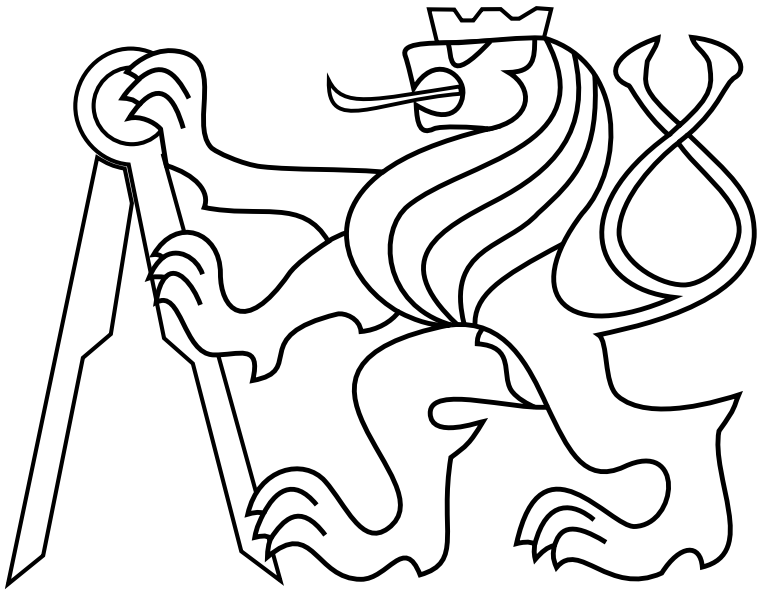
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

# BACHELOR'S THESIS



Daniel Smrčka

**Controlled interaction of an Unmanned Aerial Vehicle with a Wall**

**Department of Cybernetics**

Thesis supervisor: **Ing. Tomáš Báča**

# BACHELOR'S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Smrčka Daniel** Personal ID number: **466509**

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:

**Controlled Interaction of an Unmanned Aerial Vehicle with a Wall**

Bachelor's thesis title in Czech:

**Řízená interakce bezpilotní helikoptéry se zdí**

Guidelines:

This thesis aims to create a system, which would allow an Unmanned Aerial Vehicle (UAV) to interact with a wall. For purposes of measurement or some other task, the UAV may need to lean into a wall, which is detrimental for most of the generally used onboard controllers. To allow the measurements, an admittance control scheme [1] needs to be employed. To achieve the measurement, the student will focus on the following tasks:

- Design and build a hardware mechanism, which will serve as a contact point of the UAV with the wall.
- Integrate necessary sensors to the mechanism, to allow the UAV to sense the moment of contact with the wall.
- Familiarize yourself with the UAV platform [2], its controllers [3], and the simulator environment, which are currently being used in the Multi-robot Systems Group.
- Design and implement an admittance control mechanism that will allow the UAV to maintain an extended contact with the wall.
- Test the control mechanism in simulations and prepare the hardware for experiments, if possible.

Bibliography / sources:

[1] Augugliaro, Federico, and Raffaello D'Andrea. "Admittance control for physical human-quadrocopter interaction." , IEEE 2013 European Control Conference (ECC), 2013.
[2] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska and Vijay Kumar. "Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles". In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2018, 1-8.
[3] Lee, Taeyoung, Melvin Leoky, and N. Harris McClamroch. "Geometric tracking control of a quadrotor UAV on SE(3).", 49th IEEE Conference on Decision and Control (CDC). IEEE, 2010.

Name and workplace of bachelor's thesis supervisor:

**Ing. Tomáš Báča, Multi-robot Systems, FEE**

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

Date of bachelor's thesis assignment: **01.02.2019** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

| Ing. Tomáš Báča | doc. Ing. Tomáš Svoboda, Ph.D. | prof. Ing. Pavel Ripka, CSc. |
|---|---|---|
| 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 .....  .....

signature

## Acknowledgements

Firstly, I would like to thank my thesis supervisor Tomáš Báča for his guidance and his support which allowed me to complete this thesis. Furthermore, I would like to thank Nicolas Staub who introduced me into the problematics and provided me with valuable advice. Finally, I would like to thank my family for their encouragement and support during my whole studies.

*Abstract*

This project deals with design, implementation, and simulation of a compliant mechanism for force interaction of a UAV vehicle with a wall using designed end force effector. The project includes the whole process of mechanical design, control software implementation, and simulation in a robotic simulator with the drone equipped with the designed mechanism. The final result of this project is a control system being able to interact with the wall and stay stabilized autonomously. The mechanism can measure the interacting force and ensures constant contact with the wall by applying a controlled force. The designed system is supposed to be used in documentation of historical buildings to produce ideal light conditions for taking images with better quality than before, without the UAV.

**Keywords:** unmanned aerial vehicles, wall interaction, admittance control

*Abstrakt*

Tato práce se zabývá návrhem, implementací a simulací mechanismu pro silovou interakci UAV se stěnou. Součástí projektu je celý proces mechanického návrhu, implementace řídicího softwaru a simulace v robotickém simulátoru s dronem vybaveným navrženým mechanismem. Výsledkem tohoto projektu je řídicí systém, který je schopen řízené interakce se stěnou. Mechanismus může měřit interagující sílu a zajišťuje stálý kontakt se stěnou působením síly. Navrhovaný systém by měl být používán v projektu zabývající se dokumentací historických budov, aby vytvořil ideální světelné podmínky pro pořizování snímků s vyšší kvalitou než dříve, bez UAV.

**Klíčová slova:** bezpilotní letouny, interakce se zdí, admittance control

# Contents


**1 Introduction** **1**

1.1 State of the art . . . . . 2

1.2 Problem Definition . . . . . 2

**2 Preliminaries** **4**

2.1 Control system . . . . . 4

2.1.1 Hardware . . . . . 5

2.1.2 Control Software . . . . . 5

2.2 ROS . . . . . 6

2.2.1 File system . . . . . 7

2.2.2 Computation . . . . . 8

2.3 Gazebo . . . . . 9

**3 Force measuring** **10**

3.1 Direct force measuring . . . . . 10

3.1.1 Conversion through flexible part . . . . . 11

3.1.2 Intrinsic conversion . . . . . 12

3.2 Indirect force measuring . . . . . 13

3.2.1 Photoelectric sensors . . . . . 13

3.2.2 Inductive sensors . . . . . 13

3.3 Comparison . . . . . 14

3.3.1 Summary . . . . . 15

**4 Wall mechanism design** **16**

**5 Feedback control** **19**

**6 Verification in simulation** **24**
6.1 Mechanism simulation . . . . . 24
6.2 UAV simulation . . . . . 26

**7 Hardware verification** **31**

**8 Conclusion** **34**
8.1 Future work . . . . . 34

**Bibliography** **35**

**Appendix A CD Content** **38**

**Appendix B List of abbreviations** **39**

**Appendix C Technical drawings of model components** **40**

# List of Figures


2.1 UAV platforms developed by MRS group. . . . . 4
2.2 Description of components on the F550 platform [1] updated to solved challenge. 5
2.3 Control pipeline of the MRS UAVs [2]. . . . . 6
2.4 Typical structure of the ROS package [3]. . . . . 7
2.5 File structure of the tracker package. . . . . 8
2.6 Ros communication between master and ROS nodes [4]. . . . . 9
2.7 A screenshot from Gazebo simulator with single UAV and a pine tree. . . . . 9

3.1 Division of force sensors. . . . . 10
3.2 Example of the Wheatstone bridge diagram and the button load cell. . . . . 11
3.3 Working principle of the piezoelectric effect. . . . . 12
3.4 Working principle of "ToF" sensors. . . . . 13
3.5 Working principle of the inductive sensors [5]. . . . . 14

4.1 Possible directions of the approach of the UAV to the wall with highlighted body frame $[x, y, z]$ of the UAV. . . . . 16
4.2 Parts used in model of the pushing plate. . . . . 17
4.3 Components used in model. (a) shows the 3D model of the load cell and (b) shows flange bearing which is mounted on the plate. . . . . 17
4.4 Models of assembled parts. (a) shows the base part that has the force sensor at the bottom and (b) shows the assembled pushing plate. . . . . 18
4.5 Final model of the mechanism . . . . . 18

5.1 The UAV position and orientation in the world's frame $[e_1, e_2, e_3]$. The body frame of the UAV is $[x, y, z]$. Orientation of the UAV heading reference is described via $[\theta, \psi, \phi]$. The position and orientation of the body frame with respect to the world frame is devoted by the translation vector $\mathbf{r}$ and rotational matrix $\mathbf{R}(\theta, \psi, \phi)$, which is function of the roll, pitch and yaw respectively. . . 19
5.2 Control pipeline with admittance control implemented. . . . . 20

5.3 Model of the UAV with mounted mechanism. The body frame $[x, y, z]$ is denoted by the yellow arrows. . . . . 21
5.4 Example of the force effect to the moment of the UAV. . . . . 22

6.1 Model of the mechanism . . . . . 24
6.2 Pictures of the final model mounted on the UAV. (a) shows the UAV from general view, (b) and (c) shows details from top and side view. . . . . 25
6.3 Experiments of the model's response to the applied force. . . . . 26
6.4 Snapshot from the simulation when the UAV is stabilized on the wall. . . . . 26
6.5 Snapshots from simulation. (a) the UAV approaches the wall from ideal angle and (b) the UAV detaches from the wall. . . . . 27
6.6 Comparison of the estimated position and reference position from the top view. The wall is indicated by the gray rectangle. . . . . 27
6.7 Comparison of the positions in 3D space when the wall is approached in right angle. The starting point is the place of the takeoff. . . . . 28
6.8 Snapshots from simulation when the angle between the wall and the mechanism is 17.2 °. . . . . 28
6.9 Snapshots from simulation when the angle between the wall and the mechanism is 34.4 °. . . . . 29
6.10 Comparison of positions when the angle is 17.2 °. . . . . 29
6.11 Comparison of positions when the angle is 34.4 °. . . . . 30

7.1 Interface of the sensors with the Arduino. . . . . 31
7.2 Experiments with the force sensor. . . . . 32
7.3 The model of the mechanism mounted. . . . . 33
7.4 Details of the base part of the mechanism. . . . . 33

C.1 Drawing of the pushing plate model. Dimensions are presented in millimeters. 41
C.2 Components used in model. Dimensions are presented in millimeters. . . . . 42
C.3 Base part with the sensor. Dimensions are presented in millimeters. . . . . 43

# 1 Introduction

Through the last ten years, there has been a significant expansion of autonomous systems concerning most of the technological branches. Making other vehicles capable of driving themselves without human assistance became phenom, and many people see the future in it. This work focuses on the autonomous flight of flying vehicles best known as Unmanned Aerial Vehicles (UAVs). These vehicles are small multirotor helicopters which can carry an onboard computer powered with an appropriate battery. The flight time of UAVs is commonly up to 20 minutes. UAVs are very popular and find use in many applications, such as localization of objects in difficult terrain, monitoring of hardly reachable areas and package delivery.

With increasing popularity, the width of possible UAV applications grows as well. One of the latest comes from the culture field, specifically documentation of historical building for its later renovation. Historical buildings are usually huge and unique places yet also very complicated with a large number of hidden corners. To completely document and analyze places such as churches and cathedrals, many people and heavy equipment are required which presents time and financial burden. This challenge is exactly the place where UAVs might be very helpful.

Documentation of historical buildings using UAVs has significant potential in gaining relatively cheap method with high repeatability. Hidden or hard-to-reach places can be reached easily with the UAV, and the process can be repeatable thanks to automatic onboard control. Moreover, besides relying on the ambient light, artificial light can be carried by additional UAVs. Also, to achieve even better light conditions the light can be brought as near to a wall as possible.

This thesis focuses on designing compliant mechanism and control strategy for force interaction of a UAV with a wall. The UAV is capable of the controlled approach to the wall and subsequent stabilization at the wall using a designed mechanism. The core of the thesis is an implementation of the control system which allows the UAV to interact with the wall. The force control approach applied in this work is known as admittance control. It gained its popularity mainly in robotics where it successfully manages safe interaction of robot arms with surrounding objects and people.

The UAV flies in formation based of leader-follower type where the leader carries a camera and picks the spots for documentation. When needed, the leader can command the followers (UAVs with light sources) to approach the wall. For simplification, the followers are allowed to touch the wall in places that are reachable and do not contain complicated shapes which are not possible to stabilize on.

The result is verified in the Gazebo simulator with the mechanism mounted on the UAV. Robot operating simulator (ROS) is used as a programming environment.

## 1.1 State of the art

Force interaction of the autonomous vehicles with objects (including the wall) is a well-studied subject. It poses a subproblem to building intelligent robots which could one day safely move in a human company without danger of hurting anybody. Research in the UAV field is not nearly as significant as the one in robotics, but it also has its contributions.

The University of Siena published a paper presenting control design allowing a quadrotor to exert a 3D contact force through a rigid tool. The stability of the system was studied in detail, and the desired 3D force with a position of the tool-tip in body frame was also considered and tested. It provides insight look into their controller and analysis of the different possible situation. The conclusion is that the system is able to find equilibrium for any desired external force and successfully stabilize the UAV [6].

The Institute for Dynamic Systems and Control from Zurich, Switzerland used admittance control in a paper focusing on the physical interaction of human and the UAV [7]. This approach takes desired coordinates and modifies them based on the external forces because original coordinates might be unreachable. Force estimation is calculated from position and attitude information and then applied into the admittance control equation which can modify apart from the desired position also UAV's velocity and acceleration.

Paper [2] describes an autonomous control system developed by MRS group from the CTU in Prague. This system includes a tracker that allows precise moving along desired trajectories, moreover it includes collision avoidance with other UAVs. Our project focuses on designing a similar tracker that has an admittance approach implemented in.

MRS group at CTU Prague published a paper that focuses on the process of localization, grasping and transporting a magnetic object [8]. It points out details about designing gripper, such as that it should be designed to be as light-weighted as possible. Designed gripper consists of a mountain bracket, a compression shaft with a ball on end, a spring, a spherical joint attached to an end effector, the end effector. It also contains a spring which is put between the compression shaft and mountain bracket to avoid damaging the platform. That gives us useful insight into one option of designing the end effector.

## 1.2 Problem Definition

This thesis presents a solution to the challenge of designing a system that allows a UAV to perform a controlled touch and stabilize itself at a wall using a designed end effector. It is followed by the challenge of hovering nearby the wall where the UAV becomes unstable due to a different airflow which leads to an unpredictable behavior possibly followed by a collision with the wall. The goal is to implement an autonomous control system that is controlled via measured force and modifies the position of the UAV in the space allowing stabilization at the wall. The original purpose of the work is to allow UAV to carry a light source to produce the best light condition for documentation of historical buildings. To achieve that, the light source should be as near to the wall as possible.

The proposed mechanism should be light-weight to allow the UAV to keep its original abilities and dynamics. Force sensors should be included in the mechanism to allow force control stabilization.

The goal of this work is not to design a complete control system for a UAV. The work builds upon previous work of students and researches in the MRS group. The existing control system implemented in the MRS platform provides state estimation and non-linear control of the rotational and transitional dynamics of the UAV. Our primary focus is to extend the existing system by adding the admittance control.

# 2 Preliminaries

This chapter presents the hardware and software used in this work. The first section focuses on the software and hardware used in this work. The second section provides an introduction into the programming language which was used, and the last section talks about the robotic simulator which was used for testing the mechanism.

## 2.1 Control system

The problem solved in this thesis is expected to be part of an ongoing historical building documentation project (NAKI) in which a new UAV platform is being designed and built. This platform (Figure 2.1a) was already tested in controlled flight, but it is not prepared yet in the simulator. Thus a different platform, the F550 (Figure 2.1b), is used as a substitute for simulation since the UAV dynamics is similar when both platforms are enclosed in a control pipeline in Figure 2.3. The proposed wall-attaching mechanism can be tested on both.

(a) A new platform called NAKI

(b) A F550 platform

Figure 2.1: UAV platforms developed by MRS group.

MRS lab research provides various tools to their platforms, for example controlling through Matlab. Nowadays, however, platforms are using mostly the Robot Operating System (ROS). ROS is widely supported, fast growing and includes many features that make the development easier. MRS group is currently using multiple UAV platforms, all with the ability of autonomous flight. They are either quadrocopters or hexacopters, and they all are simulated in the Gazebo simulator.

### 2.1.1 Hardware

A platform that was used in the simulation is hexacopter based on the DJI F550 frame from MRS group. The UAV platform is composed of active members, computational resources and sensor modules [1]. The components are:

- DJI F550 frame - The first building piece around which all other parts are built. This particular frame consists of 6 arms connected in the middle,
- E310 DJI motors - 6 motors placed on every frame arm,
- Intel NUC-i7 PC - an onboard PC with enough computational resources,
- Pixhawk autopilot - a flight controller assuring the low-level system control, it contains a set of sensors including accelerometers, gyroscopes or magnetometer,
- RPLidar - a sensor providing information about the distance of surrounding objects. It is an option that replaces the GPS which cannot be used in buildings.

This core can be easily extended with other sensors and accessories, such as camera, rangefinder, display, possible gripper or end effector. Schematic of the composition of the UAV matching the needs of the solved challenge is in Figure 2.2.

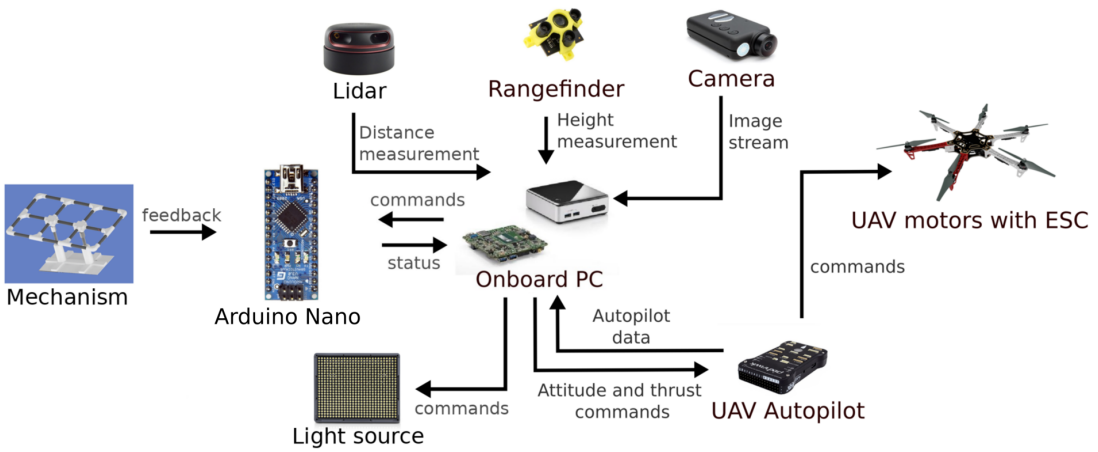



Figure 2.2: Description of components on the F550 platform [1] updated to solved challenge.

### 2.1.2 Control Software

Control software is identical over the MRS platforms which means that implemented control system will be applicable to every UAV platform in MRS group, including future platforms. Figure 2.3 shows the control pipeline which is the core of the control software.

The pipeline starts with a mission planner. It provides direct command or series of commands to the UAV, where commands contain information about high-level behavior.

We can set the UAV desired position, velocities, accelerations or attitude quaternion matrix depending on the control approach.

Collision avoidance and MPC tracker [2] are blocks which allow multiple UAVs to be safe in an enclosed area. The collision avoidance post-processes desired trajectories for the helicopters to provide collision-free flight. Based on shared original trajectories of other UAVs generates new trajectory which is converted via the MPC tracker to vector consisting from commands to position and orientation. The vector is composed of the reference position, velocity, acceleration, and desired heading.

SO(3) controller is non-linear controller which takes that reference vector and converts it to the thrust reference and orientation which goes then to the attitude controller. Attitude controller takes information from SO(3) controller and shifts his data directly into the UAV. Odometry is then evaluated and sent back to the controller and mission planner for further use.

Both the SO(3) controller and the attitude controller solves transitional dynamics that the UAV is supposed to follow. The SO(3) represents the core of the software part, the attitude controller, on the other hand, is an embedded controller and is physically on the vehicle.

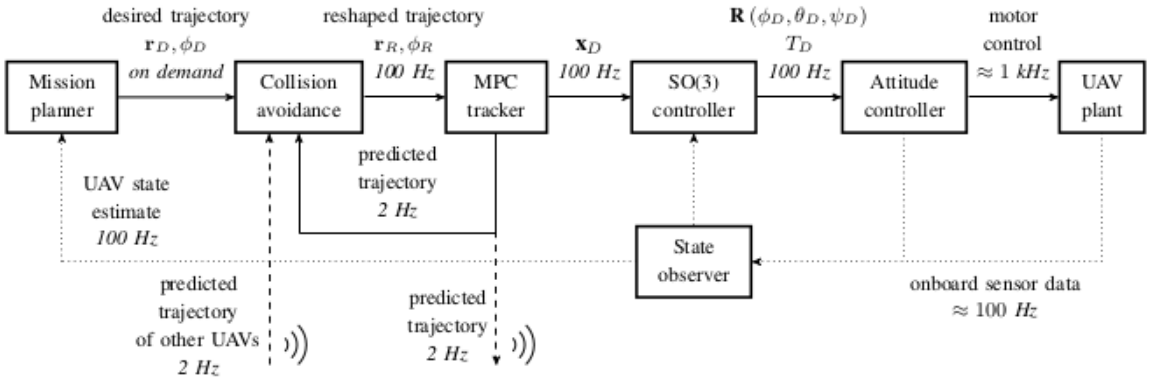



Figure 2.3: Control pipeline of the MRS UAVs [2].

The most important elements to our work are the non-linear controller together with an attitude controller. Purpose of this work is to create tracker allowing the behavior described in Chapter 1.2.

## 2.2 ROS

The Robot Operating System (ROS) is an open-source middleware that includes effective tools and libraries to help developers create robot applications. Its purpose is to make robot software easily applicable to multiple robots with just minor changes in code. The ROS became widely supported mainly thanks to its simple reusability without the need of reinventing things from the beginning and also thanks to its testing opportunities that save much time. Companies and research institutions soon started to develop and share its work which gave the ROS today's popularity [9].

ROS is a Unix-based system, and its main focus is put primarily on Ubuntu or Mac OS X as they are the most used Unix-based operating systems. It provides similar services as a traditional operating system, such as package management, low-level device control or hardware abstraction. However, it does not replace a standard OS, but instead works alongside one and adds a significant amount of tools providing building, writing and running code across the platforms. Next advantage that is welcomed is its language independence. The ROS core and libraries are initially written in `C++`, but Python and Lisp are now also supported, and languages like Java or Lua are already in the experimental process. The whole software is composed of more than 2000 packages that provide all its functionality [10], thanks to that it covers a variety of possible applications.

To be able to write code and effectively use ROS, it is necessary to understand its architecture. It is complicated and might be confusing for the first time. ROS architecture can be divided into two sections [3].

### 2.2.1 File system

ROS has its file system structure. The primary division in files becomes on the packages level. Packages are the most basic units in ROS. They contain all the tools and libraries that are used further in the process of a single package. An important part that is included in every package is called a package manifest. It is a single XML file that contains information about the package, author information or license terms. This file is called "package.xml". The package is divided into several subfolders that all have its purpose. Its typical structure is in Figure 2.4.

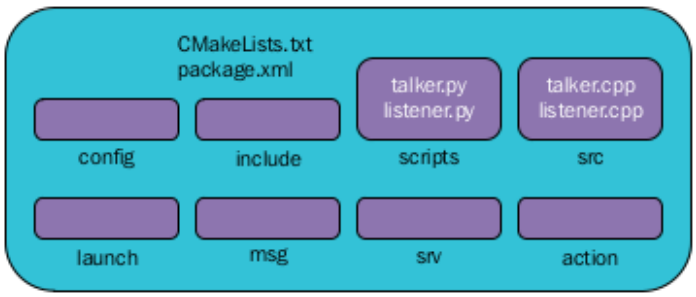



Figure 2.4: Typical structure of the ROS package [3].

*Config* folder serves to save all configuration files that are used in the package. *Include* folder includes links leading to other packages used in the current one. *Scripts* and *src* folders consist of source codes running in the package. *Msg* folder serves to store custom messages definition, and *srv* folder has the same purpose just with storing custom services. *Launch* folder contains launch files that are used to simplify package initialization and run itself. *Action* folder contains action definitions. As the last, every package contains "CMakeLists.txt" that serves to build a package and already mentioned "package.xml" containing package information.

There is one another type of package called Meta package. Meta-packages contains only "package.xml" file, nothing more and servers to gather multiple packages into one single package.


Figure 2.5: File structure of the tracker package.

### 2.2.2 Computation

This section describes how does communication between processes work. The network's core elements that take care of all processes are called ROS nodes. Every program, every data from sensors, every process is run through the ROS node. Whole communication, however, starts with the ROS master (Figure 2.6). ROS master is above the nodes and manages them. The first thing that happens when any node is launched is its registration to the ROS master. Whenever ROS accepts another node, it makes them visible to others. Without the master, the nodes would not see each other [11].

Communication between the nodes is provided via topics. Topics are base on the Publisher-Subscriber principle, where a node that computes some data, for example from a sensor, creates a topic that publishes (offers) these data to other nodes. When other nodes want to read this data, they subscribe to the topic. Data sent through topics are stored in messages. Many messages are pre-built in official libraries, but it is also possible to create a custom message. ROS also allows the request/response communication. This communication is provided via ROS services.

ROS's community structure is a very robust source of information and implemented software. Its open-minded approach is the core of its success, and that is why it is worth mentioning. ROS as many other system compounds from multiple versions and every active version has its community that involves its functionality. General information about the system, including different versions, can be found at a free wiki page [1] that is still updated. Moreover, if the problem is specific, it can be used the ROS answers page which many users use. Thus a solution to the problem can be found there.

[1] `https://wiki.ros.org`

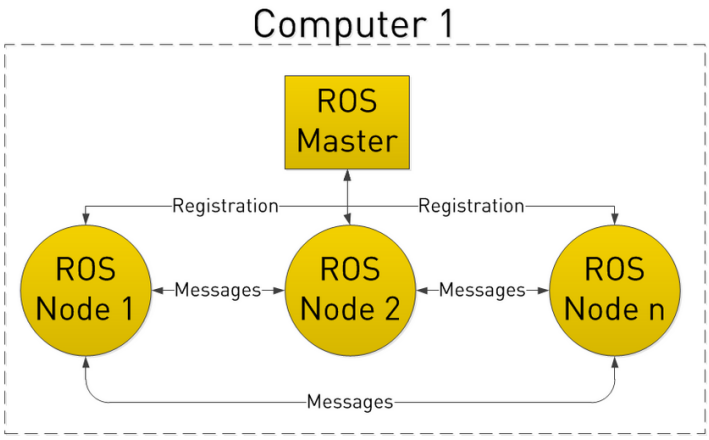



Figure 2.6: Ros communication between master and ROS nodes [4].

## 2.3 Gazebo

Gazebo is a robotic simulator used to create applications for a 3D model of a real robot and then simulate its behavior in indoor or outdoor conditions. It is an open-source solution interacting well with the ROS and is capable of realistic simulation [12]. Figure 2.7 shows an example of a model of the hexacopter UAV with the model of pine in the background run in gazebo simulator.

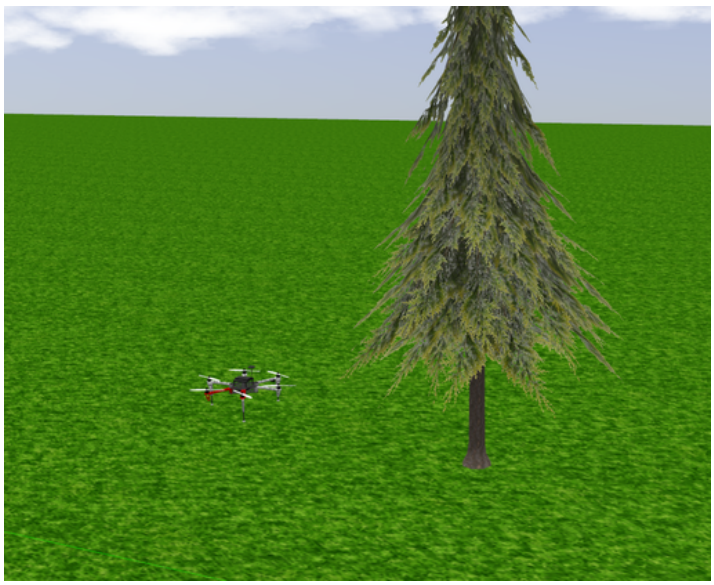


Figure 2.7: A screenshot from Gazebo simulator with single UAV and a pine tree.

Gazebo's biggest advantage is its universality, apart from a model that should be as real as possible to feedback same behavior as the real model, the instance of ROS itself behaves same as the one running on the real robot in real-world conditions.

# 3 Force measuring

It was Sir Robert Hook who as first specified out how is object's deformation connected to an applied force. He noticed that many materials have a linear region in stress-strain material dependence [13]. It led him to the definition of the well-known Hook's law which says, that force $F$ needed to compress or extend the spring has a linear dependency to compression or extension distance. Hook's law, also known as elementary hooks law, equals to

$$F = k\Delta d, \tag{3.1}$$

where $k$ $\mathrm{Nm}^{-1}$ is the spring constant and $\Delta d$ [m] is a distance change (compression or extension). In other words, a force can be measured through material deformation. We can split force measuring into two categories: direct and indirect.

Direct measuring describes sensors that convert applied force directly into easily measurable output, for example, voltage, current or frequency. Indirect measuring describes an option where force is converted from a measured distance as Robert Hook described in his law (3.1). To use this form, the mechanism needs to contain spring with the known spring constant and the measured compression of the spring is then converted to the applied force. There are several options on how to measure distance, such as photoelectric sensors, which use an emitted light beam reflecting from objects back to the sensor or inductive sensors using an electromagnetic field to detect objects. The main drawback with indirect force measuring can be a combination of uncertainties coming from both the sensor and the spring constant.

## 3.1 Direct force measuring

Direct force measuring is not direct in the meaning of explicit reading the force, but it indicates the fact that the sensor itself uses different ways to convert force. According to the fact that force is measured via deformation $\epsilon$ change in a material, we can now focus on how can be the deformation change used to get the applied force. Figure 3.1 shows the basic idea of how are force sensors divided.

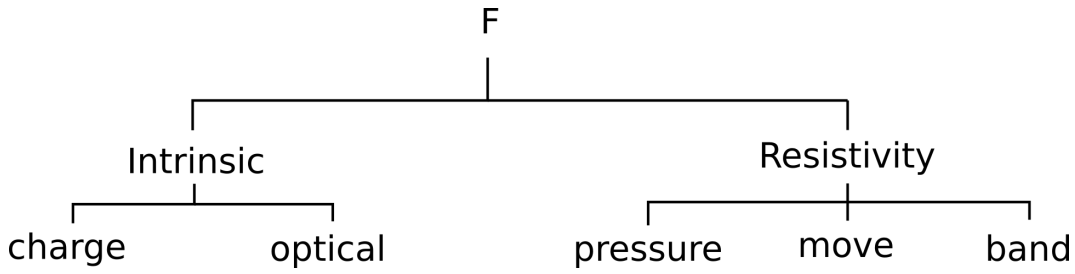



Figure 3.1: Division of force sensors.

It shows that there are two categories of force sensors. One category uses a change of resistance in the flexible part that is exposed to the force and the second one presents methods that use deformation effect to change of specific parameters, such as charge.

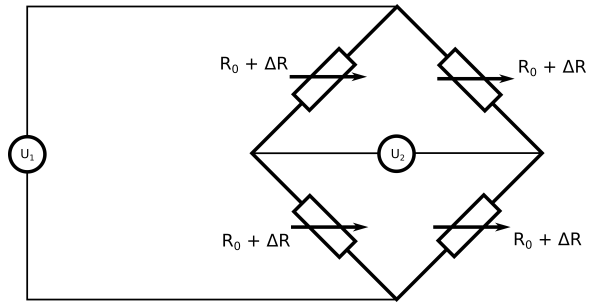
### 3.1.1 Conversion through flexible part

One possibility how to convert a deformation to a measurable output is to use some flexible part, usually called load cell, shaped in a special form (S-shaped part, cylinder). Deformation caused by the applied force is then measured via a strain gauge [14].

**The strain gauge** is a tool that converts tension, stress or torque to measurable output through resistance change. This effect is well-known as the piezoresistivity effect and to its purpose can be used fine wire or glued foil. It is easily applicable, but it also brings to the measuring many problems, such as material changes caused by temperature changes or the creation of disturbing thermoelectric voltage in a circuit. The geometry of the object that is exposed to the applied force is determined with the following rules:

- Material - the goal is to have material that has a minimal thermal expansion, minimal hysteresis, minimal Young's modulus and is resistant to corrosion,
- Directionality - compliance in the direction of applied force and resistance in the direction of unwanted effects (torque),
- Shape - conversion of applied force also to a negative change of deformation to be compatible with the full Wheatstone bridge (Figure 3.2a),
- Overload protection - maximum force should be in 10% - 30 % of Young's modulus,
- Transformation in the range of linearity.

To minimize outer effects applying on the material, it is very effective to use the composition of four strain gauges linked into the full Wheatstone bridge.




(a) Full Wheatstone bridge


(b) The button load cell

Figure 3.2: Example of the Wheatstone bridge diagram and the button load cell.

**Wheatstone bridge** is an electrical circuit commonly used in all resistance measurements. The output voltage is the difference between two voltage dividers. One outputs voltage

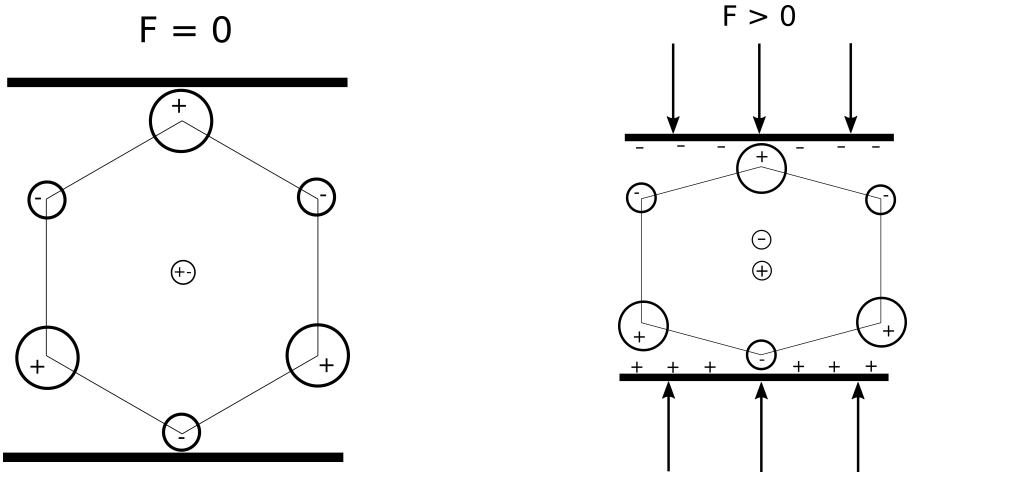
on the resistor with reduced resistance and the other one on the resistor with higher resistance. It reduces the environmental effects as all the strain gauges are exposed to the same conditions and only the resistance difference is measured. Wheatstone bridge can also be used with only two or single active strain gauge while others are changed for the simple resistors.

**Button load cell** (Figure 3.2b) is a specific type of load cell, which is used in this work. The load cells are accurate force sensors with error within 0.03% to 0.25% [15], and they are made in many shapes and sizes. The button load cell is ideal for our work as it is small, light and has a rounded shape. An amplifier that usually comes with the load cell amplifies the output of the load cell to better signal that is then converted in the micro-controller to the force.

### 3.1.2 Intrinsic conversion

Intrinsic measuring of deformation falls into several categories. Deformation can be measured via optical fiber, where mechanical force changes fiber's geometry leading to different refractive index. Alternatively, using the magnetic field of two cores, where the main core does not affect the second one if no force is applied. However, when the force is applied, the main core creates an effect on the second core and its output is then converted to the measured voltage. Nevertheless, the most known way of the intrinsic force measurement uses piezoelectric effect [16].

The piezoelectric effect is an effect where the force changes electrical states in a crystal. If we imagine the electrical state of the crystal (Figure 3.3a), we can see that its charge is neutral. However, when we apply force (Figure 3.3b), charges move, and the surface becomes charged. A necessary condition is that the crystal does not have point inversion symmetry. If it had, no charge would be created. Quartz is an example of the commonly used crystal, it is a mineral composed of silicon and oxygen ($SiO_4$).




(a) Crystal grid when no force applied

(b) Charge move while the force is applied

Figure 3.3: Working principle of the piezoelectric effect.

## 3.2 Indirect force measuring

Indirect force measuring converts the force to compression of a spring, which is then measured. To do so, there is a need for precise distance measuring. This section presents two principles of distance measuring that might be applicable in our work.

### 3.2.1 Photoelectric sensors

The first type of sensors are photoelectric sensors. Photoelectric sensors combine electronics, control and detection of light. The most common type of photoelectric sensors uses a principle called *Time of flight* ("ToF"). Sensors with suffix "ToF" are composed of a transmitter (Light emitting diode), a receiver (Photo-diode) and electronics to amplify the detected signal. The sensor emits a beam of light and measures the time to its reflection from an object. The distance to the object is then calculated as

$$d = \frac{1}{2} \cdot c \cdot t, \tag{3.2}$$

where $c = 299792458\,\mathrm{ms}^{-1}$ is the speed of light in vacuum and $t$ [s] is measured time between the light emission and its reception (Figure 3.4). "ToF" sensors are cheap and relatively precise. A considerable problem with "ToF" sensors is that they are prone to the detection of other light as it scans the whole area to reflected beam.

Another type of photoelectric sensors is usually used in places where "ToF" sensors have struggles and cannot be used. This principle is called interferometry, it uses an emitting beam of light with periodically changing frequency. From phase shift of sent and received signal is then calculated object distance. Maximum range is restricted on the fact that periods cannot repeat until the emitted beam reflects.

Both of the mentioned sensors are used in many industrial applications. Their most significant advantage is in large detection range, typically from 1 mm to 60 m, moreover, it is possible to use them for most of the materials, such as plastic, paper or metal [5].

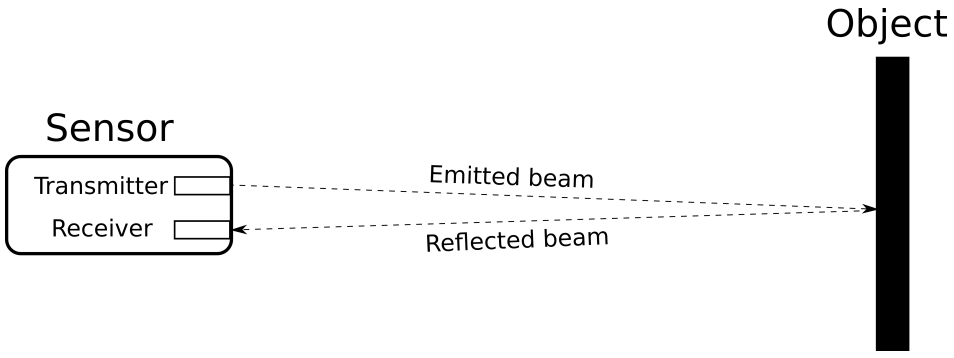



Figure 3.4: Working principle of "ToF" sensors.

### 3.2.2 Inductive sensors

Inductive sensors are very precise sensors used for detection of metal objects. The inductive sensor (Figure 3.5) uses the core with the coil to generate a magnetic field. If a

metal object is near, it changes the magnetic field and the current flows in the object. This current sets up a new magnetic field with opposite direction than the original field. This effect invokes a change of the inductance of the coil in the inductive sensor which can be measured [17]. Its normal range to detect is from 1 mm to 60 mm [5]. The main disadvantage of inductive sensors is that they react only to metal materials (such as iron, steel or aluminum).

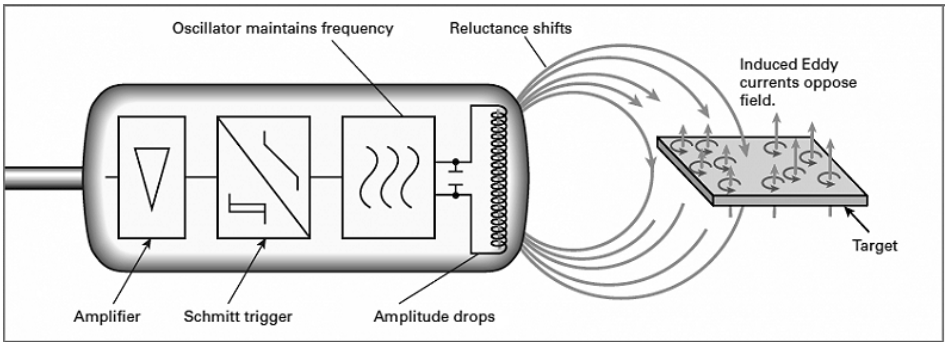



Figure 3.5: Working principle of the inductive sensors [5].

## 3.3 Comparison

To decide which sensor to use we provide a table 3.1 that compares four chosen load cell sensors, three photoelectric sensors, and one inductive sensor. The chosen load cells are all button load cell as they would fit well to the mechanism. Main attention is put on precision.

| Force sensors | Max. weight [N] | Uncertainty [N] | Price [$] |
|---|---|---|---|
| Button load cell 1 | 2000 | ±0.40 | 45 |
| Button load cell 2 | 500 | ±0.12 | 45 |
| Button load cell 3 | 10000 | ±2.00 | 50 |
| Button load cell 4 | 2000 | ±0.60 | 59.95 |
| Photoelectric sensors | Distance [cm] | - | - |
| ToF rangefinder 1 | 4-400 | ±0.60 | 11.95 |
| ToF laser rangefinder | 2-200 | ±0.32 | 29.99 |
| ToF sensor | 10 - 200 | ±0.30 | 15.95 |
| Inductive sensors | Distance [cm] | - | - |
| Ind. sensor | 0.15 | ±0.002 | 10.60 |

Table 3.1: Comparison of options to measure applied force.

Precision of load cells can be found in their datasheet. But to calculate precision in Newtons with use of distance sensors we need to explicitly calculate the uncertainty of measurement. The measured force is dependent on measured distance and on constant of toughness $k$. That is why we need to include both in the measured uncertainty. Thus, we are interested in the uncertainty coming from the measured tools, that means uncertainty of type B ($\sigma_b$), calculated as

$$\sigma_b = \sqrt{k_{error}^2 + d_{error}^2}, \tag{3.3}$$

where $k_{error}$ is the uncertainty caused in the spring constant and $d_{error}$ is the uncertainty from distance sensor. Czech spring company **Vanel** says that the uncertainty of constant at their springs is $\pm 0.15$ [18] and an error from distance sensor can be found at a datasheet.

### 3.3.1 Summary

As it seems from the table 3.1, using photoelectric sensors appears to be the cheapest option and the final error is also reasonable. On the other hand, there is still a chance of error coming from the surrounding area.

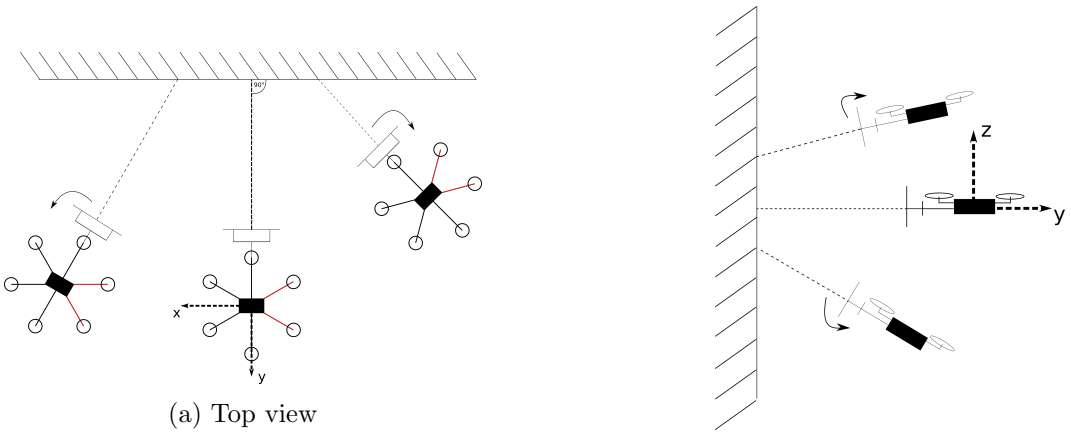
The inductive sensor has the best precision and is surprisingly cheap. The biggest struggle in using inductive sensor comes from its sensitivity. The measuring could be easily affected by an electromagnetic field coming from nearby UAV's motors and wiring or a fact that a UAV is not stable and experiences vibrations which would make a difference in small measurement like the ones measured with the inductive sensors. That means moving spring would have to be very well designed and mounted to prevent this unwanted movement.

The best option for us is to use the button load. Despite the higher price, the application can be much more elegant, more straightforward and we will be getting the immediate force. Although the final precision was similar with measuring distance procedure, using load cell completely erases problem coming from the spring. The spring itself is another element that has its attributes, and one of them is that it is prone to degrade with time due to repeated stress and possible corrosion. All these things make changes to the spring constant and measurements might be soon very inaccurate.

# 4 Wall mechanism design

The mechanism was designed to be as light as possible and as simple as possible. We cannot expect that the UAV will always have the perfect conditions while approaching the wall. The UAV might not approach under the correct angle (Figure 4.1a) or have the ideal tilt (Figure 4.1b). That is why the mechanism cannot be substantial part with minimal movement but have to allow the UAV to correct errors.

Another important feature that will help with stabilization is the size of the end effector. We decided to design the end-effector as a rectangular plate. The size of the plate is important for further control design. If the plate was too small, force sensors would be closer together and it would lead to smaller differences in measured force on each sensor. While approaching in the wrong direction one sensor would be much more affected than the other one, and it would make the stabilization difficult. On the other hand, if the plate was too big, its size might significantly limit the UAV's mobility. That is why the size is the first key step to successful design.



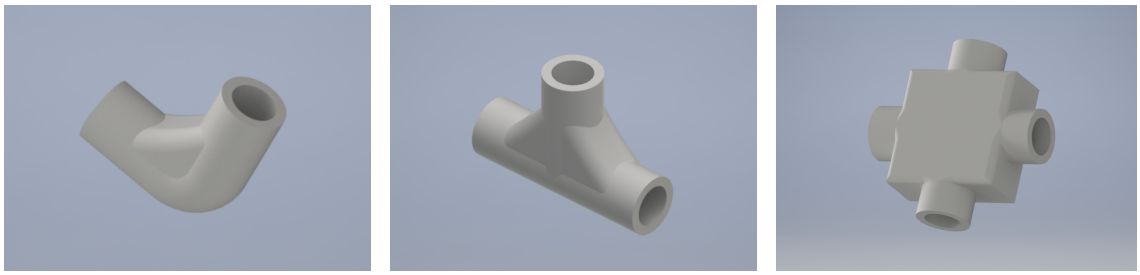

(a) Top view

(b) Side view

Figure 4.1: Possible directions of the approach of the UAV to the wall with highlighted body frame $[x, y, z]$ of the UAV.

The plate is composed of small 3D printed plastic parts connected with carbon fiber tubes to achieve its low weight. Carbon fiber tubes are very light and still solid enough to endure harder touch with the wall.

First, we started modeling the connecting parts of the plate. The parts are the corners, middle parts, and center parts. The corners (Figure 4.2a) have two holes to fit the tubes and both its ends are strengthened to absorb higher impact. The middle parts (Figure 4.2b) are used to connect the corners with the center parts.



(a) Corner part (b) Middle part (c) Center part

Figure 4.2: Parts used in model of the pushing plate.

The center parts (Figure 4.2c) design was prepared to fit special flange bearings which connect the plate with the base part on the UAV via a metal shaft. Using these bearings (Figure 4.3b) saved us from inventing complex mechanism compounding from multiple joints to allow the desired movement.

The desired movement of the mechanism is shown in Figures 4.1. Arrows near the mechanism sketch depict desired behavior in the indicated situation. If the UAV is not in the ideal position to approach the wall, it needs to adapt. The required movement represents two degrees of freedom (DOF) from the UAV position, where DOF is a number of parameters that define the position and orientation of the object in the space. That means allowing the end plate to move in the compression direction (around z-axis, as shown in Figure 4.1a). To allow the UAV appropriate tilt while it is attached to the wall, the mechanism has to move around the x-axis (Figure 4.1b). The chosen flange bearing provides three DOF, movement around the ball joint in all angles. However, when we use two bearings, the initial three DOF to every bearing is reduced to the desired two DOF due to the fixed position of the base part to the UAV. The final model of the pushing plate can be seen on the figure 4.4b.

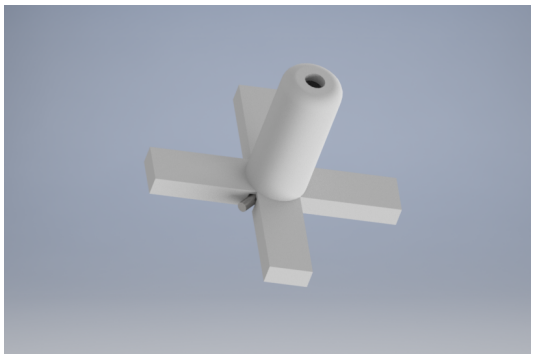
(a) 3D model of a used load cell (b) Spherical joint

Figure 4.3: Components used in model. (a) shows the 3D model of the load cell and (b) shows flange bearing which is mounted on the plate.

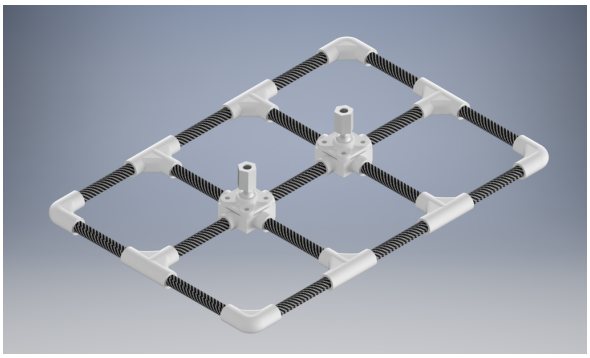
The final part of the mechanism is the base part. It is the part connected to the UAV which includes the force sensors (3D model – Figure 4.3a). To reduce the first impact with the wall and to allow the mechanism desired movement, we decided to put the spring over the force sensor.

It is not still clear how the mechanism will be mounted on the UAV. Figure 4.4a is an

illustration of the possible form of the base part. The important segment of the base part is a center section consisting of the force sensor at the bottom, the spring and space for the connecting shaft.



(a) Design of the base part of the model



(b) Final model of the pushing plate.

Figure 4.4: Models of assembled parts. (a) shows the base part that has the force sensor at the bottom and (b) shows the assembled pushing plate.

3D models were created using the Autodesk Inventor software. It provides a simple and very intuitive environment to model all parts for the 3D print. Figure 4.5 shows all the parts mounted together in the Inventor software. Drawings of the designed parts with marked dimensions are attached in the appendix C.

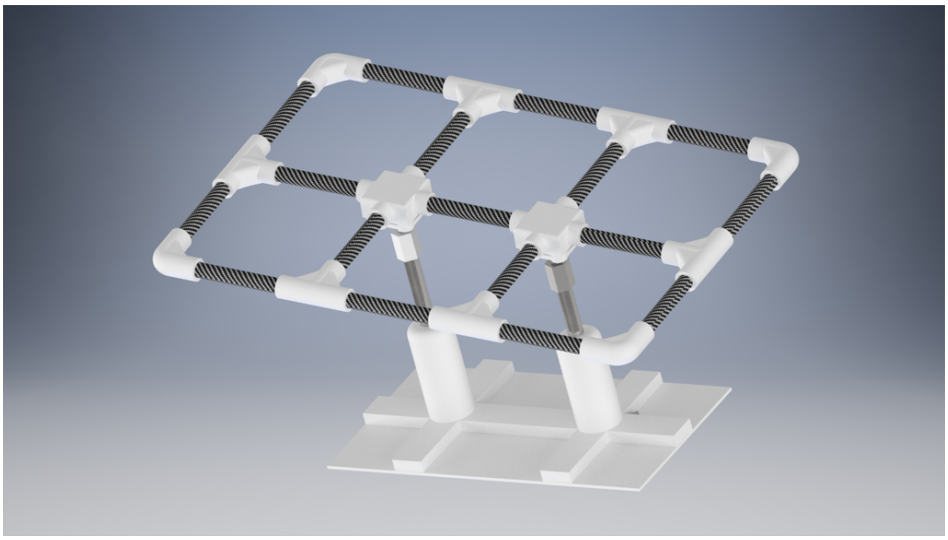


Figure 4.5: Final model of the mechanism

# 5 Feedback control

The following part of the thesis describes the control design and implementation. The original control pipeline (Figure 2.3) of the MRS platform [2] is modified to include the admittance control. The original control was designed to allow the coordinated flight of multiple UAVs. However, the system in the current form is not able to handle interaction with the environment, which would limit the DOF of the UAV.

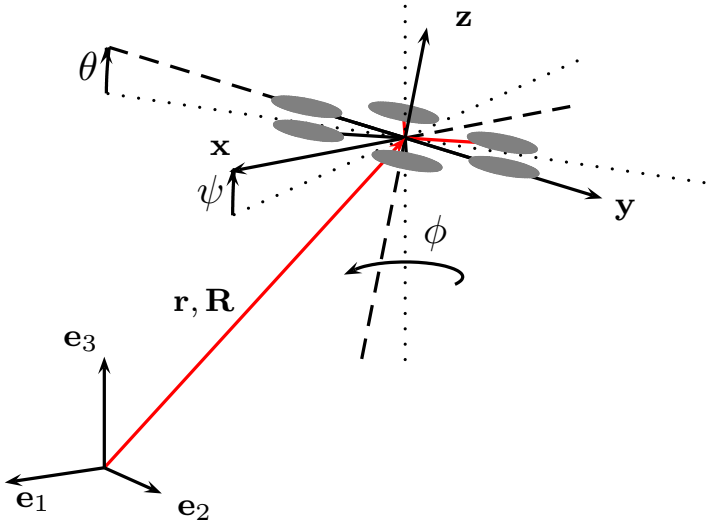



Figure 5.1: The UAV position and orientation in the world's frame $[e_1, e_2, e_3]$. The body frame of the UAV is $[x, y, z]$. Orientation of the UAV heading reference is described via $[\theta, \psi, \phi]$. The position and orientation of the body frame with respect to the world frame is devoted by the translation vector $\mathbf{r}$ and rotational matrix $\mathbf{R}(\theta, \psi, \phi)$, which is function of the roll, pitch and yaw respectively.

To control the system that interacts with objects, it is not possible to use usual approaches. The usual approach can be described as a simple move to desired coordinates without expecting any obstacles or objects in the way. The reaction of a standard UAV to interaction with an object is very unpredictable and depends on the point of contact. When the object is a wall, the UAV probably breaks its propellers first and then drops on the floor. When the UAV would have some protection case, it would suddenly stop moving. Its response would be to try to push harder against the wall and after few attempts, the UAV would try so much that it would lose stability and crashed.

To avoid this behavior, we use an approach called admittance control. It is a control approach which uses measured force applied to the vehicle. Admittance control takes desired coordinates and modifies them based on the external forces. When the UAV reaches the

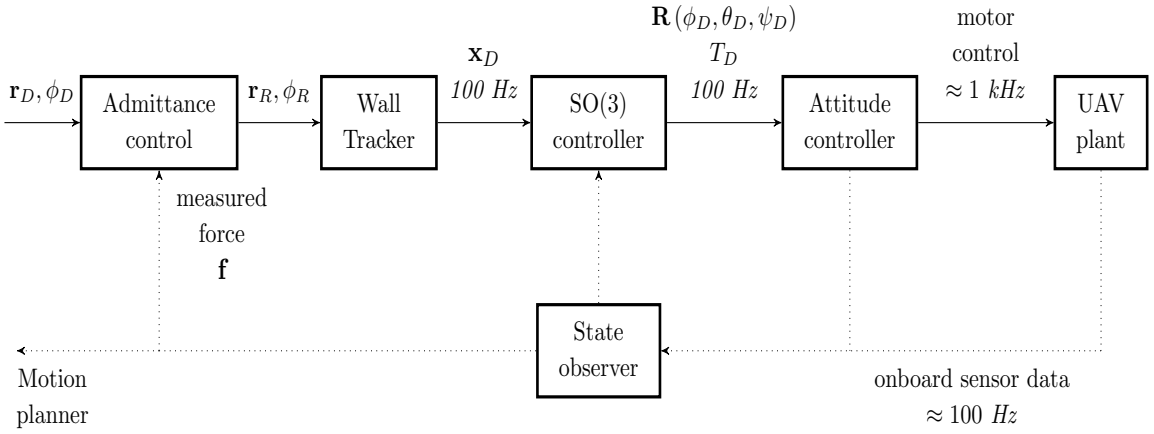



Figure 5.2: Control pipeline with admittance control implemented.

object and cannot move forward, it modifies its desired coordinates to maintain a stable state. Pipeline including the admittance control, can be seen in Figure 5.2.

Admittance controller block modifies desired coordinates $\mathbf{r}_D$ and desired yaw rotation $\phi_D$ based on the external force $f$. Then it outputs reference coordinates $\mathbf{r}_R, \phi_R$ to the designed Wall tracker. This tracker converts the desired position to position commands $\mathbf{r}_D, \dot{\mathbf{r}}_D, \ddot{\mathbf{r}}_D$ and heading commands $\psi_D, \dot{\psi}_D, \ddot{\psi}_D$. The SO(3) controller then produces the orientation and thrust reference. Modification of the desired coordinates is based on following equation

$$M(\ddot{\mathbf{r}}_D - \ddot{\mathbf{r}}_R) + D(\dot{\mathbf{r}}_D - \dot{\mathbf{r}}_R) + K(\mathbf{r}_D - \mathbf{r}_R) = -f, \tag{5.1}$$

where $M$, $D$, $K$ are diagonal matrices defining inertia, damping and stiffness of the vehicle. $f$ are the external forces applying on the vehicle and $\mathbf{r}_D$ and $\mathbf{r}_R$ are desired and reference coordinates [7]. When the vehicle is pushed it will behave according to (5.1).

According to the Figure 5.1, the UAV is described using coordinate systems. The world frame $F_1 = [e_1, e_2, e_3]$, which is the world reference system and the UAV's body frame $F_2 = [x, y, z]$, which has its origin placed in the UAV's center of mass. The position of the center of mass in the world frame is given via the transitional vector $\mathbf{r} = [e_1, e_2, e_3]^T$ and UAV's rotation is expressed by rotational matrix $\mathbf{R}(\theta, \psi, \phi)$. Figure 5.3 shows the body frame on the UAV with the mounted mechanism. The front of the UAV is heading along the x-axis and the mechanism is mounted on the negative side of the y-axis.

To simplify the coordinates modification, we are moving in the body frame coordinates. The mechanism is mounted on the side of the UAV, that is why we are modifying only its y-axis value. The equation (5.1) can be then simplified to

$$M(\ddot{r}_{Dy} - \ddot{r}_{Ry}) + D(\dot{r}_{Dy} - \dot{r}_{Ry}) + K(r_{Dy} - r_{Ry}) = -f. \tag{5.2}$$

Moreover, we control only the position and let the controller set appropriate velocities and accelerations based on the reference position. That simplifies the equation to

$$K(r_{Dy} - r_{Ry}) = -f. \tag{5.3}$$

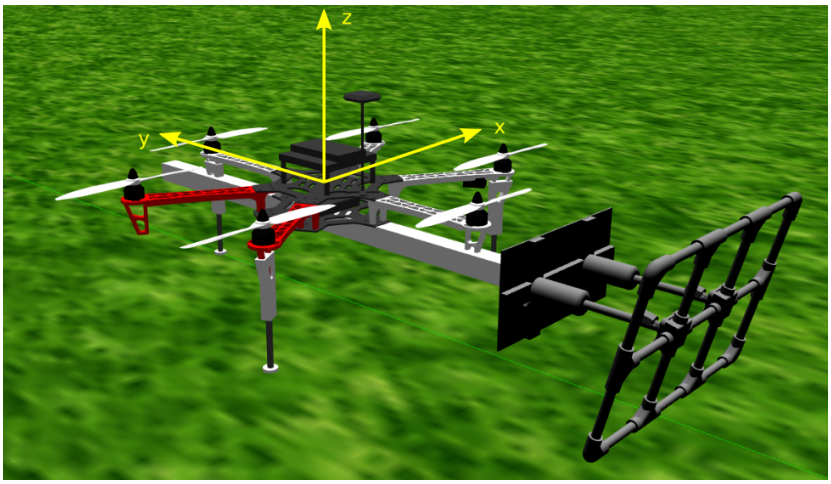



Figure 5.3: Model of the UAV with mounted mechanism. The body frame $[x, y, z]$ is denoted by the yellow arrows.

The reference position $\mathrm{r}_{Ry}$ is expressed as

$$r_{Ry} = r_{Dy} + \frac{f}{K}. \tag{5.4}$$

Modification of the position is not enough to achieve stabilization. Whenever the UAV would approach the wall from the side or while applying force from a different direction, the UAV would try to keep its current yaw. To control the yaw of the UAV, we use the data from both force sensors. Controlling of the yaw is the reason why the mechanism has two sensors. When we know the applied force on both the sensors, the whole applied force is equal to its sum. To calculate the desired yaw angle, we calculate the moment of force applied on the UAV. The moment is given by

$$\tau = \mathbf{r} \times \mathbf{F}, \tag{5.5}$$

where the $\mathbf{r}$ is a vector from a center of mass to the point of applied force and $\mathbf{F}$ is a vector of applied force. According to the relative movement and fact that the UAV behaves like a solid object, the force applied to the mechanism has the same effect as the equal force applied to the point on the x-axis (shown in Figure 5.4).

It is described in the following equation

$$\mathbf{r}' \times \mathbf{F}' = \mathbf{r} \times \mathbf{F}, \tag{5.6}$$

which says that moment applied to the UAV is equal to the force applied through the corresponding arm $\mathbf{r}$. The force applied on the mechanism has y-axis value, thus we can find corresponding arm on the x-axis. That means that the vector is expressed as $\mathbf{r} = [x, 0, 0]^T$ and the force vector equals $\mathbf{F} = [0, y, 0]^T$ as the force is applied along the y-axis so $x$ and $z$ elements are omitted. That simplifies the (5.5) to multiplication

$$\tau = r_x \cdot F_y. \tag{5.7}$$

To successfully rotate the UAV into the desired orientation, the goal is to compensate the created moment. Setting the reference yaw to the negated moment was not ideal and led to the

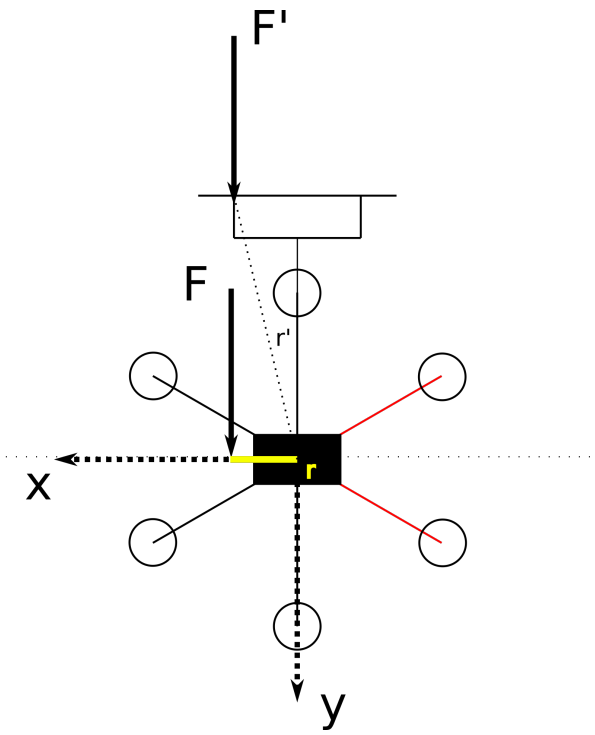



Figure 5.4: Example of the force effect to the moment of the UAV.

unstable state. That is why the reference yaw is the moment scaled down by the factor of 10. This modification stabilized the UAV and still has the needed reaction time for stabilization.

Keeping the UAV at the wall requires having a particular tilt. As the controlled state in this work is only UAV's position and yaw, the tilt is controlled via reference position. Setting the reference further into the wall makes the UAV push more in the direction of the reference and thus generate necessary tilt. The further the reference is, the more the UAV tilts. However, its modification needs to be controlled to prevent uncontrolled tilt which would flip the UAV over. To do so, the modified reference is saturated once it reaches the distance that was found experimentally. This distance stops moving the reference position further, in exchange for making the desired force unreachable.

The control system is designed as a state machine consisting of three main states. The *approaching*, *stabilization* and *detaching* state. The *approaching* state is started via an external command to attach to the wall. It slowly moves the reference position forward in the direction of the mounted mechanism and waits until the force is applied. When it happens, the *stabilization* state starts. This state is an implementation of described methods to stabilize the UAV (see chapter 5). The *stabilization* state runs until an external command to detach from the wall is called. When the *detaching* state starts, it sets the reference position further from the wall. To prevent the mechanism from getting stuck on the wall, the UAV sets reference altitude lower by 20 centimeters until the UAV detaches the wall. Then it returns to its original altitude. The designed state machine of the controlled routine handles the constant reading of the external forces. The pseudocode of the stabilization state is following

**Algorithm 1** The stabilization algorithm that reshapes desired position $\mathbf{r}_D$ based on external forces.

**procedure** GET_REFERENCE
  **Input:**
    $\text{force}_1, \text{force}_2$ ▷ data about force from left (1) sensor and right (2) sensor
    $\mathbf{r}_D$ ▷ vector of desired position
    desired_force ▷ desired force set by user
  **Output:**
    $\mathbf{r}_R$ ▷ reference position vector
    $\phi_R$ ▷ reference yaw orientation
  $\text{force} \leftarrow \text{force}_1 + \text{force}_2$
  $d_x \leftarrow \dfrac{\text{force}_1 \cdot joints\_dist}{\text{force}}$
  $\text{relative}_\text{y} \leftarrow \dfrac{force - desired\_force}{K_{dy}}$ ▷ desired movement in the body frame
  $\text{relative}_\text{yaw} \leftarrow -d_x \cdot \text{force} \cdot 0.1$
  $[\mathbf{r}_R, \phi_R] \leftarrow \text{move\_relative}(\text{relative}_\text{x}, \text{relative}_\text{y}, \text{relative}_\text{z}, \text{relative}_\text{yaw}, \mathbf{r_D})$
**end procedure**

**procedure** MOVE_RELATIVE
  **Input:**
    **relative** ▷ vector [x, y, z, yaw]
    $\mathbf{r}_D$
    **odometry** ▷ estimated position and orientation of the UAV
  **Output:**
    $\mathbf{r}_R$
    $\phi_R$
  $\mathbf{R} \leftarrow rotationMatrix(odometry_{yaw})$
  $\mathbf{rot} \leftarrow \mathbf{R} \cdot [relative_x, relative_y]^T$ ▷ transformation into world frame
  $\text{r}_{Rx} = rot(1) + r_{Dx}$
  $\text{r}_{Ry} = rot(2) + r_{Dy}$
  $\text{r}_{Rz} = relative_z + r_{Dz}$
  $\phi_R = relative_{yaw} + r_{Dyaw}$
**end procedure**

Algorithm 1 describes core of the control system. Function *Get_Reference* processes the data from the force sensors and calculates the desired movement from the body frame. Variable *desired force* works as an offset to the applied force. Without the offset, the UAV would be on edge between the wall and free flight and the constant touch would not be assured. Function *Move_Relative* uses the UAV's odometry to convert relative movement into the world coordinates. Odometry is the estimated position and orientation coming from sensors on the UAV. From the heading ($odometry_{yaw}$) is computed the rotational matrix $\mathbf{R}$ which is then multiplied with the vector of the relative motion. The result is added to the desired position. Relative yaw and altitude ($relative_z$) of the UAV are directly added to the desired altitude and orientation.

# 6 Verification in simulation

Simulation is key in the process of successfully implementing work in real-world experiments. Getting information about the behavior of the designed mechanism or control system is crucial. MRS group provided their UAVs in the Gazebo simulator, which allowed us to concentrate mainly on making the model of the designed mechanism. As the platform that should carry the mechanism is not built yet, we use the F550 platform instead. The way how the mechanism is mounted is not important as long as it fulfills similar conditions and effect acting on the UAV is the same. The final look of the mechanism in the simulator is in Figure 6.1.

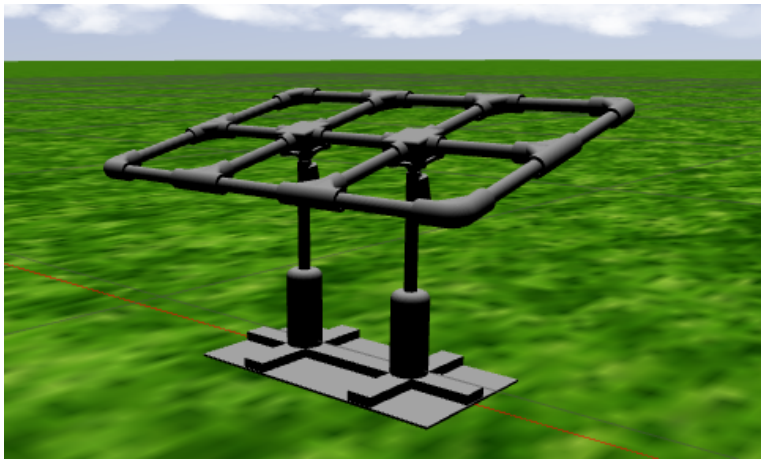
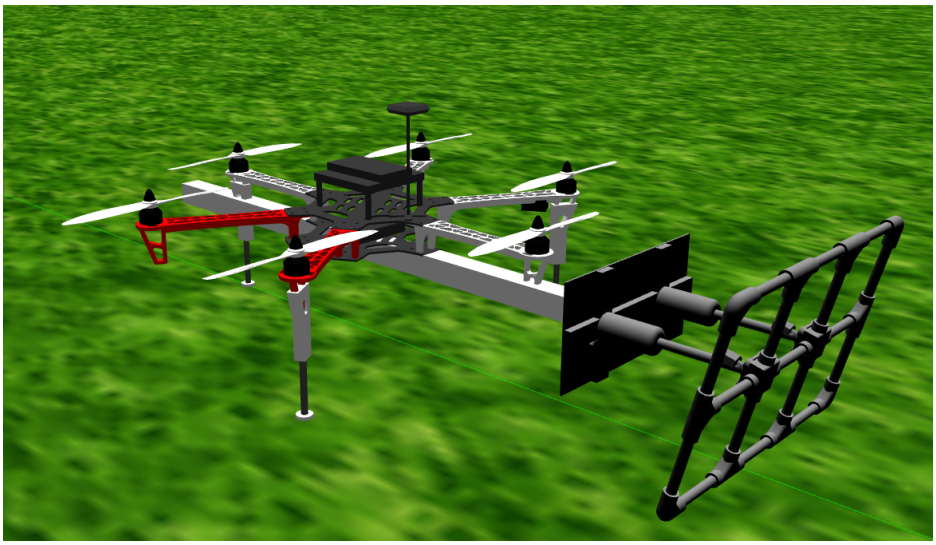


Figure 6.1: Model of the mechanism

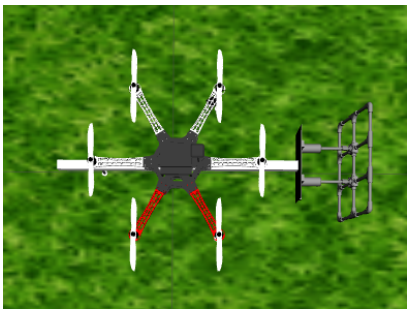
The mechanism was mounted to the platform to one side of a beam that was mounted under the platform. The beam itself has no weight and provides only a mounting point to the mechanism. A counterbalance is mounted opposite than mechanism to shift the resulting center of gravity under the geometric center of the UAV. Figure 6.2 shows how is the mechanism mounted on the UAV.

## 6.1 Mechanism simulation

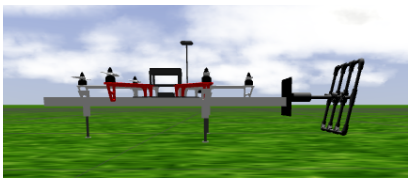
Gazebo simulator does not include a direct simulation of a spring. However, it provides an option to simulate a prismatic joint which allows it to behave like a spring. It is done by setting the bounds of the movement and stiffness to the joint. Another feature of Gazebo are sensors. It can simulate either distance and force sensors. Even though the real mechanism is equipped with force sensors, we decided to use the distance sensor with the spring to evaluate



(a) Model of the UAV with the mechanism mounted.



(b) Top view



(c) Side view of the model

Figure 6.2: Pictures of the final model mounted on the UAV. (a) shows the UAV from general view, (b) and (c) shows details from top and side view.

applied force in the simulation. Its implementation is much simpler and conditions in the simulation do not affect the measuring as the real world conditions would, for example, the spring constant uncertainty is not modeled. If we would like to use the force sensor, the implementation would be difficult. The spring is not physically in the place, it is only moving joint. Thus there would be no contact to the force sensor.

Data from sensors are published to ROS. A ROS node was created to calculate applied force from the raw data. The node compares the distances to the predefined threshold which leads to compression ratio and after this value is multiplied with the stiffness of the spring. The stiffness of the springs does not match the stiffness set in the joint definition, so its value was experimentally set. To verify the model's precision we conducted several experiments (Figure 6.3) applying the force to the mechanism not mounted on the UAV.

The figure shows the response of the model to the applied force. It shows the response to the 50, 100, 150 and 200 N. The model's response equals approximately the applied force. Response to the 50 N reaches lower values, between 40 - 45 N and the response to the 200 N shows higher values, up to 230 N. The response of the model is not consistent. However, it is satisfying for our purpose.

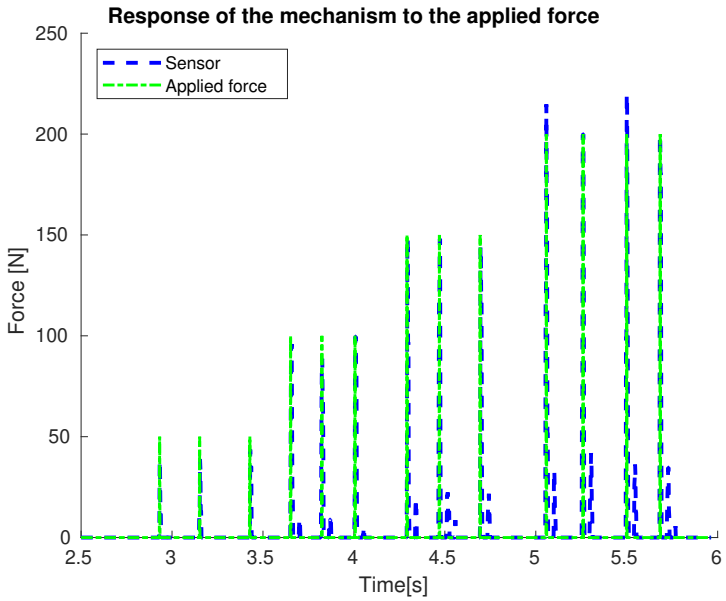



Figure 6.3: Experiments of the model's response to the applied force.

## 6.2 UAV simulation

The UAV equipped with the designed mechanism was undergone to experiments testing its behavior to interaction with the wall. The model was tested to the ability of a controlled approach if it can continuously approach with optimal speed. Another section focuses on the behavior while attached to the wall, its stability and reaction while approaching from the non-ideal state. The last section tested the behavior of the model while detaching the wall.

Figure 6.4 depicts the UAV stabilized on the wall when the force offset was set to 1 N and the angle between the wall and the mechanism was zero. Constant interaction with the wall is ensured. Figure 6.5 shows the UAV while approaching the wall. The UAV keeps the speed up to 0.2 $ms^{-1}$. The detaching process (Figure 6.5b) is controlled and the UAV then hovers approximately 2 meters from the wall.

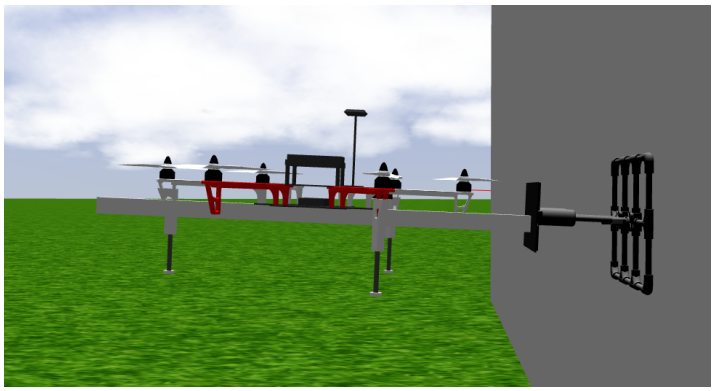
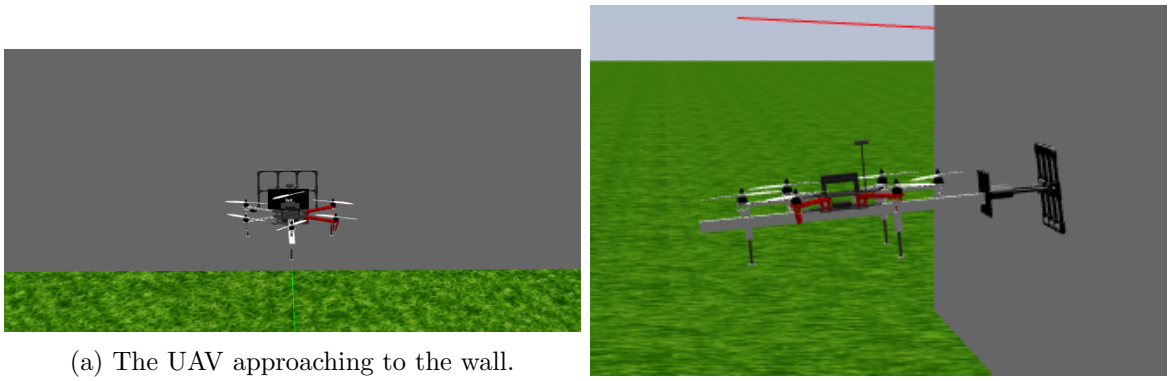


Figure 6.4: Snapshot from the simulation when the UAV is stabilized on the wall.



(a) The UAV approaching to the wall.

(b) The UAV detaching from the wall.

Figure 6.5: Snapshots from simulation. (a) the UAV approaches the wall from ideal angle and (b) the UAV detaches from the wall.

Figure 6.6 shows a comparison of the UAV's estimated position and the reference position from the top view while approaching in the right angle. The positions are displayed against the UAV's center of mass. Thus the contact point is not detailed and the wall is placed further from displayed trajectories. When the UAV touches the wall, the estimated position stops but the reference position moves further to the wall to produce desired force and constant touch. The yaw orientation of the UAV is in this case constant. Figure 6.7 shows the same case as the Figure 6.6 but in the 3D space and without the wall highlighted for simplification. The reference position is recorded from the UAV's takeoff. Both approaching and detaching states overlaps each other. However, the process of detaching is more visible. Firstly, the reference position lowers the altitude and then returns to the previous reference.

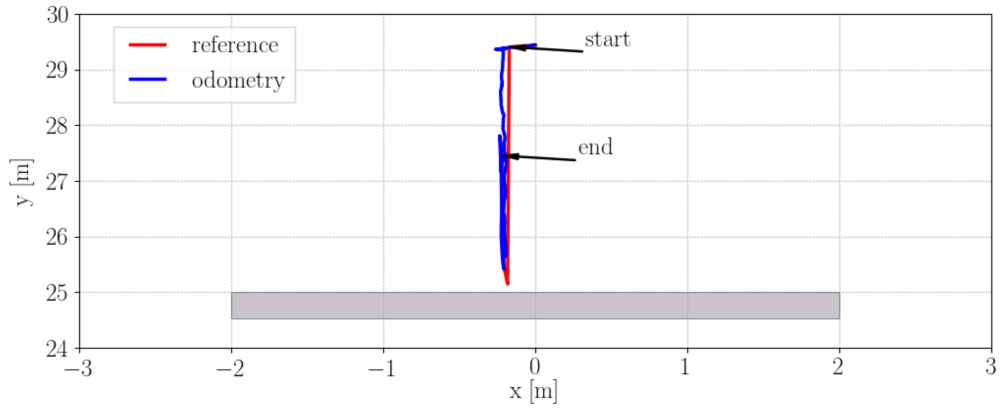



Figure 6.6: Comparison of the estimated position and reference position from the top view. The wall is indicated by the gray rectangle.

The UAV is able to stabilize itself when the angle between the mechanism and the wall is within 35°. Figure 6.8 shows the UAV, (a) before the first touch with the wall and then (b) stabilized. It describes the case when the angle is 17.2°. Figure 6.9 shows the behavior

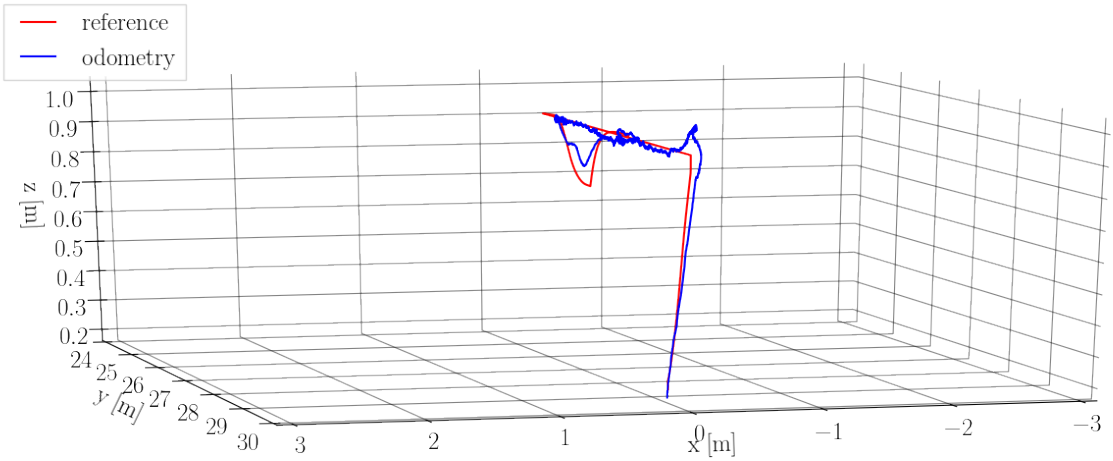
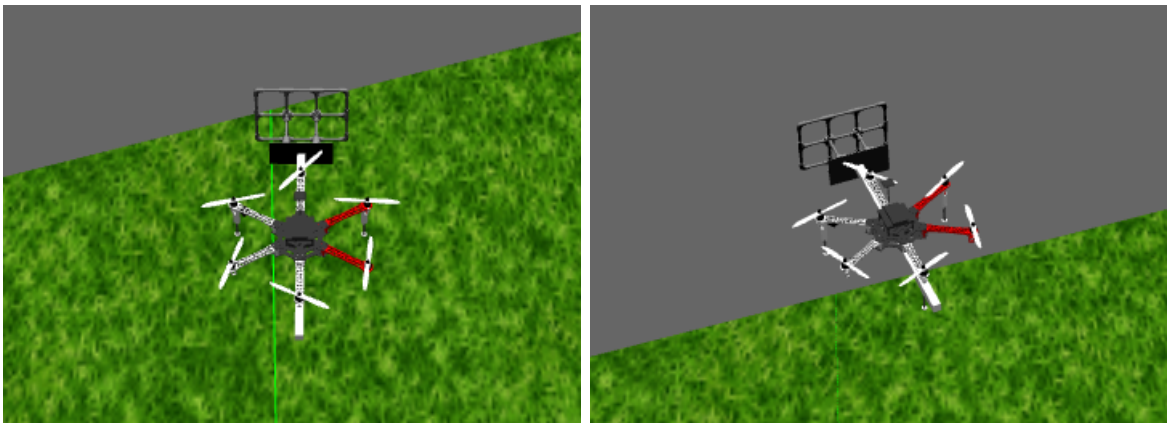



Figure 6.7: Comparison of the positions in 3D space when the wall is approached in right angle. The starting point is the place of the takeoff.

when the angle is 34°. Snapshots from the detaching process are not shown as the process is the same as in Figure 6.5b. The UAV also reaches the same state as in Figure 6.4. Thus the Figures 6.8 and 6.9 depict only the ability of successful rotation to stabilized state.

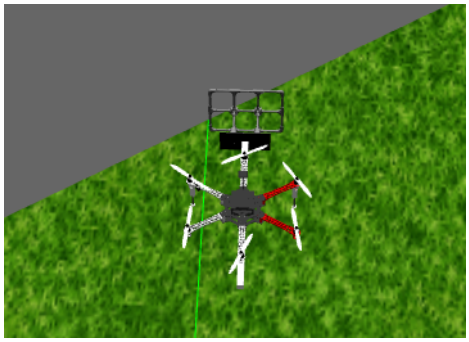


(a) Top view while approaching the wall.

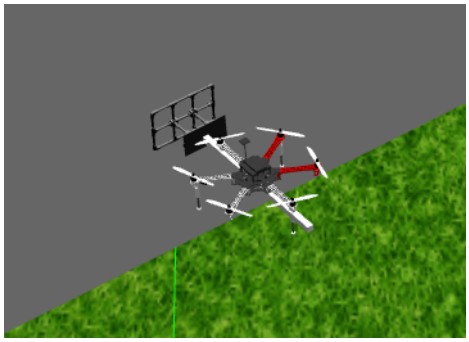
(b) Following stabilization on the wall.

Figure 6.8: Snapshots from simulation when the angle between the wall and the mechanism is 17.2 °.

Comparison of the reference and estimated position for the angle equal to 17.2° is in Figure 6.10 and for the angle equal to 34.4° is in Figure 6.11. Figures show that the UAV is able to follow the reference position even when the interaction with the wall is performed. The UAV successfully approaches the wall and stabilizes itself with necessary modification of the reference position. Then it detaches and hovers two meters from the wall. The higher the angle between the mechanism and the wall is, the more steps take to the UAV to accomplish the desired yaw. Videos from the simulation can be found at `http://mrs.felk.cvut.cz/theses/smrcka2019`.

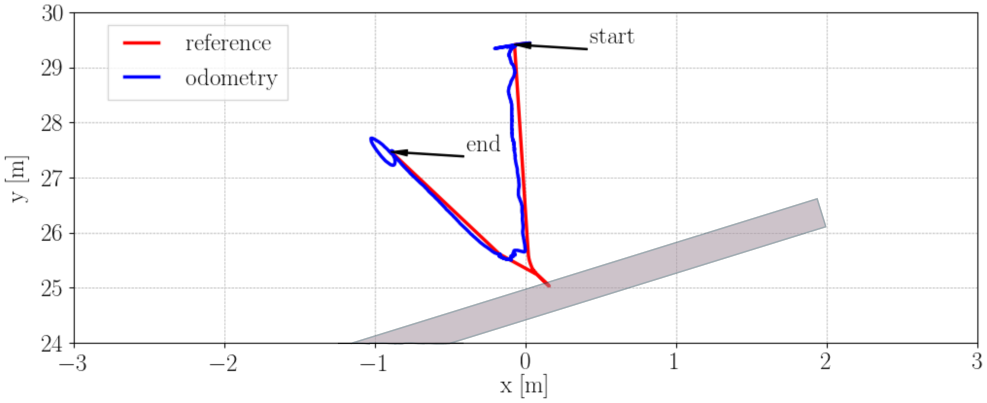


(a) Top view while approaching the wall.

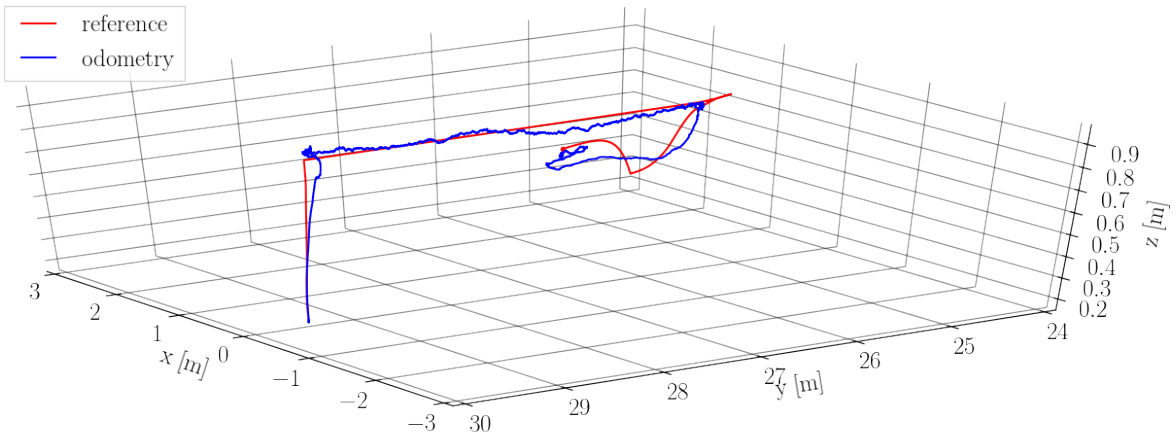


(b) Following stabilization on the wall.

Figure 6.9: Snapshots from simulation when the angle between the wall and the mechanism is 34.4 °.

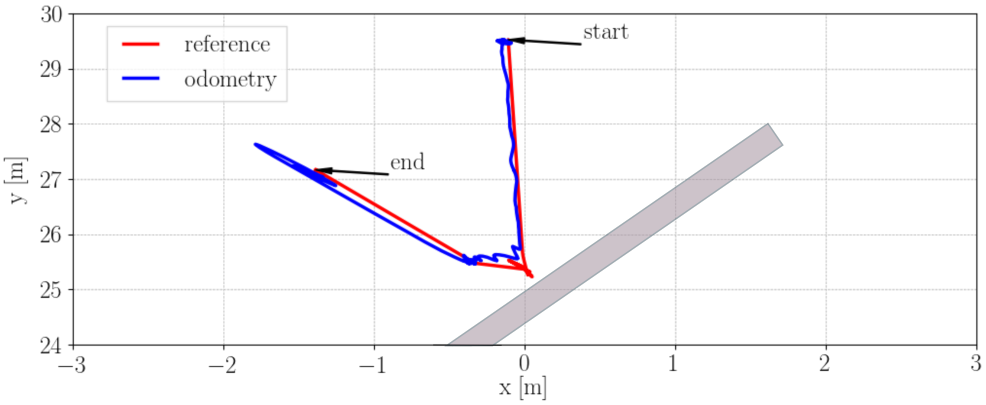



(a) Comparison from the top view.

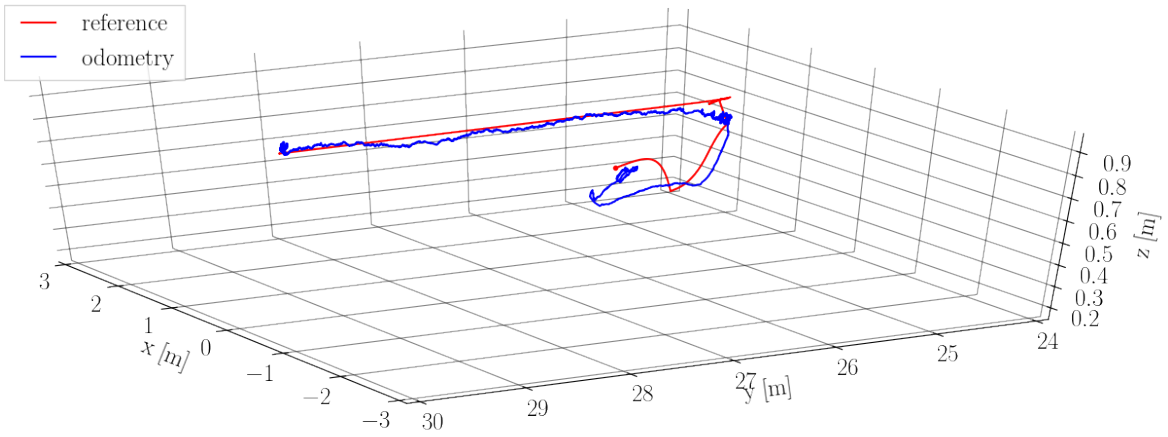



(b) Comparison in the 3D space.

Figure 6.10: Comparison of positions when the angle is 17.2 °.




(a) Comparison from the top view.



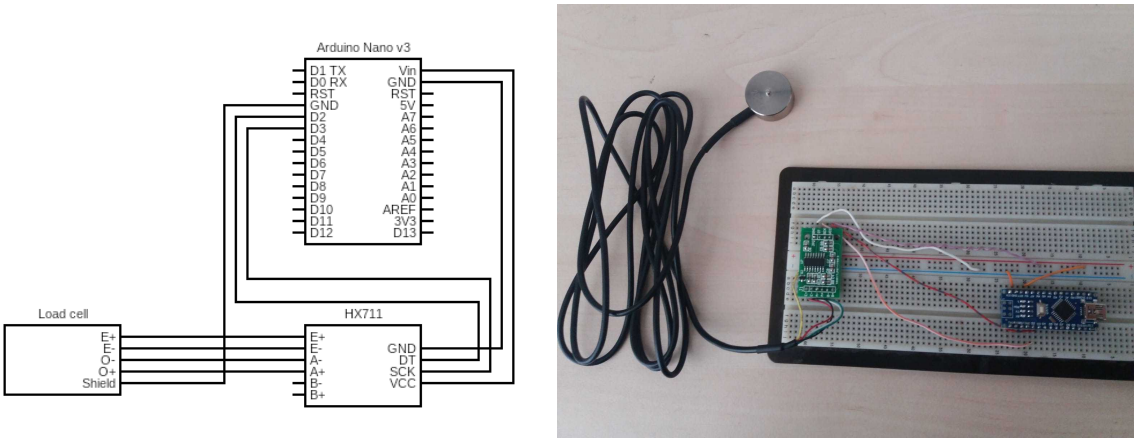

(b) View in the 3D space.

Figure 6.11: Comparison of positions when the angle is 34.4 °.

# 7 Hardware verification

This chapter describes the implementation of the hardware and its testing. It presents the implementation of force sensors and the assembly of the mechanism.

Force sensors used in this work are button load cells (Figure 4.3a) that are built to tolerate up to 200 kilograms (2000 Newtons). Interfacing the sensors with the onboard PC of the UAV is provided through an Arduino Nano micro-controller. HX711 amplifiers were used to amplify the change in resistance and sample it using integrated Analog-Digital converter before it is sent to the Arduino. To connect the Arduino with the onboard PC running with ROS, ROS provides a library called Rosserial-Arduino. This library allows connecting the Arduino board with ROS directly through a serial link. Figure 7.1a shows wiring diagram and Figure 7.1b shows a photo of the sensor wired to the Arduino board.




(a) The wiring diagram

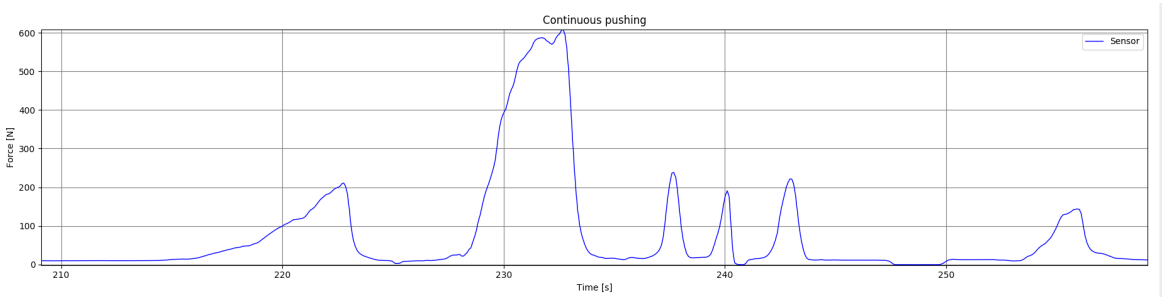
(b) Hardware wiring

Figure 7.1: Interface of the sensors with the Arduino.

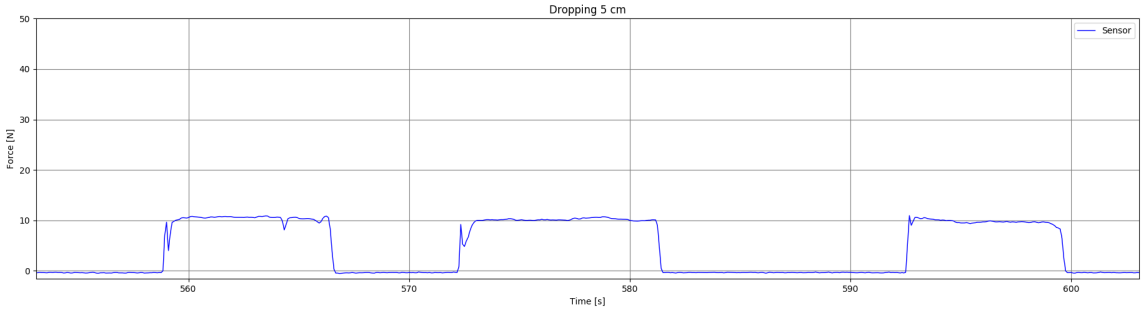
Several experiments were conducted to show the functionality of the force sensor in multiple challenges such as continuous pushing of 1 kilogram object (Figure 7.2a), dropping of same object on the sensor from height of 5 centimeters (Figure 7.2b) and pushing sensor itself on to a wall (Figure 7.2c).

Both experiments of continuous pushing of an object and pushing the sensor to a wall showed that the sensor is capable of recognizing fast continuous changes of applying force. The sensor is able to measure heavyweights, as in Figure 7.2a. It measured 600 N, which means 60 kg of weight. It is not likely that UAV could generate higher force in controlled flight.

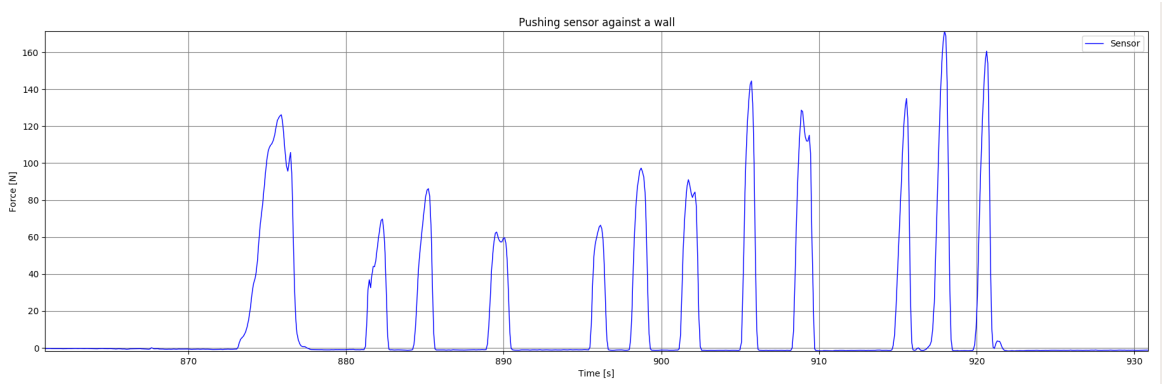
The experiment of dropping a 1-kilogram object on the sensor showed that the sensor is not capable of recognizing step changes of force. Figure 7.2b shows three drops, and each did




(a) Continuous pushing of 1 kilogram object to the force sensor.




(b) Dropping 1 kilogram object to the force sensor from height of 5 cm.




(c) Pushing the force sensor against a wall with a different speed and force.

Figure 7.2: Experiments with the force sensor.

not show the initial force impact. The force then reached the value that the object generates in steady state.

This behavior could be improved by increasing the frequency of transmission which is currently 10 Hz. It means that there are ten new samples every second and that might be not enough to such a quick change. However, the button load cell will not be able to completely reconstruct the whole signal as the principle of the piezoresistivity is not capable of recognizing fast moment impacts. We expect primarily continuous impacts and increasing the frequency to 100 Hz is satisfying to our purpose. Model of the mechanism was successfully mounted. Its picture is in Figure 7.3. Figure 7.4 shows the base part in detail. The base part includes the button load cell in the bottom.

Figure 7.3: The model of the mechanism mounted.

(a) Base part with connection shaft.

(b) Detail of the force sensor implemented into mechanism.

Figure 7.4: Details of the base part of the mechanism.

# 8 Conclusion

This thesis presented a solution to the challenge of the controlled interaction of an Unmanned Aerial Vehicle with the wall using a compliant mechanism. The thesis dealt with the implementation of the admittance control to the MRS group's control pipeline. The mechanism was designed based on conditions described in Chapter 4 and included two force sensors to measure acting force while attached to the wall.

The force measurement was studied in Chapter 3. We presented multiple solutions to this problem and considered the possible errors that might be crucial in the real world experiments. After considering the advantages and disadvantages of studied principles and options, we decided to use the piezoresistive button load cells.

Due to lack of the final platform that should carry the mechanism, this work is based on the F550 hexacopter instead. The mechanism was mounted on the side of the UAV and provided reliable response while attached to the wall. Experiments showed that the mechanism mounted on the UAV provides the right information about the applying force.

Results were presented through several experiments in simulation. The UAV, equipped with the designed mechanism and controlled via the implemented control approach, is capable of controlled interaction with the wall. It stabilizes itself even if the wall is not approached in the ideal angle. However, the experiments also showed design imperfections. The initial touch with the wall is not ideal.

## 8.1 Future work

Work presented in this work provides good basics into the following research of the wall interaction. Future work will be to integrate the designed mechanism on the real UAV platform and test its behavior in real-world experiments. The behavior at the first touch with the wall could be improved. Data from the lidar sensor or a new distance sensor could be used to get information about the wall distance which would allow the possible reduction of the initial impact on the UAV. The mechanism could be equipped with an inertial measurement unit to obtain information about the tilt from the end effector. The force estimation could be improved with a filter to reduce oscillations in the measurements for better control.

# Bibliography


[1] T. Báča, P. Štěpán, V. Spurný, D. Hert, R. Pěnička, M. Saska, J. Thomas, G. Loianno, and V. Kumar, "Autonomous landing on a moving vehicle with an unmanned aerial vehicle," *Journal of Field Robotics*, 2018.

[2] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[3] L. Joseph, *Mastering ROS for robotics programming : design, build, and simulate complex robots using Robot Operating System*. Birmingham, UK: Packt Publishing, 2018.

[4] "Intro to the robot operating system," https://robohub.org/ros-101-intro-to-the-robot-operating-system/, online, February 2019.

[5] "Proximity sensors," www.machinedesign.com/, online; February 2019.

[6] G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff, and A. Franchi, "Turning a near-hovering controlled quadrotor into a 3d force effector," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6278–6284.

[7] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *2013 European Control Conference (ECC)*, July 2013, pp. 1805–1810.

[8] G. Loianno, V. Spurný, T. Báča, J. Thomas, D. Thakur, D. Heřt, R. Pěnička, T. Krajník, A. Zhou, A. Cho, M. Saska, and V. Kumar, "Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments," *IEEE Robotics and Automation Letters*, 2018.

[9] J. M. O'Kane, *A Gentle Introduction to ROS*. Independently published, Oct. 2013.

[10] "Robot operating system," http://wiki.ros.org/ROS/Introduction, online, February 2019.

[11] A. M. Enrique Fernández, Luis Sánchez Crespo and A. Martinez, *Learning ROS for Robotics Programming - second edition*, P. Publishing, Ed. Packt Publishing, 2015.

[12] J. Lentin, *Learning robotics using Python : design, simulate, program, and prototype an interactive autonomous mobile robot from scratch with the help of Python, ROS, and Open-CV*. Birmingham, UK: Packt Publishing, 2015.

[13] S. K. F. Ansel C. Ugural, *Advanced Mechanics of Materials and Applied Elasticity*. Prentice Hall, 2012.

[14] R. B. Northrop, *Introduction to Instrumentation and Measurements*. Taylor and Francis, 2005.

[15] "What is a load cell," www.loadstarsensors.com/what-is-a-load-cell.html, online; February 2019.

[16] D. Placko, *Fundamentals of Instrumentation and Measurement*. ISTE, 2007.

[17] "Inductive sensors," www.electrical-engineering-portal.com/, online, February 2019.

[18] "Vanel, spring manufacture," https://www.vanel.com/, online, February 2019.

# Appendices

# A CD Content

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

| **Directory name** | **Description** |
|---|---|
| thesis | the thesis in pdf format |
| src/thesis | latex source code |
| src/tracker | sources for the tracker |
| src/simulation | sources for the simulation |
| src/wall | sources for the mechanism model |
| src/3D_print | stl files for 3D print |
| src/load_cell | sources for Arduino Nano |

Table A.1: CD Content

# B List of abbreviations

In Table B.1 are listed abbreviations used in this thesis.

| Abbreviation | Meaning |
|---|---|
| **API** | application programming interface |
| **ROS** | robot-operating system |
| **ToF** | time of flight |
| **MRS** | multi-robot systems |
| **MPC** | model predictive control |
| **CTU** | Czech Technical University |
| **UAV** | unmanned aerial vehicle |

Table B.1: Lists of abbreviations

# C Technical drawings of model components

Figures C.1, C.2, C.3 show technical drawings of the designed parts.

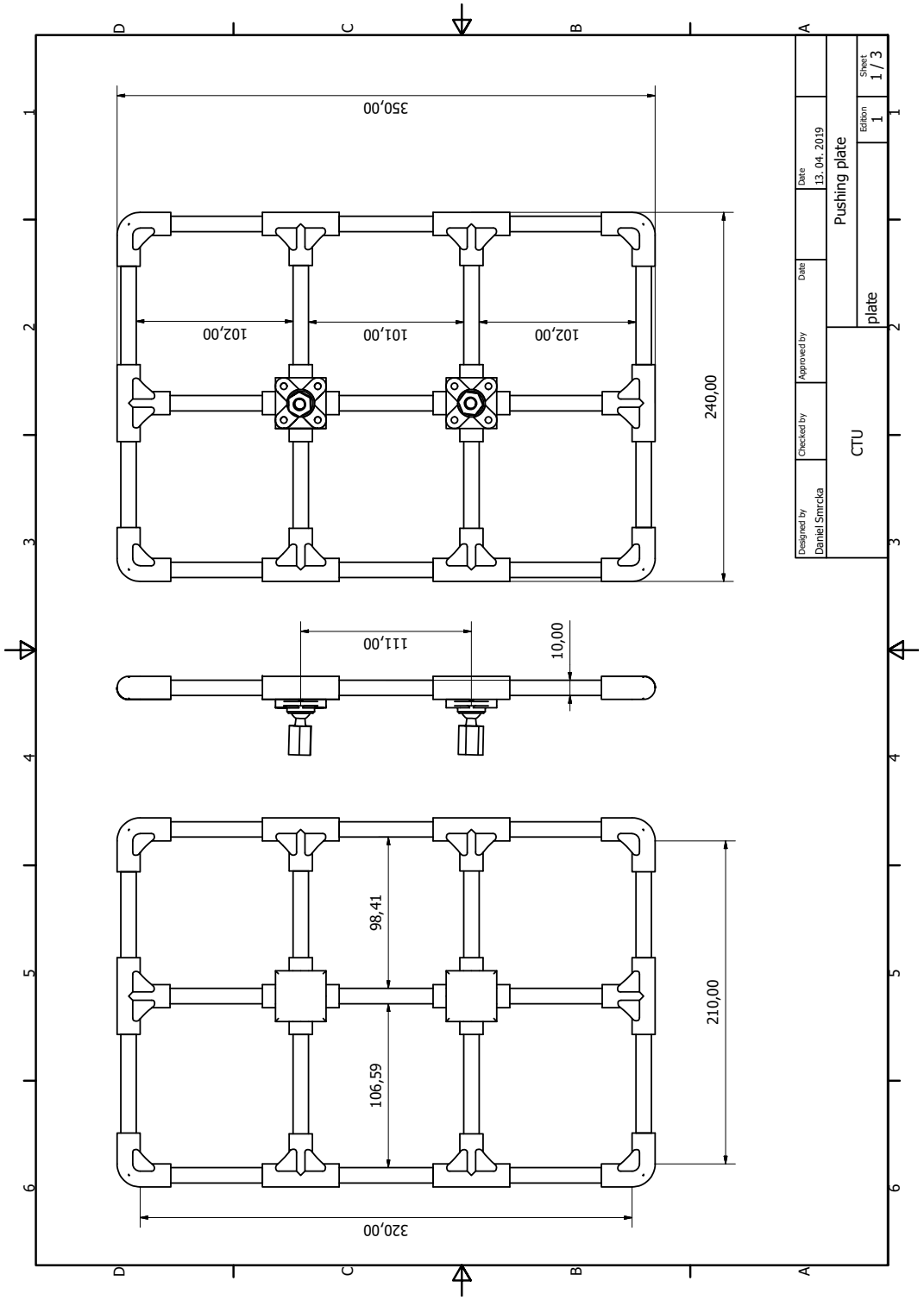



Figure C.1: Drawing of the pushing plate model. Dimensions are presented in millimeters.

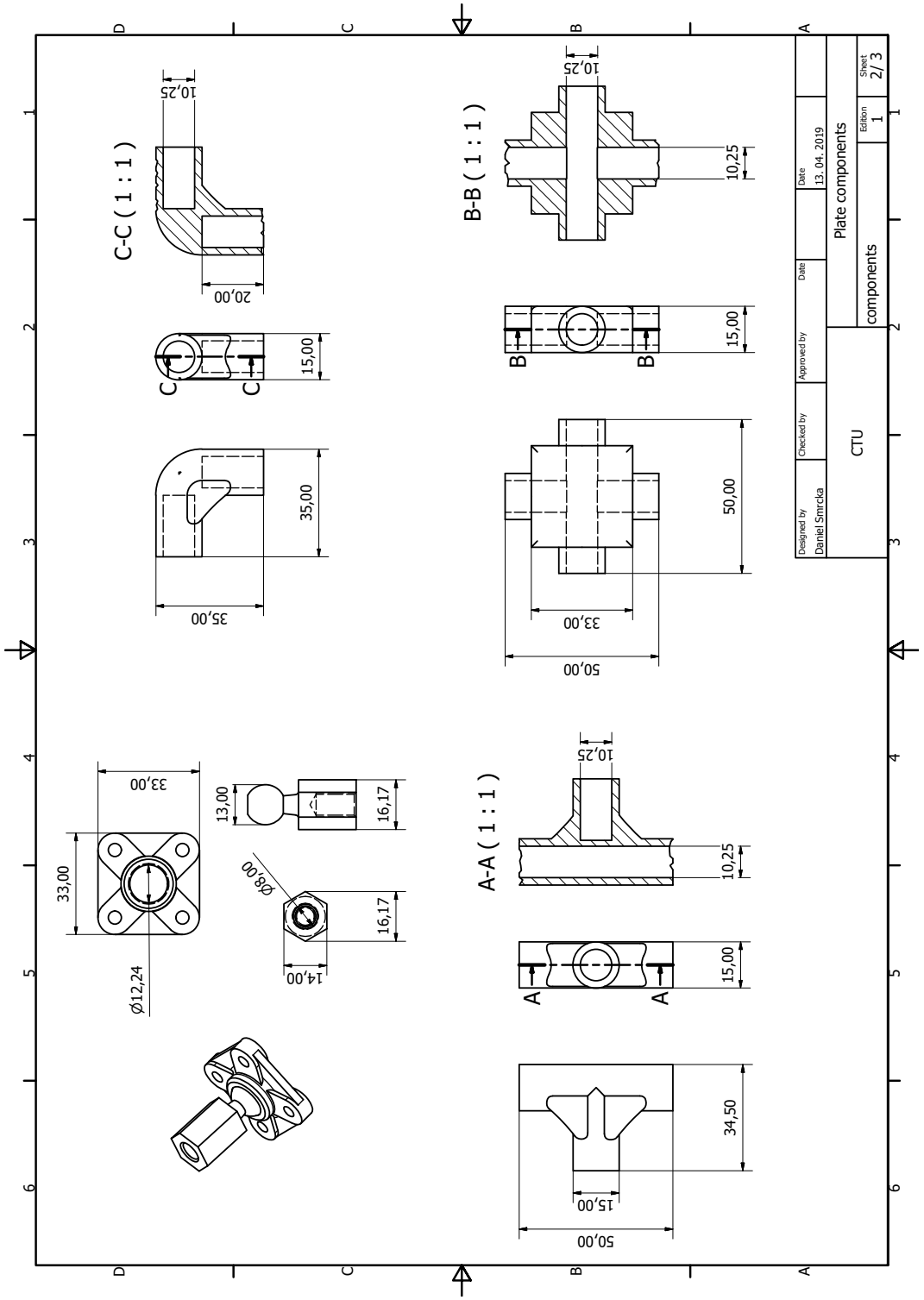



Figure C.2: Components used in model. Dimensions are presented in millimeters.

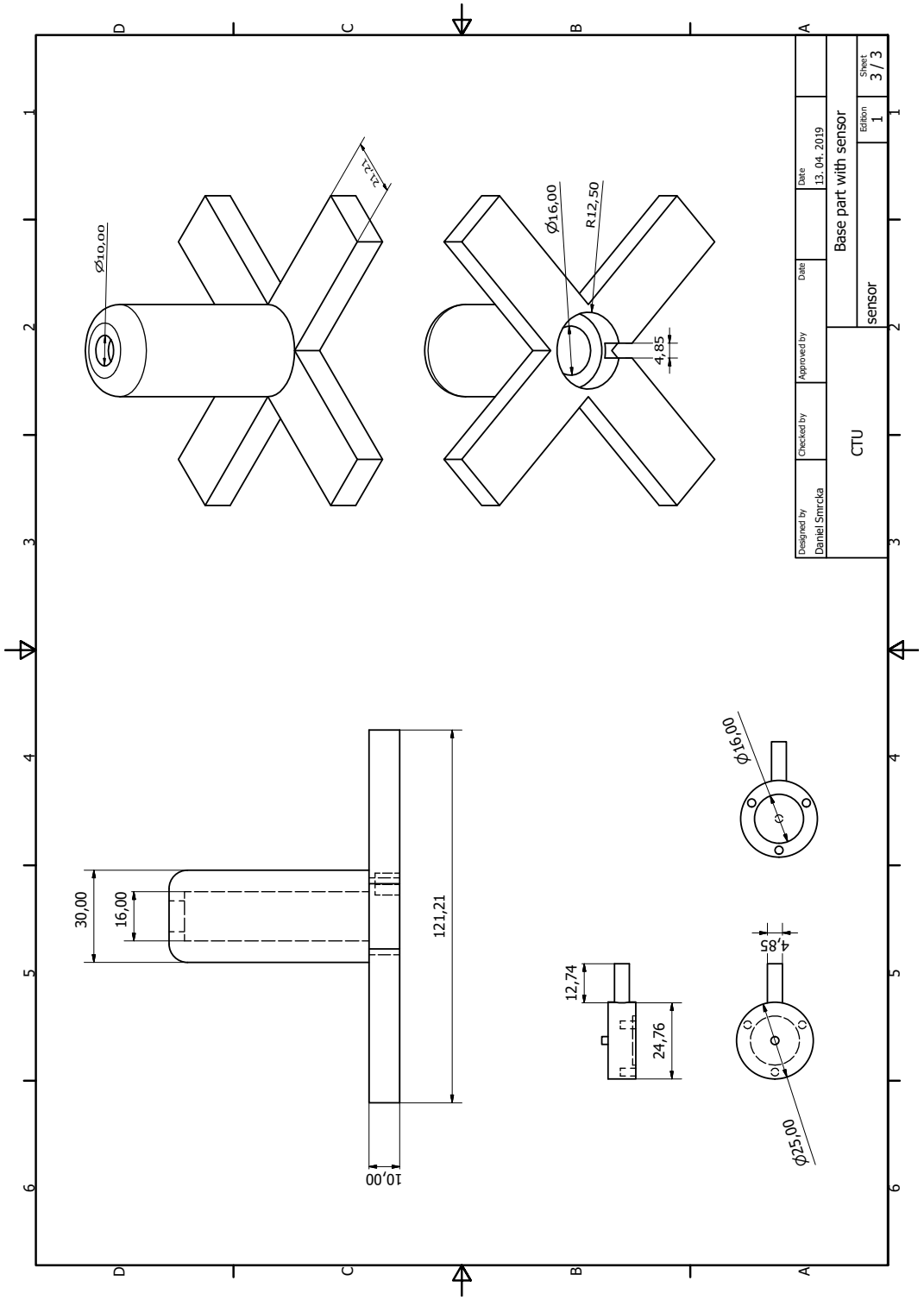



Figure C.3: Base part with the sensor. Dimensions are presented in millimeters.