory. The IDE also integrated a tool that allows simple control on microcontroller peripherals called STM32CubeMX.

STM32CubeMX

STM32CubeMX is a graphical tool that generates code based on user's input. The user chooses which peripherals will be used and STM32CubeMX generates code using functions from HAL libraries. For example, in this case, GPIO pins for the pump, UART for servos and USB communication for this project was set via this tool.

HAL

Hardware Abstraction Level (HAL) is a library that ensures the portability of the code between different types of STM microcontrollers. An application programming interface (API) hides all peripherals and low-level code into the end-user's library. This leads to disadvantages like higher memory usage and lower speed than direct access to the registers. Another example of HAL that can be used with Nucleo-STM32F042 is mbed but this platform aims to simplicity for the user like another HAL on a different platform, Arduino, so it makes it less convenient for this project.

4.2 Implementation description

The application is divided to several parts, as shown at the diagram 4.1. The first box called sensors is not part of this work because the raw data from

sensors have to be processed. The data from sensor are sent in the form of the vector `(x, y, z, pump)`, where `x, y, z` are coordinates of the target and `pump` is pump trigger are coming from the other parts of the system.

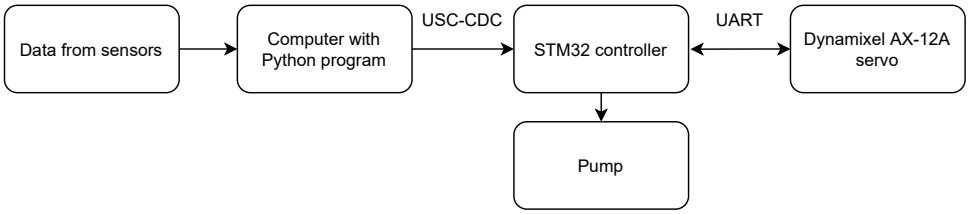


Figure 4.1: Program scheme

As mentioned before, two programming languages were used, C and Python. Part written in C operates on STM32 microcontroller. C programming language was chosen because it is de facto standard in programming microcontrollers. This part of program ensures that received data from vehicle computer are processed and the manipulator is set thanks to servos into desired position. It also provides communication between servos and the program.

The Python part runs on vehicle's computer that gives enough resources to do complex computations like inverse kinematic task from post-processed sensor data. Python programming language was chosen because the mathematical operations are straightforward thanks to libraries like NumPy. The final angles computed from processed sensor data are sent to the microcontroller via USB.

4.2.1 Inverse kinematics implementation

Since inverse kinematics was described in the section 3.2 so this section is only about the program implementation itself. Functions are mainly rewritten mathematical operations. The main point of the program is described by diagram at 4.2. Data from ROS are enumerated in the program.

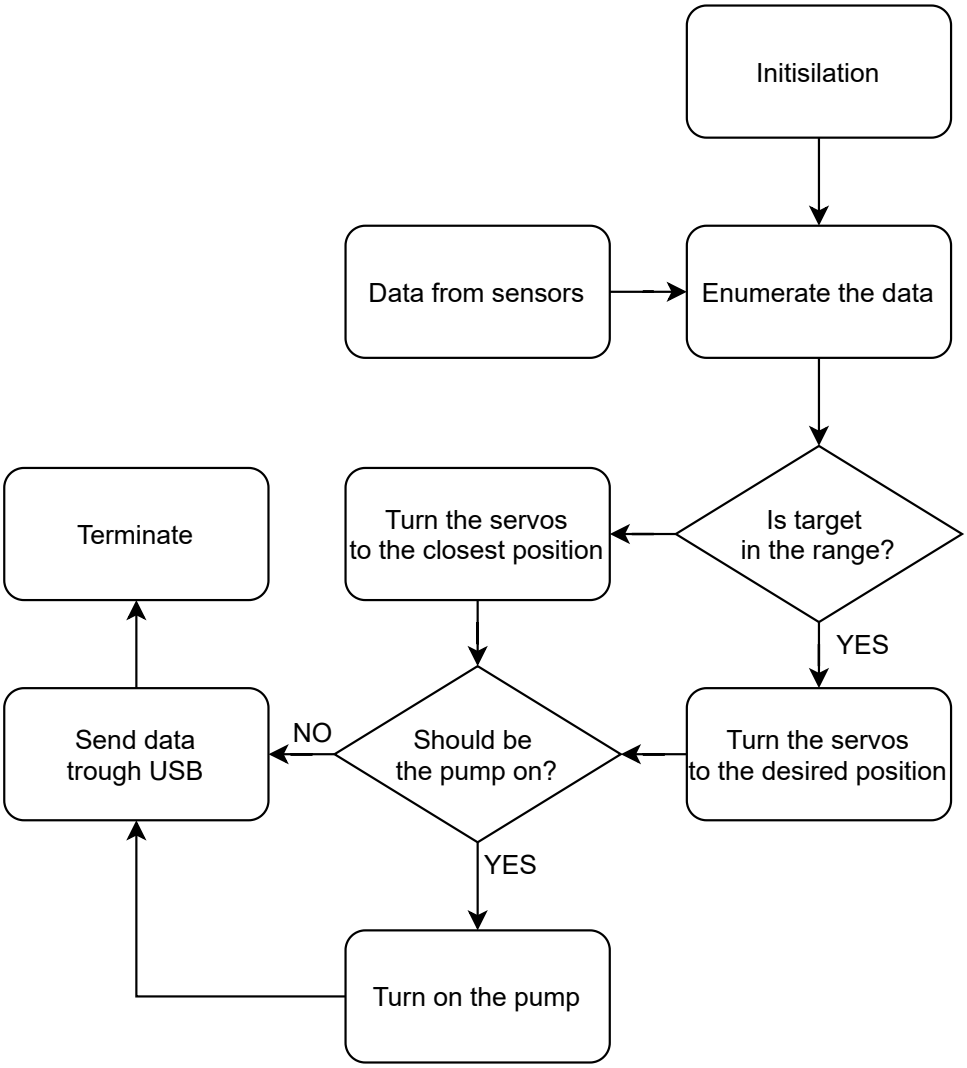


Figure 4.2: IKT implementation scheme

4.2.2 USB communication

The enumerated data are sent via USB to the microcontroller in the format `$alpha$beta$pump`, where `alpha` and `beta` are joints angles in degrees and pump can be 1 or 0. The data are sent through the virtual COM port that is emulated by microcontroller.

4.2.3 Robotic Operationg System

Robotic Operating system used by MRS group is an software environment used for programming robots, originally created at Stanford university. It connects smaller tasks (nodes) into a big system. These nodes are connected together. Simple ROS node that gets information from other drones was written.

4.2.4 Microcontroller implementation

Software on microcontroller consists of initialisation where all peripherals are initialised. After that a infinite loop starts.

Initialisation

In this part, HAL, GPIO and other peripherals are initialised. Also servo settings like clockwise and anti-clockwise angle limits are set. This is achieved through Dynamixel protocol.

Infinite loop

The infinite loop contains a queue of messages from the computer. There is an attempt to pop from the queue every iteration. If there are some data in the queue, these data are processed and translated to the angles α and β and the information about pump. The loop is written that the latest data from the sensor are processed.

These data are sent to the microcontroller via half-duplex UART to the servos. The communication is described in 2.2.3. The status packet is also received.

Chapter 5

Real life experiments

Unfortunately I did not have access to the suitable ground vehicle during fall 2021 so I made experiments only with the manipulator mounted on the piece of wood (see 5.1). These experiments showed that the manipulator works and the water is delivered to the desired location and the inverse kinematics task is computed properly.

Figure 5.1: Real manipulator

Another downside is the pump. Tests showed that 800 $lh^{-}1$ is not enough, so one of the improvement would be stronger pump so the distance can be longer.

Chapter 6

Conclusion

The goal of this thesis was to develop fire extinguisher for ground unmanned vehicle inspired by the MBZIRC 2020 competition. The system consists of manipulator, the pump, control board and the system. Since I did not have access to the vehicle, I used a bucket as a tank but the tank would be made to measure the used vehicle.

The developed software consists of two parts - Python program and microcontroller program. The Python program enumerates the inverse kinematics task and sends data to the microcontroller. The microcontroller enumerated the data and sends commands to the servos.

The inverse kinematics was firstly simulated in Python and then tested with real manipulator and water. Unfortunately, I was not able to access a suitable vehicle so only manipulator itself was tested. The manipulator managed to point the water stream in to the desired direction.

6.1 Future improvements

The design should be tested on real ground vehicle so the missing part with water tank could be done. There are also possible software improvements like bidirectional communication between computer and microcontroller.

Bibliography

[1] *STM32F042K6* [online]. [2022-12-28]. Available from: https://www.st.com/en/microcontrollers-microprocessors/stm32f042k6.html

[2] *STM32F0 HAL description* [online]. [2022-12-28]. Available from: https://www.st.com/resource/en/user_manual/dm00122015-description-of-stm32f0-hal-and-lowlayer-drivers-stmicroelectronics.pdf

[3] *Dynamixel protocol 1.0* [online]. [2021-12-28]. Available from: https://emanual.robotis.com/docs/en/dxl/protocol1/

[4] *STM32CubeIDE* [online]. [2022-12-28]. Available from: https://www.st.com/en/development-tools/stm32cubeide.html

[5] FAROOQ, Adil, Muhammad IRFAN, Abdullah SAEED, Tayyaba ANSAR, Sundas ARSHAD a Narumol CHUMUANG. Design and Development of a Low-Cost Indigenous Solar Powered 4-DOF Robotic Manipulator on an Unmanned Ground Vehicle. *2019 14th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. IEEE, 2019, 2019, , 1-5. ISBN 978-1-7281-5631-6. Available from: doi:10.1109/iSAI-NLP48611.2019.9045302

[6] IMDOUKH, Abeer, Ahmed SHAKER, Aya AL-TOUKHY, Darin KABLAOUI a Mohammed EL-ABD. Semi-autonomous indoor firefighting UAV. *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, 2017, 2017, , 310-315. ISBN 978-1-5386-3157-7. Available from: doi:10.1109/ICAR.2017.8023625

[7] GUPTA, Anik Das, Zarif BIN AKHTAR, Mithun Chandra SARKAR, Tanshen DHAR a Pranta DAS. Unmanned Disposal Rover Along

With Fire Extinguishing Capacity on Both Ground and Air. *2019 Global Conference for Advancement in Technology (GCAT)*. IEEE, 2019, 2019, , 1-5. ISBN 978-1-7281-3694-3. Available from: doi:10.1109/GCAT47503.2019.8978425

[8] PROCHÁZKA, Ondřej. *Robotic Fire Extinguisher Mounted to an Unmanned Aerial Vehicle*. Prague, 2020. Dostupné také z: http://hdl.handle.net/10467/87699. Bachelor thesis. Czech Technical University in Prague.

[9] HEREDIA, G., A.E. JIMENEZ-CANO, I. SANCHEZ, D. LLORENTE, V. VEGA, J. BRAGA, J.A. ACOSTA a A. OLLERO. Control of a multirotor outdoor aerial manipulator. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, 2014, , 3417-3422. ISBN 978-1-4799-6934-0. Available from: doi:10.1109/IROS.2014.6943038

[10] *Dynamixel AX-12A* [online]. [2021-08-09]. Available from: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/