ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

BAKALÁŘSKÁ PRÁCE

2024

JAN RYCHTERA

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Rychtera**	Jméno: **Jan**	Osobní číslo: **509151**
Fakulta/ústav:	**Fakulta strojní**		
Zadávající katedra/ústav:	**Ústav přístrojové a řídící techniky**		
Studijní program:	**Teoretický základ strojního inženýrství**		
Studijní obor:	**bez oboru**		

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Implementace algoritmu tlumení kyvu břemene v autopilotu PX4

Název bakalářské práce anglicky:

Implementation of the load swing damping algorithm in the PX4 autopilot

Pokyny pro vypracování:

1) Seznamte se s open-source autopilotem PX4 v kombinaci s HW PixHawk a možnostmi napojení tohoto řídicího systému na prostředí MATLAB/Simulink.
2) Seznamte se s algoritmem prevence/tlumení kyvu břemene zavěšeného na dronu založeném na principu tvarování signálu.
3) Ověřte možnosti implementace uživatelských kódů v rámci autopilota PX4.
4) Implementujte výše zmíněný algoritmus v prostředí MATLAB/Simulink tak, aby se vykonával v řídicím obvodu dronu.
5) Simulačně ověřte funkčnost zvolené realizace.

Seznam doporučené literatury:

VYHLÍDAL, Tomáš; KUČERA, Vladimír; HROMČÍK, Martin. Signal shaper with a distributed delay: Spectral analysis and design. Automatica, 2013, 49.11: 3484-3489.
VYHLÍDAL, Tomáš; HROMČÍK, Martin. Parameterization of input shapers with delays of various distribution. Automatica, 2015, 59: 256-263.
KUŘE, Matěj; BUŠEK, Jaroslav; VYHLÍDAL, Tomáš. Swing compensation of a payload suspended to a planar copter. In: 2021 23rd International Conference on Process Control (PC). IEEE, 2021. p. 7-12.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jaroslav Bušek, Ph.D. U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.04.2024** Termín odevzdání bakalářské práce: **31.05.2024**

Platnost zadání bakalářské práce: ______

[illegible]	[illegible]	v.z. [illegible]
Ing. Jaroslav Bušek, Ph.D.	prof. Ing. Tomáš Vyhlídal, Ph.D.	doc. Ing. Miroslav Španiel, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

24.4.2024	[illegible]
Datum převzetí zadání	Podpis studenta

Acknowledgment

I would like to thank thesis supervisor Ing. Jaroslav Bušek Ph.D. for his leadership throughout this work, his willingness to help and his human approach.

I would also like to thank the institution of České vysoké učení technické for the privilege that is education.

At last, I would like to thank my family for their mental and material support throughout my studies.

Declaration

I declare, that the submitted work has been created independently and I have stated all used literature.

In Česká Lípa 11.05.2024

Abstract

This thesis covers up a method of implementation of zero vibration shaper algorithm for a PX4 autopilot based drone, with MATLAB/Simulink software, both of them being chosen prior to this work by the main supervisor, Ing. Bušek Ph.D. The main result should be a simulation testing and concept proofing usage of Simulink software and PX4 based board.

The work covers up a brief introduction to the used autopilot and software as well as a brief introduction to drones, to have a basic understanding of what may be controlled using zero vibration shaper, and an introduction to mentioned zero vibration shaper.

An overview of possibilities of code implementation of PX4 autopilot in MATLAB Simulink is covered up next, with the realization that used UAV Toolbox offers many possibilities and an introduction to some of them.

In the latest stage a successful control that output voltage can be obtained from the used Pixhawk board controlled using Simulink, as well as a model of implemented ZV shaper that is ran on Pixhawk board and writing to and reading from messaging process of PX4 in Simulink, that should carry out the movements of the drone.

Tato bakalářská práce se zabývá metodou implementace algoritmu zero vibration shaperu pro dron na bázi autopilota PX4 pomocí softwaru MATLAB/Simulink, přičemž oba tyto softwary byly vybrány před touto prací vedoucím práce, Ing. Buškem Ph.D. Hlavním výsledkem by mělo být simulační testování a ověření konceptu užití softwaru Simulink a řídicí desky založené na PX4 autopilotu.

Práce zahrnuje stručné seznámení s použitým autopilotem a softwarem a také stručný úvod do dronů, kvůli získání základní představy o tom, co lze pomocí zero vibration shaperu řídit, a úvod do zmíněného zero vibration shaperu.

Dále je uveden stručný přehled možností implementace kódu autopilota PX4 v prostředí MATLAB Simulink s pozorování, že použitý UAV Toolbox nabízí mnoho možností, ze kterých jsou některé představeny.

V poslední fázi bylo úspěšně ověřeno, že lze získat výstupní napětí z použité desky Pixhawk řízené pomocí Simulinku, a poté je proveden model implementovaného ZV tvarovače, který je spuštěn a běží na desce Pixhawk a zapisuje se do a čte z něj proces zpráv PX4 v Simulinku, který by měl řídit pohyby dronu.

Contents

Pictures

Introduction

This thesis is set around trying to implement a drone control using MATLAB Simulink software with the goal of proving a concept, whether a chosen hardware, being a Pixhawk board, and software are eligible for usage of a drone control with weight suspended under its body. Both software and hardware used for the purposes of the thesis had been chosen prior to authors involvement in this work. Later in the work the main focus is placed on proving a concept of realization of control using the Simulink software while the hardware is being used only to run the generated code from Simulink. The software used may have been chosen since ČVUT already has license for using Simulink and should be user-friendly. Within the confines of this work, only simulation testing will be undertaken, with hopes of confirming that the chosen method is not a dead-end and continuation of work using Simulink may bring results in the future however no practical test with a drone will be undertaken.

1 About drones

In the first part of the theoretical portion small research of information about drones will be covered, to get a very basic understanding of drones and how they operate and function, as the goal of the work would be to control one. Therefore it is for the better to have a basic overall picture about the whole problematics, rather than having none.

1.1 Flying UAVs categorizing

To gain some basic understanding about drones and the movements that can be regulated, a following section covers the basics about drones.

A drone is a common name often used for unmanned vehicles, which includes both unmanned aerial vehicles (UAV) as well as unmanned ground vehicles (UGV). Both types of the devices have the same idea of having devices capable of transporting payloads and performing tasks utilizing remote human interventions.

These vehicles could find and do find their usage in areas, in which alleviate risks of endangering human life, such as during space missions or military usage, as well as they could reduce the need of people in areas such as delivering packages, therefore reducing the overall cost. This concept is currently under the development of companies such as amazon. For now, the most commercially used types of drones are for aerial photography and filmmaking, since the drones significantly lower the cost of the whole production, eliminating the need for usage of helicopters or planes.

During the remainder of this work, word drone will be used as a synonym for unmanned aircraft vehicles, despite the UAV Toolbox having an option of designing path followers.

(UAV) Drones are manufactured in many variants, mostly categorized by amount and layout of rotors. Examples of drone categories differentiated by their bodies include multi-rotor drones, fixed-wing drones, single-rotor drones and fixed-wing hybrids [1] [2] [3]. Understanding that these drones vary in their build and their flight method, it is good to have basic idea, that differently built drones vary in their flight mechanics, for situations where is a need to control their movements from a custom created control models (in sense of controlling directly their rotors) , which is trying to be avoided in this work.

The most notoriously known type of drones and most importantly the type of drone this bachelor's thesis is intended for is multi-rotor drones [1] [2]. These drones are usually built upon a concept of multiple rotors placed vertically on the frame of the UAV [1] [2]. From there comes another way to differentiate UAVs. The way multi-rotor drones can be categorized is by the number of rotors connected to the frame of the drones, such as tricopters (three rotors), quadcopters (with four rotors), hexacopter (six rotors), octacopters etc.

For optimal efficiency of the flying devices, high requirements are placed on the bodies of the drones, mainly the rigidness of the construction and low weight of the body, therefore alleviating the amount of force needed to be produced by rotors of the drone and sturdiness of the whole device.

1.2 Basic drone movements

In the following section a very simplified model of a drone will be described as well as describing some basic movements of the whole vehicle. This model can be described in more detail by more complicated mechanical equations, but as it is not the main focus of this thesis, it's not going to be covered more.

Objects in the 3-dimensional space do usually have 3 degrees of freedom, them being translation in three directions perpendicular to each other in the directions of X, Y, Z axes and rotations around these axes. The rotors of the UAV generate forces and rotational moments. Adjusting the forces and moments produced, impacts the total velocity of the vehicle, and leads to complex movement of drones, and gives the ability to perform desired tasks and movements [1] [2].

Four rotor drone is underactuated with only four actuated degrees of freedom, therefore at least two pairs of movement are bound together. To describe the movement of a multi-rotor drone, four basic movements are being used, to be exact [2]:

- Vertical lift
- Pitch
- Yaw
- Roll

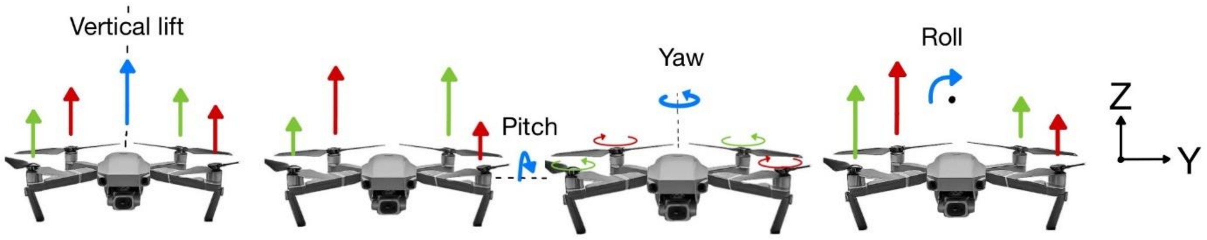

Figure 1.2.1. – Movements illustration

Vertical lift or a throttle is a type of movement that is parallel to axis Z of the coordinate system placed located in the center of the gravity. Under the presumption that the vehicle is balanced in the way, that every propulsor is in the same distance from the center of the gravity, it's caused by an equity of forces produced by the engines. There is no rotation around any axis.

Pitch is created by inequity of force sum of two pairs of rotors. The result is an angular rotation around the Y axis and is bound with the motion in the direction of the axis X. The pitch is usually done by lowering of velocity of one pair of motors and increase in the velocity of the other pair, which is in the mirror position by the axis of rotation.

Roll is created by inequity of force sum of two pairs of rotors [1]. The result is an angular rotation around the X axis and is bound with the motion in the direction of the axis Y. The roll is usually done by lowering of velocity of one pair of rotors and increase in the velocity of the other pair, which is in the mirror position by the axis of rotation.

Yaw is a movement created by inequity of moments created by rotors [1]. Technically is caused by lowering of moments crated by pair of rotors placed centrically symmetrically to the origin of coordinate system and increasing of the moments of the other pair of rotors [1].

All these described movements do work for the setup illustrated in the figure [1.2.1]. These models can be changed depending on the UAV design, but the idea behind all of the motions remains the same for vehicles with different numbers of rotors, different layouts and different directions of rotations of rotors.

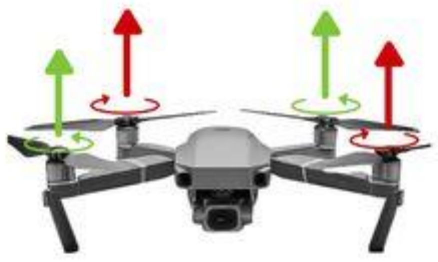

Figure 1.2.2. – Simplified forces on quadcopter

Another thing to mention is flight modes in which the drone can operate. PX4 autopilots supports acro and stabilized mode. In acro mode sticks of yaw, pitch and roll control angular rotation and after releasing them, the drone stays in its current orientation [5]. In the stabilized mode, the yaw, pitch and roll stick behave the same, however after centering them, the drone returns to into the position of just hovering in the same place [6]. If tested later outside of this work, the drone would be intended to work in stabilized mode.

After establishing some basic information about what can be controlled about drone and a very simplified information about what causes the movements to be controlled, following part will be introduction to the chosen hardware and then followed by introduction of used software.

2 Introduction to used hardware

2.1 History of Pixhawk and PX4

This part starts with a little history background of utilities, as it is nice to have at least a little knowledge about the story of what is worked with.

The origins of the Pixhawk project dates back to 2008 and a man called Lorenz Meier, who tried to make drones fly autonomously. [4] For that task he recruited 14 other students, and in 2009 the team called "Pixhawk" won the European Micro Air Vehicle competition in the indoor autonomy category. This version using protocol MAVLink was later released as open-source software [7]. After approximately two years later, user interface QGroundcontrol was released [7]. The software we use currently is called PX4 and is a fourth from scratch rewrite of the original project [7].

Pixhawk is a hardware board for controlling drones and other flying devices. The first one released under the name Pixhawk, followed by Pixhawk 2 (is referenced to under the nickname "Cube") [7]. Following the trend versions 2, 3, 4 have been released in collaborations with companies mRo, Drotek and Holybro, including versions Pro (v. 3) and Mini (v. 4). All of these types are discontinued [7]. The currently manufactured versions are Pixhawk 5X, 6C, 6C Mini, 6X in collaborations with Holybro and CUAV Pixhawk V6X [7].

After a brief introduction to Pixhawk, next logical step is to get basic information about hardware, that is going to be used.

2.2 Technical possibilities of Pixhawk 6x mini

For the realization of the work, a board Pixhawk 6x mini had been asset in the assignment. This decision originates from the fact, that despite attempts of working with older version, Pixhawk 1, author was unable to make the Pixhawk 1 board function properly in Simulink. Then the choice of Pixhawk 6x mini was made by ing. Bušek ordering a newer version, probably as it had been the newest board available at the time of creation of this work.

In this section of the document, properties of the board we use, that is Pixhawk 6x mini, will be described as the used board is vital part of the whole project as well as it gives compacted info about some basic properties.

The whole board weights around 26,5 grams, with dimensions of 43,4 x 72,8 x 14,2 *mm*, with built in gyroscope, accelerometer types (ICM-42688-P, BMI055), compass (BMM150) and a barometer (BMP388). The FMU processor powering the vehicle has 2 MB flash and 1 MB RAM memory and is working on frequency of 480 MHz [8].

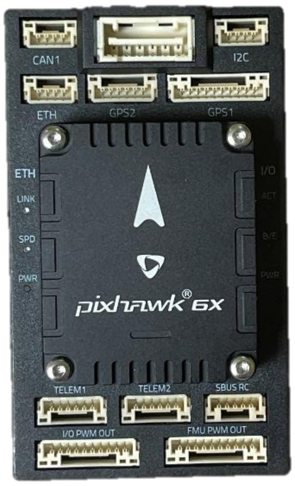

Figure 2.2.1. – Pixhawk 6x mini board

One power input for the source is to be found on the top of the board, two GPS inputs, one with safety switch port, second one is basic, two telemetry receivers for radiocommunication with ground stations or other modules, an ethernet port supporting data transfer up to the speed of 100 *MB/s,* micro SD card slot, pulse width modulation output and input (PWM I/O) ports with possibility to connect power distribution board, FMU I/O port, a CAN slot for CAN supported devices, SBUS RC port for manual operating using controller in this case with FrSKY 9D+ controller and FrSKy X4R receiver, I2C port and FMU and I/O Debug ports [9] [8].

At last, there is a USB-C port for powering and data transfer while connected to the ground station such as computer. This is a significant upgrade over the micro-USB used on older models such as Pixhawk 1. This upgrade does significantly reduce the time of setup and starting of models from computer, over the older models.

Current input ports are limited to 1,5 *A*, with various input voltages, depending on the port used and the whole board is operational within the voltage range of 4,75 to 5,25 *V* [9] [8]. According to the documentation provided by the manufacturer Holybro, the board should withstand and be operational within the temperatures of -40 to 85° C, but there is no guarantee that other connected periphery do operate in the same range [9].

After getting to know the board, next step is going to cover, how is it going to be operated with the board in confines of this work.

3 Communication with and within the drone

For the correct control of an UAV a way of data transmission between user and system needs to be set. In the following part it is going to be covered basics of how the connection is made as well as how this communication could be done if this project is worth further exploring. A protocol on which the drone operates will be mentioned too.

3.1 Station – device communication

An important part of the whole system is setting the communication between the system and a control station. Explicitly in this case the station being a notebook with Simulink and a drone powered by PX4 autopilot software. This section will cover a bit of vision for the future work with Pixhawk board and explains decisions made within this work.

The main protocol on, witch all devices using the PX4 autopilot communicate remotely with control stations, is MAVLink protocol (explained further in the thesis). The MAVLink protocol, used for data transmission in Pixhawk powered devices, supports transmissions of data through WiFi and Ethernet as well as serial telemetry or can be bypassed via cable [9]. For the future usage, it makes more sense to use serial telemetry, because of the hope of being able to fly the device outside specifically a telemetry radio would be likely to be used, however whilst working on this thesis, a cable had been used for the communication with control station. [8].

The radio most likely to be used is SiK Telemetry radio V3 by Holybro [8], that can be bought in combinations of power 100 or 500 *mW* and working on Frequencies 433 *MHz* or 915 *MHz* (their usage being depending on the region of usage, as operating on some frequencies is forbidden by the law) and has a plug and play feature, which should make the usage easier. For operating in the EU *433 MHz* is to be used, as the other frequency is forbidden [10].

But whilst working on this thesis, the communication between station, being a notebook, and the Pixhawk board is being realized by USB and USB-C cable, as it is completely suitable and easiest mean of data transfer and as of now, there aren't other requirements apart from testing the functionality and proving a concept. Other bonus of usage of a cable is unnecessity of need for battery powering the board, as the board is powered via the cable.

In addition to communication with the control station, other control is to be connected to the Pixhawk board, that being an RC controller. This controller should be used for manual flight control and obtained RC signal is to be shaped by a shaper. As for hardware, the used controller is FrSky Taranis X9D+ with FrSky RX6R receiver, because it was already in possession of České vysoké učení technické .

This section explained, how the hardware communication is carried out, next a software part will be touched upon.

3.2 MAVLink

A very crucial part of automatic control of the drone is a protocol used for communication. The Pixhawk boards use MAVLink, software first released in 2009 by Lorenz Meier [11].

MAVLink is a messaging protocol, that may and does find its usage in communicating with drones or other independent vehicles and as onboard communication protocol [11] [12]. The protocol is a publish-subscribe and point-to-point hybrid, where the publish-subscribe function is used for data streams, while the point-to-point part is used for retransmitting subprotocols, such as parameter protocols and mission protocols [11] [12]. The sent messages are in XML files which are later used by code generators to create software libraries in specific programming languages for latter usage in ground control stations or other MAVLink – based devices [11] [12].

According to MAVLink organization [11], the protocols (versions MAVLink and MAVLink 2) are very efficient due the usage of small packets (8 and 14 bytes), with the latter being more secure and therefore can be used in applications with limited bandwidth of communication. The protocol also provides for detecting of packet drops, corruptions and for packet authentications. It supports up to 255 concurrent vehicles and control stations on the network and does work on major operating systems such as Windows, Linux, MacOS, Android and iOS [11].

The majority languages are generated by a tool called mavgen, which support common programming languages such as C, C#, C++, Python, Java, Javascript and others. But there are other tools by independent projects that can be used too [11] [12]. This work will not cover them, as it is not this works main content.

In this work, there has not been a specific need to work with MAVLink, by itself, however an introduction to MAVLink was requested by the supervisor of this thesis, as well as if continued working further with drones and MAVLink based products, a need of working with it might be required.

4 The signal shaper algorithm

This work, as it has been set in the assignment, is set around trying to eliminate oscillation of a weight placed under a flying drone, specifically using a method with signal shaper. Following section covers the basics of control as well as describes the theoretical part of the shaper.

This technique of vibration reductions has been in the research and development from around the year 1957, presented by Otto J.M. Smith of University of California, Berkley [13] [14], and later followed by other works, that will not be covered in this brief introduction to this problematic as the goal is to try to implement this technique to the quadcopter control using MATLAB Simulink.

4.1 Laplace transformation

Physical systems are described by differential equations, that for the sake of solving are being linearized into linear differential equations [13]. This part is important, as the vibration damper, that is later to be implemented uses the coefficients counted from linearized model from an equation given from [13], that describes a system of physical oscillator, that would be created by a weight suspended under the body of a drone, which would be regulated by models from Simulink and therefore it is an information that should be mentioned how to count them, despite having no need to count them yet, as it is information closely related to the ZV shaper.

Laplace transformation is a transformation of a differential equation to algebraic equation [15], that should be simpler to find a solution to. The Laplace transformation can be used only under the assumption of the system defined by differential equation being convergent [15].Upon usage of Laplace transformation, the linear differential equation can be transformed from time dependent function to an algebraic equation.

The calculation formula is [15]:

$$\mathcal{L}\{f(t)\} = F(s) = \int_0^\infty f(t) \cdot e^{-st} dt \tag{1}$$

And in the case of constant coefficients of the linearized system, the differential equation could be simplified to [15]:

$$\begin{aligned} &a_n y^n(t) + a_{n-1} y^{n-1}(t) + \cdots + a_1 y\,(t) + a_0 = \\ &= b_0 + b_1 v(t)\; + \cdots + b_{m-1} v^{m-1(t)} + b_m v^m(t) \end{aligned} \tag{2}$$

Where:

n, m … are orders of derivations within the system

a, b … constant coefficients of the derivations

y, v … time dependent variables

The result of Laplace transformation used on equation (2) would be:

$$a_n s^n Y(s) + a_{n-1} s^{n-1} Y(s) + \cdots + a_1 s\, Y(s) + a_0 = \\ = b_0 + b_1 s\, V(s) + \cdots + b_{m-1} s^{m-1} V(s) + b_m s^m V(s) \tag{3}$$

4.2 Transfer function

In the example of compensating a single oscillatory mode system, as will be done in this work, there will be considered transfer function of the system [13] [16] [14]:

$$G_{(s)} = \frac{y_{(s)}}{v_{(s)}} \tag{4}$$

where $v_{(s)}$ is the systems input and $y_{(s)}$ is the system output in form of system after Laplace transformation (covered in section 4.1) and the transfer function is being defined as their division. The transfer function within the question of this thesis is function of physical oscillator under a fixed object, that may look as following expression that had been taken from [13]:

$$G_{(s)} = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2} \tag{5}$$

Where ω_0 represents the natural oscillatory frequency of the oscillatory system and ξ represents the damping ratio of the system [13] [16] [14]. This equation is according to a weight hanged under a fixed body. Solving the quadratic equation from the denominator of equation (5) of the oscillating system in this form, results in a solution, with complex roots of the equation, that are being substituted by real coefficients β and Ω, in a form of:

$$r_{1,2} = -\beta \pm j\Omega \tag{6}$$

Where:

$$\beta = \xi\omega_0 \tag{7}$$

is the real part of the coefficients, and where:

$$\Omega = \omega_0\sqrt{1-\xi^2} \tag{8}$$

is the imaginary part of the solution of the denominator of the equation (5).

4.3 Zero vibration shaper

As the goal of this thesis is to proof a concept of control using Simulink, a simple enough method had been chosen, as the need for more complicated algorithms with more complicated usages could be contra productive, therefore for the purpose of dampening the oscillatory mode in this case, the zero vibration (ZV) shaper will be used. Such shaper is described by the equation [13] [16] [14]:

$$v(t) = Aw(t) + (1-A)w(t-\tau) \tag{9}$$

Where A is gain, a constant coefficient, that ranges within the interval of [0.5; 1], *w(t)* is the input signal and *v(t)* is the output signal of the shaper, both of them being time variable [13] [16] [14]. As seen in the equation, one of input signal is being delayed by time constant τ. The time delay and gain are both dependent on the properties of the oscillatory system and are derived from the oscillatory frequency and the dampening of the system [13] [16] [14]. Where gain is calculated as following expression:

$$A = \frac{e^{\frac{\beta}{\Omega}\pi}}{1+e^{\frac{\beta}{\Omega}\pi}} \tag{10}$$

and the time delay is calculated as [13] [16] [14]:

$$\tau = \frac{\pi}{\Omega} \tag{11}$$

4.4 Zero vibration shaper with time distributed delay

There are also other ways of dampening the oscillation of a system, such as using a time distributed zero vibration shaper (DZV) [13] [16] [14], that may result in better functionality and more desired results, but their implementation can bring other difficulties [13] [16] [14].

The DZV implements an idea of distributing the time delay equally within the timeframe of a delay, therefore the changes to the system may result in not as drastic interventions to the regulated system but its application should be considered accordingly to the subject of study (for example might not be suitable for initial testing).

The DZV can be described with usage of following equation [13]:

$$u(t) = \int_0^T w(t-\eta)dh(\eta) \tag{11}$$

where *w* and *u* are system input and output respectively, and where *h* is a function of time determining the delay distribution over the internal of the integral *[0, T]* [13] [16] [14].

The distribution can be described using a trapezoidal function as seen in the equation (11) [13] [16] [14]. The trapezoidal function can be described in the time interval of [0, ϑ]. On this interval the function can be prescribed as [13] [16] [14]:

$$h(\eta) = 0, \qquad \eta < 0 \tag{12}$$

$$h(\eta) = \frac{1}{\vartheta}\eta, \qquad \eta \in [0, \vartheta] \tag{13}$$

$$h(\eta) = 1, \qquad \eta > T \tag{14}$$

Using this expression, the equally distributed time delay can be than expressed by the equation [13] [16] [14]:

$$u(t) = \frac{1}{\vartheta}\int_0^\vartheta w(t-\eta)d\eta \tag{15}$$

And the shaper function can be described using the following equation [13] [16] [14]:

$$v(t) = Bw(t) + \frac{1-B}{\vartheta}\int_0^\vartheta w(t-\eta)d\eta \tag{16}$$

Where v is the output signal, w is the input signal, and B is a numerically obtained parameter corresponding to the solution of equation as proven in the paper [13].

For the implementation part of DZV, the main obstacle can be seen as the need for discretization of the algorithm. The discretized version of the DZV shaper can be described as a middle ground between ZV and continuous DZV, with various results depending on the used discretization algorithm and time parameters such as time step [13] [16] [14]. However, within the confines of this work, the further theoretical substance of the zero-vibration shaper with time distributed delay won't be covered further, as the main goal was to show other existing way that might result in different, maybe more desirable results.

After establishing the utilities used for control, next part goes over where the signal shaper will be established within the regulation process.

5 Control process

Drone control, due to its high demands on the quality of control of the system, is a problematic and can be categorized into the section of analog control, as there is a need for other, than two state logical control (despite the fact, that some properties of the system may be regulated by logical control).

In the confines of analog regulation, the control may be conducted with or without regulation feedback. In the case of drone, there is a need for feedback, however, as will be further specified, the work within this thesis does not involve need for work with feedback of the system, as the main goal is to shape input signal through given method (ZV shaper).

In analog control the lived customs are, representing the system using control loops. Using such method a simplified model is being used, constituting of regulator (that can be represented by an automatic control unit in this case Pixhawk), and a regulated system (also referred to as system or regulated object, e.g. a drone with weight). Example of schematics of the drone control is shown in fig. [5.0.1]. With regards to the complexity of models, they are vastly dependent on the complexity of regulation of a system and number of controlled values. As seen in fig. [5.0.1] information between elements of regulation is transitioned using unified signal.

The idea of ZV dampening is that of placing function shaper block executing the signal shaping in between of the remote control, and the actuators control of the quadcopter. In the case of devices used during this thesis, the controller used is FrSKY Taranis X9D+ and the Pixhawk 6x mini board. There shouldn't be a need to work directly with MAVLink, as the RC sent signal is not in MAVLink form. Simultaneously to direct RC signals, a control model of the shaper is being loaded from control station (notebook via USB) to Pixhawk. The usage of USB bypasses usage of MAVLink connection. There shouldn't be a need for making changes to the sole control of a drone flight mechanics because there is a hope of using the premade modes of the drone (stability and acro modes, [5] [6]) and messages controlling the movement. The scheme can be visualized as fig. [5.0.1].

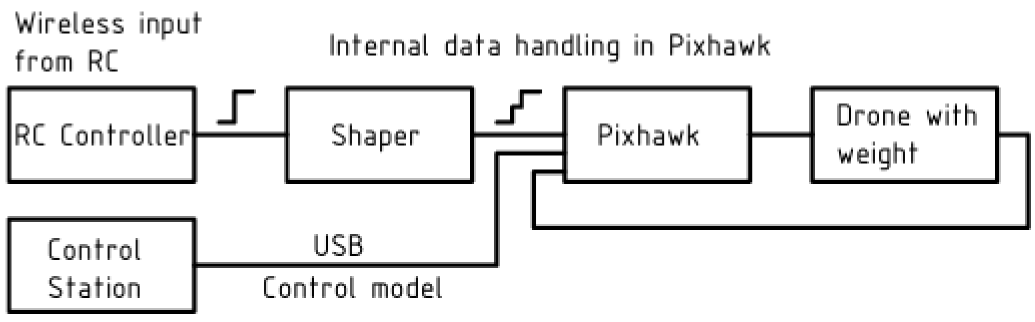

Figure 5.0.1. – Simplified schematics of drone control task with shaper

In older experiments, that had been realized by (Daniel Bukovský and Jaroslav Bušek [17]), employees of the Czech Technical University, the signal shaper had been realized by a device outside of the Pixhawk board. In this case, the importance is placed onto executing the shaper function within the software of a board, as it should be simpler, and more versatile way and may lead towards other, more advanced usages. The movement should in this work be carried out by uORB messages, a part of PX4 software.

After establishing what is going to be done in this work, the practical part follows. The process will be taken from installation of necessary software, as problems did occur in this process, until the implementation of ZV shaper and testing it using simulation in Simulink.

6 Possibilities of code implementation of UAV Toolbox

This part is on the verge between the theoretical and practical part of this thesis, as it describes the motivation of using MATLAB and Simulink's UAV Toolbox (that being its possibilities), while being already written from the practical experiences of working with mentioned software. However next section 7 is to be described as solely practical.

Within this thesis, a MATLAB Simulink and its add-on UAV Toolbox was chosen in the assignment. For getting a basic orientation in UAV Toolbox in Simulink, this part will touch the surface of software possibilities of mentioned add-on.

The main strength of this method of working with PX4 is the unnecessity of programming the code itself, which means, that there is a little to no need for the knowledge of the syntax of the used language, as Simulink does create a code C++ automatically. Therefore, the user experience should be easier and more enjoyable and should be made simpler for users.

In Simulink the model creation is based on the method of placing function blocks from libraries and connecting them throughout the graphic interface. Then the created sequence of blocks with functions is being carried out in the order of the connections between blocks.

The blocks by itself are categorized in libraries, with built-in Simulink functions and can be extended by utilizing add-ons, such as already mentioned UAV Toolbox, but should be able to work with other extensions too, therefore securing a wide specter of possibilities. The coverage of the whole Simulink possibilities is very comprehensive, therefore only the blocks used in the control are being covered and parts of the UAV Toolbox add-ons.

Within the UAV toolbox add-on are other blocks with various purposes to be found. They are categorized into two main folders:

- UAV Toolbox
- UAV Toolbox Support Package for PX4 Autopilots

In the first named are subcategories of:

- Algorithms
- MAVLink
- Simulation 3D
- UAV Scenario and Sensor Modeling
- Utilities

With all of them being quite convoluted functions and not being used in the framework of this thesis, only few examples will be mentioned.

There are pre-defined functions of minimum jerk and minimum snap polynomial trajectories, path managers, obstacle avoidance functions, sensor simulation blocks, scenario functions and coordinate transformation blocks and other. Their description can be found on MATLAB support on the internet or in the block description within the

Simulink after hovering mouse cursor over a to be used block, but will not be covered further, as they will not be used and are named just for the example of functions that can be found.

The UAV Toolbox Support Package for PX4 Autopilots folder is more useful for this thesis, as there are blocks such as PWM outputs and inputs, as well as for example sensors blocks, that can be used for real-time data gathering and subsequentially can be used for the control of the whole system, depending on the need of data usage. There are also blocks for writing and reading messages of the system, uORB blocks and other utility blocks. The functions used in the sole algorithm will be expanded further in the section of the algorithm implementation.

7 Installation of the UAV Toolbox

Finally getting to the main software part of this work, installation of used Simulink add-ons will be described, as it is a begging of the whole process.

For purposes of this bachelor's thesis, the desired software is MATLAB Simulink, concrete version in usage being MATLAB 2023b.

The Installation is done through Add-ons icon, that is found on homepage after opening MATLAB. Next desired step is clicking on Get Add-ons by pressing the arrow button under Add-ons.

If the steps have been done correctly, Add-on explorer should open. To find the required Add-Ons, use the search bar, looking for a software named UAV Toolbox support Package for PX4 Autopilots, but searching solely for keyword "PX4" is enough to find desired Add-Ons.

After opening UAV Toolbox support Package for PX4 Autopilots, there is a list of requires, under the section of "Requires" (located on the right side of the screen). The requires for proper functionality is having installed following programs and MATLAB extensions:

- Simulink
- Embedded Coder
- MATLAB coder
- Simulink Coder
- UAV Toolbox.

All the parts requested must be installed, before installation and opening UAV Toolbox support Package for PX4 Autopilots. If done successfully, an installation process is eligible to be done.

During the Installation the first step is to assign installation folder, where the PX4 software will be located in computer's database. The folder can be chosen by clicking the button "Browse". In this case I used the default settings of installation. After choosing postulated folder, clicking on "Verify installation" is required before being able to continue using "Next" button.

If the first step compiled successfully, the second step is to validate the PX4/Home folder, which should be found automatically inside the PX4 folder, and the whole step should be as simple as pressing the "Validate" button and then "Next" button.

The third step consists of selecting Simulink application. The chose is between:

- Design flight controller.
- Design path follower.

In our case we have chosen the first option and have continued.

Fourth step is choosing the device in usage, for purposes of this work, it's been tried with PX4 Pixhawk 1 (issues and hardships with Pixhawk 1 are described further in this work) and PX4 Pixhawk 6x. There is also a need for choosing firmware to be installed. In this work the px4_fmu-v6x_multicopter has been chosen. There is also a possibility of installing Custom CMake file for developers or other versions of firmware, that can be found on official PX4 GitHub.

Fifth step is selecting startup script. In this work the option of Use default startup script (rcS) has been chosen.

Sixth step is to install QGroundcontrol, mission planner mentioned earlier. On the screen there is a link to QGroundcontrol website, from where this program can be installed, since I already have had this software, I only did the step of Verifying installation and continued by pressing the "Next" button.

Seventh step consists of building firmware to the computer where it should be as simple as pressing the button "Build Firmware". The firmware should start building with the process of installation can be monitored in MATLAB's command window.

Cannot find source file:

monocypher/src/monocypher.c

The problems were resolved by manually creating folders monocypher and src and manually inserting programs monocypher.c and monocypher.h. This software is cost free obtainable from internet and the final folder tree of changes looks like:

C:\PX4\monocypher\src

The whole process of building the firmware is dependent hugely on the internet connection of the PC and on the power of computer that is worked on. This process can take well over 15 minutes on a laptop for casual usage (Acer Nitro 5 had been used).

If the building was successful and pressing the "Next" button, the eighth step appears. In this step, connecting the Pixhawk board via cable is required. A COM port of the PC into which the board is plugged in, needs to be filled into the "<Enter Upload Port>" field, and then the "Upload Firmware" button must be filled in. If everything goes well, a pop-up window appears where a need is to follow directions on screen.

There is also a required step of pressing the "Get Accelerator data" button, which is located next to the "Upload Firmware" button. From where current data from boards built in accelerometer can be accessed (one data set each click of the button). If the data are accessed successfully, a "Next" button can be pressed.

The ninth step is just to click "Finish button" and the whole installation should be completed.

Before the version Pixhawk 6x had been delivered to the facilities of ČVUT, there was an attempt to try and make work the older version of the board, Pixhawk 1, which was already in possession of České vysoké učení technické. This attempt had ultimately failed, due to combination of reasons. During the time of creation of this work was Pixhawk 1 already a discontinued product without technical support. The firmware offered in the fourth step of installation was px4_fmu_v3_default was ultimately unable to be built into the board. An attempt of building Custom CMake file has also been made, with the px4_fmu_v3_default firmware downloaded from official PX4 GitHub.

This attempt had also been ultimately unsuccessful. During one try, amongst reinstallation attempts, the firmware was able to build, but could not be uploaded to the board. The only thing, that could have been done was try to overwrite silicon errata of the Pixhawk 1 board, but in the interest of perseverance of the board, this option wasn't followed through, as the board, as the information has been obtained from QGroundcontrol, only had 1 *MB* flash, instead of required 2 *MB* required for overwriting the silicon errata safely. Despite the manual of Pixhawk 1 stating, that the board should have 2 *MB* flash, the risk of damaging the board wasn't deemed worth the try.

After the installation, mainframe of the work, that is implementation of ZV shaper will be covered, from explaining blocks, that will be used, through first task to final models.

8 Selection of used utilities PX4 in MATLAB Simulink

The following part of this bachelor thesis will cover some of the blocks and functions that can be used within the model sequence and will later be used, therefore they are described more thoroughly.

While opening the libraries, there are blocks that are the basic facilities of the Simulink, while also the blocks that are added on by Simulink add-ons such as the UAV Toolbox, the add-on PX4 works with. Obviously on every device the blocks found in the libraries may vary, depending on installed add-ons.

In the libraries are blocks to be found with various functions and variously convoluted operation modus. Within the code implementation for zero vibration shaper, very basic blocks are used, as the whole function is a trail of mathematical operations. The blocks used are:

- Radio Control Transmitter – a block for connecting radio transmitter used for drone control, that outputs numerical values depending on lever positions of the controller. These outputs should be used as input data for the system and are to be shaped by the shaper.
- Constant – a block that gives constant numerical output. The output can be limited by value constraints, and the data type can be set to desired format, such as double, uint16 or Boolean, depending on the usage needed later in the Simulink model.
- Function – this block is a definable mathematical function, that upon double-click opens a MATLAB function tab, where using MATLAB syntax a custom function can be created. Outputs and inputs are customizable with multiple instances of both.
- Display – a block that upon connecting a numerical input displays its value. This block can be used as real time control of the model.
- Delay – multiple types of delay can be found in Simulink libraries, some used for analog simulations other for discrete operations. Within the later shown model either "Integer Delay" or "Variable Integer Delay" can be used, with the latter being chosen, as it has been deemed more user friendly to control time delay through a constant block, rather than setting it via block settings. Thing to note is that the delay is dependent on sampling time of Simulink. With different sample time, different inputs need to be inputted, for desired functionality.
- Data type conversion – a block from default Simulink libraries, used for converting data types of given value to a different data type i.e. transforming data type from "double" to "single".
- PX4 PWM Output - a block from UAV toolbox, that outputs values out of the Pixhawk board. It has multiple input channels, according to the RC control of the drone, reading the uint16 data type, as well as setfailsafe and arming of the outputs, reading Boolean data type.
- uORB Read – a UAV toolbox block, used for reading Pixhawk board status. The topic being read can be set within the block settings accessible after double-clicking the block, for example for PWM outputs value, that read the width of rectangle pulses sent to the actuators via "actuators output" topic.

- uORB Message – a block for selecting a topic to be modified in combination with a "Bus Assignment" block and input signal. The topic, that can be selected in the block settings can then be sent to subscribe via bus signal.
- Bus Assignment – a block from standard Simulink libraries, that can find its utilization in combination with uORB messages, as using this block a selection of one or more properties of selected topic can be chosen and transformed into bus signal mandatory for correct functionality of "uORB Write" block.
- uORB Write – a subscriber block from UAV Toolbox library, for executing a bus signal of modified uORB message. The topic setup in this block is necessary to be the same selected as the uORB message inputting the signal to this block.

9 PWM output control

The first task of working with PX4 autopilot using the UAV Toolbox extension to Simulink was to be able to establish connection within the board and successful usage of Simulink for actuators control.

Motors, that can be connected to the board are powered by voltage from ports, that can be found on the board, marked as PWM output. PWM stands for "Pulse width modulation" and is a way of creating analog output of discrete values obtained control. This technology is to be found in most applications related to electromotors.

The whole setup was prepared by following a tutorial created by Mathworks group, that can be obtained cost-free on their official website.

Following the step-by-step guide from internet, the whole setup was consisting of three constant blocks, a block of PX4 PWM output, uORB read block, bus selector and a display block.

Two constant blocks are connected to the Arm and set failsafe as seen in schema [9.0.1]. These constants are needed to be set as boolean type signal, that can be performed after double-clicking the constant block and inside the "Signal attributes" folder and under the "Output data type" section. The Boolean attribute of 1 is, following the accustomed rules, used as an "ON" option, while the Boolean 0 is representing the option "OFF". This functionality is very intuitive, but deemed necessary to be mentioned, to clear the fact of the program working as expected.

The remaining constant is outputting the signal in the type of uint16, straight to the channels input ports. These channels represent the PWM output ports, from which the voltage to the actuators can be lead, while the value of the constant represents the width of voltage pulses dependent on the sample time. These values should be found within the range from 900 to 2000.

The uORB read block enables to read the protocol within the board works and outputs the message, depending on carried information. In this case the information "<output>" is selected in the bus selector and displayed. These values should equal to the values input to the channel ports. The whole functionality process can be than observed using an oscilloscope, after connecting cramps to pertinent outputs on the board.

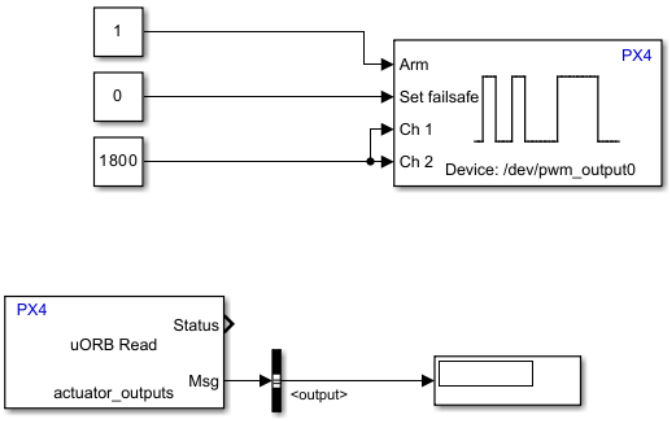

Figure 9.0.1. – Model of direct PWM output

For the desired functionality, this model was run in the external mode settings on the board. The building, loading, and running of the model was done without an issue, however even for the not as complicated, such as this, task, it took some time.

As the result of this model, the pulse width observed on the oscilloscope had width according to the input constant. This constant could have been changed even in the on-board running model, therefore it also proved the possibility of real time control of a system created within the Simulink software. The values were set to *1800 ms* and *1400 ms* and accordingly read on the oscilloscope.

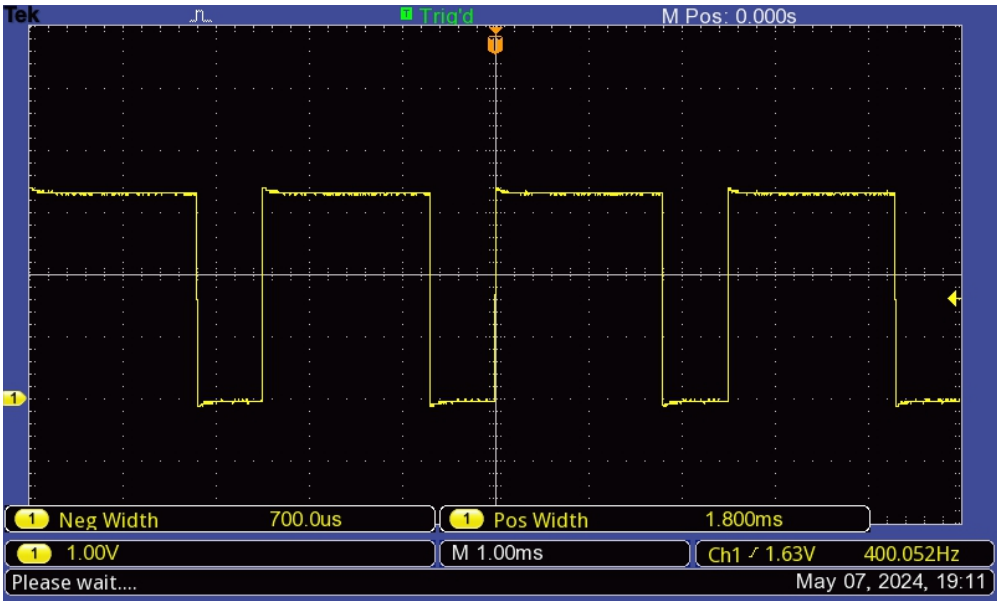

Figure 9.0.2 - – Oscilloscope measurement with 1800 value inputted from model

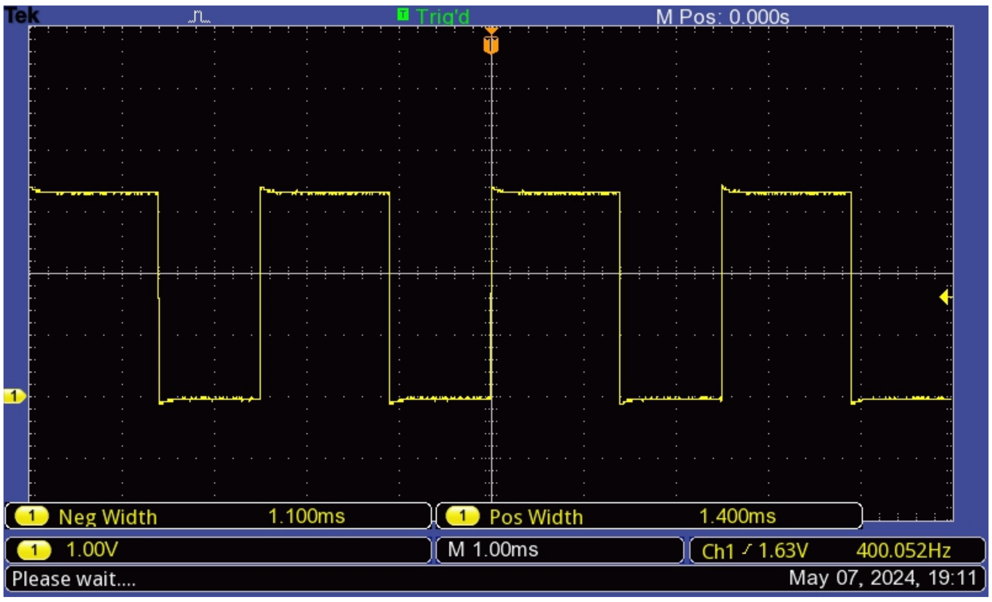

Figure 9.0.3 - Oscilloscope measurement with 1400 value inputted from model

Following this control, a more complicated, completely custom, model for a drone control can be created using Simulink and could be an opportunity for future projects.

10 Zero vibration algorithm implementation

As it already had been previously stated in the section 5, the explained shaper algorithm is set to be implemented outside of the closed loop drone control, in between the remote controller and the pre-designed control of the drone. Therefore this part of this thesis will cover the whole created Simulink model from inputs to outputs.

10.1 Controller inputs

During the implementation, the first step is to set connection of used remote controller, that is later to be shaped. Using the UAV Toolbox, there are at least two ways of reading RC data, one of them being using premade "Radio Control Transmitter" block, and second on is to read the uORB messages of the system using "uORB Read" function block. Their usage may differ, with regards to preferences of person working with the toolbox.

To read uORB messages, the "uORB read" block can be selected in the "PX4 uORB Read and Write Blocks" folder and set to read manual_control_setpoint message, that is to be set after double-clicking the block and then selecting desired message. This way the information of internal control messages are being read.

In contrast, the "Radio Control Transmitter", found in "PX4 Sensor Blocks", is a block, that is designed to read current values set on the connected controller. These values are dependent on stick positions and outputted using "Channels" outputs. Upon double-clicking on the block, the read channels can be chosen (the block supports up to 8 output channels), as well as other settings.

For the purposes of this work, the "Radio Control Transmitter" block was chosen as the main way to input information to the system, as it seemed to be less complicated to set up and it is also more graphic, therefore readability of the model seemed easier.

10.2 Shaper function

The obtained signal can then be shaped by desired signal shaper. In the case of this work, a ZV shaper is realized, using "MATLAB Function" block, that is standard MATLAB utility. Inputs into the shaper are:

- *w* – which in this model is the output signal obtained from the "Radio Control Transmitter" block
- *w_delayed* – is also the output signal from the "Radio Control Transmitter" block, but delayed by a time constant τ, using "Variable Integer Delay". It is according to $w(t-\tau)$ from (9). Multiple delay functions can be found in Simulink libraries, but for the model to function properly, only discrete time delays can be used, as if tried for continuous delay, the model will not build and shows errors. For user friendliness the "Variable Integer Delay" is being used, as the length of the delay is being set using a constant, and therefore there is close to no need for user to open the settings of the delay block. (The only need for dealing with the block settings should occur only in the case of time delay exceeding the upper limit of the time delay input of 20 000).
- *A* – a constant of the shaper from a range between [0.5; 1]. This constant is calculated as seen in (10), and in early versions of this work had been calculated inside the model, from the oscillatory system properties, however that proved to be ineffective while setting the constant value and later changed to simple input realized directly by an input constant A.

These inputs are defined via MATLAB's syntax, as after double-clicking the "MATLAB Function" block, the program opens a text interface strongly resembling common MATLAB's command window, where a function can be written. Neat part of this process is the fact, that the basic syntax of defining the function is already prepared by the program.

The changes made into the function then is defining input matrix, consisting of the previously mentioned inputs, name of the function, in this case for clarity of the system, the function was named "Shaper" and outputs, in this case only a variable *v*.

Following the creation of the main frame of the function, the computing part is written into the body of the function. The whole function looks like:

```
function v = Shaper( A, w__delayed, w)
v = A * w + (1 – A) * w__delayed;
```

Figure 10.2.1. – Shaper function code

10.3 **Data transformation**

With regards to the desired usage in drone control, the output obtained for the "Shaper" needs to be recalculated, as the uORB message, that is used later within the model is operating in the range of input [-1, 1]. The message is called manual_control_setpoint, however many of uORB messages do operate in the same range of values, therefore this transformation can find other usages.

The recalculation is based on triangle similarity as shown in fig [10.3.1]. This similarity can be expressed by a general equation:

$$y = \frac{v - \min}{\max - \min} \tag{17}$$

Where *v* is the current value of input, *min* represents the minimal input value from the RC controller (982) and is larger than *0* and *max* the maximal input value (1996). The output *y* then is a value within the range of [0, 1].

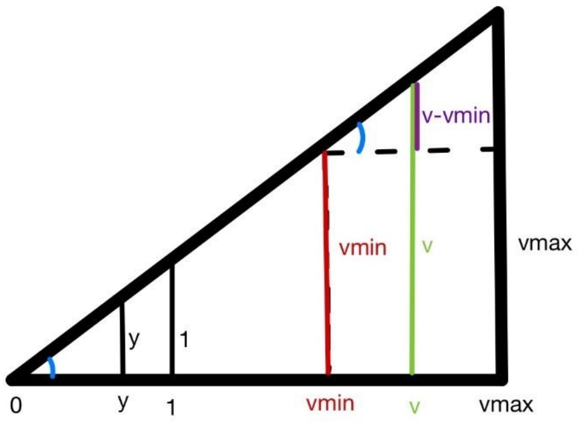

Figure 10.3.1. – Triangle similarity for signal transformation

For adjustment of this function to output values in the desired range of [-1, 1], the function is modified to the following form:

$$y = \frac{v - \min}{\max - \min} \cdot 2 - 1 \tag{18}$$

Where the whole expression is multiplied by two, to expand the range and then 1 is subtracted to move the values to be set around 0.

10.4 uORB message

Moving onto the final part of the algorithm implementation of signal shaper, the last remaining thing to do is to convert the already transformed signal onto drone movements.

For that purpose, a whole drone control system should be allegeable for creation in MATLAB Simulink, using PWM output control, and using channel controls straight from the user created model. Example of such model can be found for instance within the Mathworks PX4 examples on their website, where a model designed for altitude control can be found.

However, such systems do require extensive interventions if premade, or are complicated to design, therefore it was decided, to try and use uORB messages, that should work within the built-in drone flight modes.

uORB is a publisher-subscriber protocol system and in Simulink is utilized by "uORB Message" and "uORB Write". The chosen topic of control is manual_control_setpoint message, that should have the options of controlling the movement of the drone dependent on directions of movement. However, ambiguities occur in this part, as the uORB manual of the used topic, that is to be found on the internet, states, that movements of the drone are roll, pitch, yaw, thrust, however the outputs of the uORB message in Simulink are *x, y z, r*.

Using logic, the movements are expected to be as:

- r – as rotation around the vertical axis.
- z – thrust as a movement in the direction of vertical axis.
- x – for pitch, the movement "forward".
- y – for roll – the movement "sideways".

However, without practical experiment using this uORB message, only the assumption can be used.

Assuming the described description of movements, the ZV shaper is implemented for a one-dimensional movement, since the main goal is to prove a concept, more comprehensive usage could bring longer loading and if there would be a need to implement the shaper for other movements, same build could be used, just multiple times with different inputs. Chosen movements is either pitch (*x*) or roll (*y*) (in this case pitch). For proper functionality of uORB messaging a system of "uORB message" "Bus Assignment" and "uORB write" blocks are used. In both publisher and subscriber blocks is selected "manual_control_setpoint" topic is selected.

To input transformed data obtained from the shaper into the uORB messaging system, the output signal is connected to the "Bus Assignment" block, in which the topic of *x* or *y* is selected (in the example model parameter *x* was chosen). The selection is being carried out by double-clicking the "Bus Assignment" block, removing all default or to the chosen topic non-related signals and then selecting the *x* signal. This setup does write the calculated data into the system and for the control they can be obtained by using "uORB" read block.

11 Simulink model execution

The whole model of zero vibration algorithm control described earlier had been implemented to Simulink and then had been ran in external mode, on the Pixhawk 6x mini board connected to the control station via USB-C cable on multiple instances and versions, as the whole process had undergone bit of evolution in terms of expectations of how the end result should look like, despite the original pitch being well in the direction of the way of the end result.

For demonstration purposes, the control values from the RC controller are being read from channel 3, despite the corresponding lever normally controlling vertical lift. This had been chosen for the reason of this lever being the only one that after deflection from its central position stays at the same position, therefore controller operator does not bring additional errors to the system by unsteadiness of their hands.

11.1 Initial version

The first and the second iteration of the model was similar to the schema shown in fig [11.1.1]. This model follows the direction described in section 5, however the first model involved two more calculations of coefficients A and τ from the real and imaginary coefficients of equation (6) of the system, using "MATLAB Function" blocks. These settings, however, proved inefficient with regards to being user-friendly, as for the demonstrative purposes the changes to the shaper coefficients were too complicated to find results of the calculation according to the intervals they should occur in.

Another difference between the first and the end product was in the system output. In the initial version, the system output was to be executed via pre-prepared

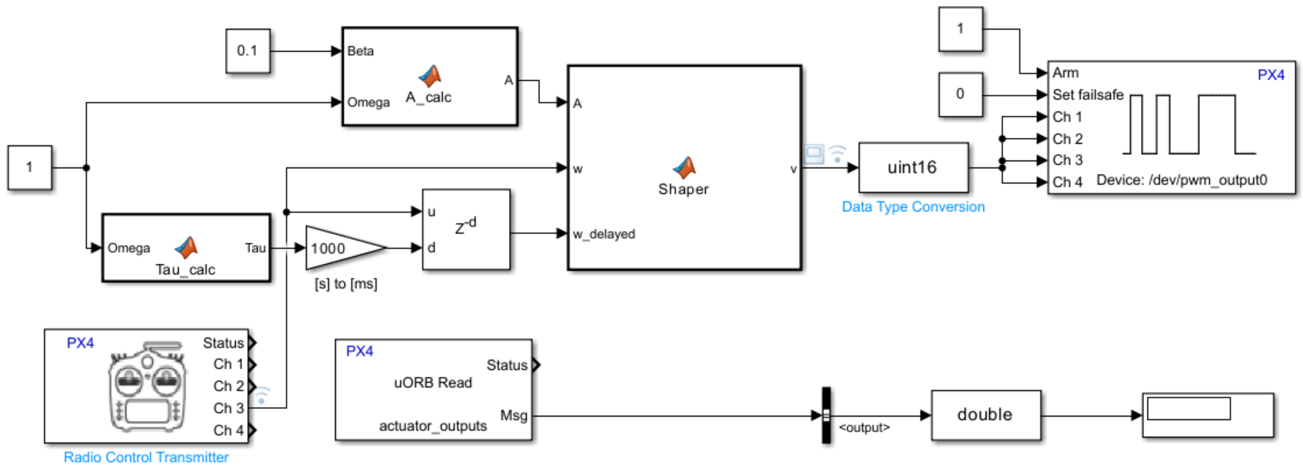

Figure 11.1.1. – Initial version of shaper implementation

uORB message block of "PX4 PWM Output", that, as explained, was deemed as not the way of desired output. In regard to the loading time of the initial model, proper measurements were not undertaken, however this version didn't have any noticeable differences in loading time to the end version.

11.2 Final version

The end version is shown in fig. [11.2.1.] This version is noticeably simplified, for better user experience. The inputs of coefficients had been changed to constants, with the need of them being calculated outside of the model. For better orientation, both units of A and τ are closer specified under their constant block. In the case of the time delay τ, transfer of units is realized by "Gain" block, from seconds, inputted by user, to milliseconds, the sample time in which is Simulink operating. If different sample time is used, different gain must be set.

However, the most noticeable change is the output of the control algorithm is being carried out via "uORB Message" and "uORB Write" blocks with connected input from transformation into the range of [-1, 1]. Process of work with uORB messages is closer described in section (10.4).

The whole process is also being monitored on multiple instances within the system. The data, despite being slightly delayed by the program, are being observed by "Display" blocks, that are not to be found in fig [11.2.1.], as they have been left out for purposes of visibility of printed version.

Display block described in the system as "Shaper control" visualize numerical value of signal obtained straight from the shaper. These values range within the numerical output of the RC controller, specifically values of [982, 1996]. These values may vary depending on the controller and are important, as their confine values need to be set as minimal and maximum values in "Transform" function.

The output of "Transform" function is being displayed by "Transform control" display. These values should occur in range of [-1, 1] if correctly set within the body of the function. When set incorrectly, the uORB messaging system can handle the value, however this situation should probably not appear, as the chance of correct function of the system is not likely.

Last display block is set to read the data selected and then transformed to different data format from “uORB Read” block. The data displayed in this block should be equal to data displayed in “Transform control” display. (While testing, the numerical data were indeed equal, however due to skipping of displays from the schematics it is not to be seen in the fig [11.2.1.]).

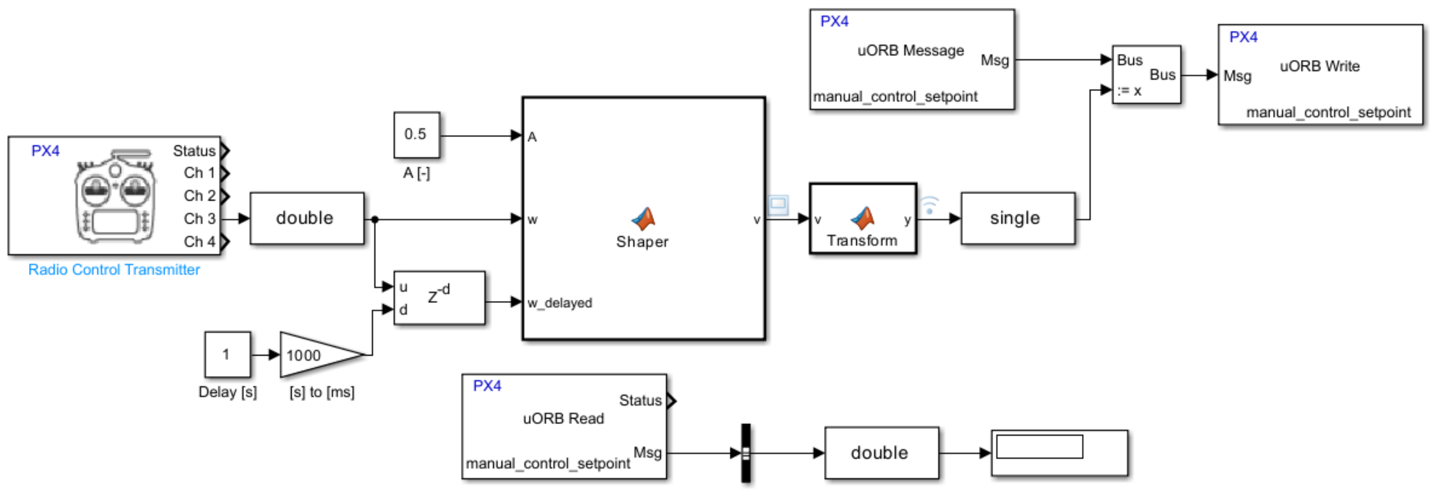

Figure 11.2.1. – Final version of shaper implementation

11.3 Function outputs in time

A desired output of the whole system can be considered, as the output signal depicted in time. Simulink offers an option to monitor data in time of the simulation or externally running model using "Data Inspector" option that can be found in the "Hardware" section of Simulink.

Plotted data in fig [11.3.1.] and fig [11.3.2.] are represented values of transformed signal (redrawn to a graph, not direct output), that is being written to uORB messages. The output value found in fig [11.3.1.] and fig [11.3.2.] signalizes a signal that is supposed to control the drone via uORB messages. The tests were undergone with coefficients *A=0,5* and with $\tau = 1\ [s]$ and with sample rate of *t=0.001 [s]*. The data had been smoothen for better visualization and the coefficients, since there haven't been any particular given system of drone with weight to calculate them from, were chosen to have clear visual results in terms of step size and time duration of the visual part of the step. If they were to be calculated it would be with for instance $\beta = 0$ and $\Omega = \pi$:

$$A = \frac{e^{\frac{\beta}{\Omega}\pi}}{1 + e^{\frac{\beta}{\Omega}\pi}} = \frac{e^0}{1 + e^0} = 0{,}5\ [-] \tag{19}$$

And:

$$\tau = \frac{\pi}{\Omega} = \frac{\pi}{\pi} = 1\ [s] \tag{20}$$

In fig [11.3.1.], the starting value was a central point of the controller lever, upper limit was position of lever completely up, and the finishing position was lever in the starting position. Imperfections caused by human inputs can be found in this graph. Within perfect system, the time of change of values would be $t = 0\ [s]$, however as seen the time of changes are not zero. In fact, these values are dependent largely on user controlling the system, and as seen, while returning to the starting point, an error caused by user may be spotted, as the slope of change is more gradual than the first, steeper input. Interestingly, the curve before the "step" is copied with the imperfection after the "step", therefore it may imply that the system can withstand even human caused errors in controlling the drone.

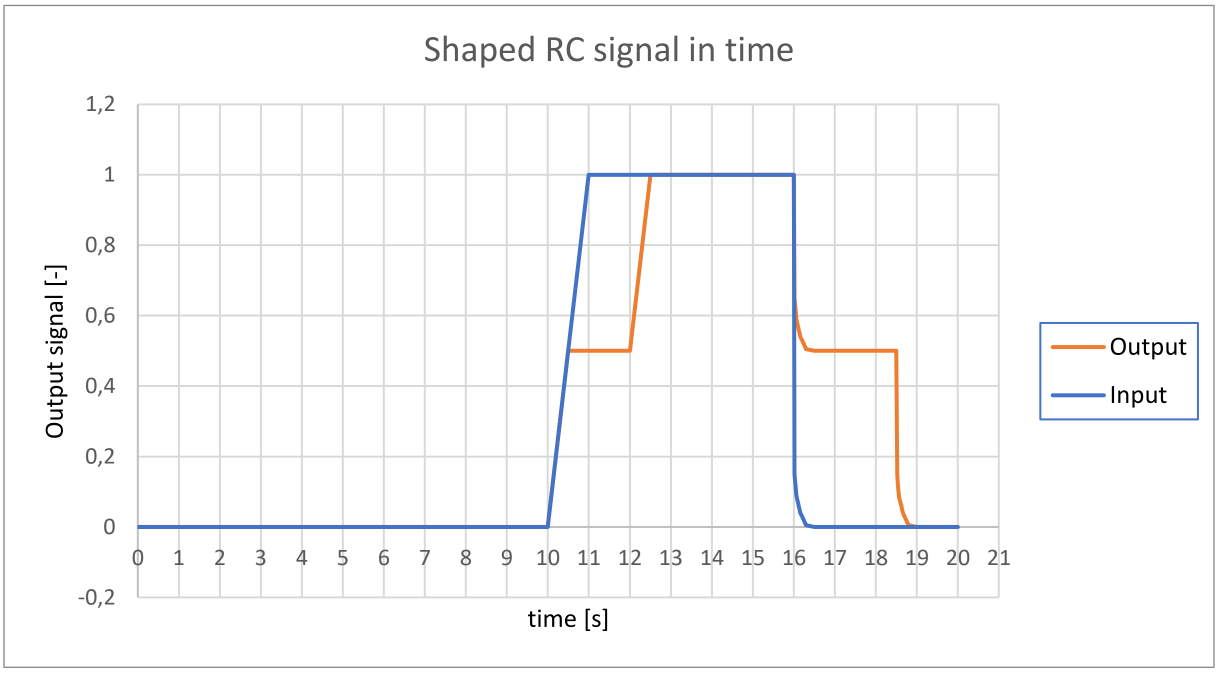

Figure 11.3.1.– First test results visualization

The second test was done for additional assurance of the system outputting correct data within the whole range of lever movements. The lever had been flicked from central, to up, to down position and the result graph is shown in fig [11.3.2]. Notice, that with *A=0.5* the output from up to down position is lumped around the *0* value.

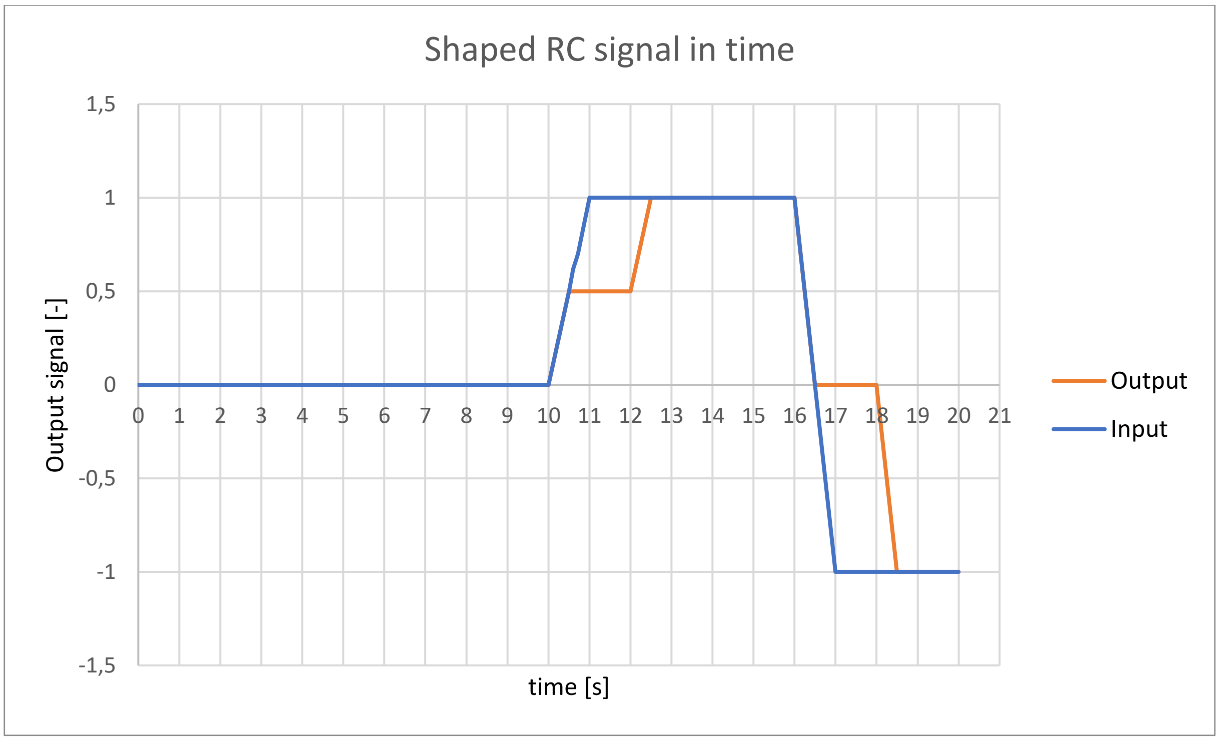

Figure 11.3.2. – Second test result visualization

Conclusion

This thesis covered a brief introduction to Pixhawk boards and PX4 autopilot in combination with MATLAB Simulink environment in its theoretical part, as well as an introduction to zero-vibration algorithm.

In practical part of this thesis the possibilities of applications in Simulink were covered and a simulation model was made.

The simulation model was later tested in software running on the Pixhawk board.

The practical part indicates the possibility of functioning drone control using MATLAB Simulink, as output voltage can be obtained from board's outputs, therefore a creation of whole control system may be possible, and system messages, that should control drone in pre-defined modes, can be created and written into the system, however the control via messages haven't been tested on real drone yet, therefore an exact conclusion shouldn't be drawn yet.

While creating the control model, an emphasis was put on simplicity and user-friendliness of the model, with the hopes of further continuation on this project, with experiments undergone on complete drone device.

Author is unable to state, whether using Simulink for drone control is a good method. The positives are being able to create control with little to none programming skills, and the fact, that ČVUT already owns licenses for MATLAB Simulink. The biggest negative may be the loading time of models to board, as they may take time (minutes), sometimes not loading at all, however it is possible, that other options are not any faster.

Bibliography

[1] C. LEE, S. KIM and B. CHU, "A Survey: Flight Mechanism and Mechanical Structure of the UAV," *International Journal of Precision Engineering and Manufacturing,* vol. 22, no. 4, pp. 719-743, 2021.

[2] H. YANG, Y. LEE, S.-Y. JEON and D. LEE, "Multi-rotor drone tutorial: systems, mechanics, control and state estimation," *Intel Serv Robotics,* vol. 10, pp. 79-93, 2017.

[3] B. ZHANG, Z. SONG, F. ZHAO and C. LIU, "Overview of Propulsion Systems for Unmanned Aerial Vehicles," *Energies,* vol. 15, p. 455, 2022.

[4] S. PANIGRAHI, Y. S. S. KRISHNA and A. THONDIYATH, "Design, Analysis, and Testing of a Hybrid VTOL Tilt-Rotor UAV for Increased Endurance," *Sensors,* vol. 21, no. 18, p. 5987, 2021.

[5] Dronecode Foundation, "PX4 Guide (Main) - Acro Mode," 2023. [Online]. Available: https://docs.px4.io/main/en/flight_modes_mc/acro.html. [Accessed 10 5 2024].

[6] Dronecode Foundation, "PX4 Guide (Main) - Manual/Stabilized Mode," 2023. [Online]. Available: https://docs.px4.io/main/en/flight_modes_mc/manual_stabilized.html. [Accessed 10 5 2024].

[7] Auterion, "The story of PX4 and Pixhawk," 2023. [Online]. Available: https://auterion.com/company/the-history-of-pixhawk/. [Accessed 3 2 2024].

[8] Holybro, "Holybro store," 2024. [Online]. Available: https://holybro.com/products/pixhawk-6x?variant=43008610402493. [Accessed 6 2 2024].

[9] Holybro, "Holybro Docs," 2023. [Online]. Available: https://docs.holybro.com/autopilot/pixhawk-6x/overview. [Accessed 6 2 2024].

[10] Český telekomunikační úřad, "všeobecné oprávnění č. VO-R/10/07.2021-8," 20 7 2021. [Online]. Available: https://ctu.gov.cz/sites/default/files/obsah/vo-r10-072021-8.pdf?_gl=1*nu6m9y*_ga*NTA5MjU0ODc2LjE3MTUzNTgzMzM.*_ga_0JN41LRMT4*MTcxNTM1ODMzMi4xLjEuMTcxNTM2MDA0Ny4wLjAuMA... [Accessed 10 5 2024].

[11] Dronecode Foundation, "MAVLink Developer Guide," [Online]. Available: https://mavlink.io/en/. [Accessed 4 2 2024].

[12] A. KOUBÂA, A. ALLOUCH, M. ALAJLAN, Y. JAVED, A. BELGHITH and M. KHALGUI, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," *IEEE Access,* vol. 7, pp. 87658-87680, 2019.

[13] M. KUŘE, J. BUŠEK and T. VYHLÍDAL, "Swing compensation of a payload suspended to a planar copter,," in *23rd International Conference on Process Control (PC)*, Strbske Pleso, Slovakia, 2021.

[14] T. VYHLÍDAL, V. KUČERA and M. HROMČÍK, "Signal shaper with a distributed delay: Spectral analysis and delay," *Automatica,* vol. 49, no. 11, pp. 3484-3489, 2013.

[15] M. Hofreiter, Základy Automatického Řízení, Praha: České vysoké učení technické, 2012.

[16] T. VYHLÍDAL and M. HROMČÍK, "Parameterization of input shapers with delays of various distribution," *Automatica,* vol. 59, pp. 256-263, 2015.

[17] D. BUKOVSKÝ, "Tlumení výkyvu závaží zavěšeného na dronu", Praha: České vysoké učení technické, 2019.